This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE P2P 2010 proceedings.

Hivory: Range Queries on Hierarchical Voronoi Overlays

Matteo Mordacchini*, Laura Ricci†, Luca Ferrucci†, Michele Albano‡, Ranieri Baraglia*
*ISTI, CNR, Pisa, Italy
e-mail: {m.mordacchini,r.baraglia}@isti.cnr.it
†Dip. di Informatica, Univ. di Pisa, Pisa, Italy
e-mail: ricci@di.unipi.it,lferrucci@gmail.com
‡Inst. de Telecomunicações, Campus Univ. de Santiago, Aveiro, Portugal
e-mail:michele@av.it.pt

***Abstract*—The problem of defining a support for multidimensional range queries on P2P overlays is currently an active field of research. Several approaches based on the extension of the basic functionalities offered by Distributed Hash Tables have been recently proposed. The main drawback of these approaches is that the locality required for the resolution of a range query cannot be guaranteed by uniform hashing. On the other way, locality preserving hashing functions do not guarantee a good level of load balancing. This paper presents Hivory, a P2P overlay based on a Voronoi tessellation defined by the objects published by peers. Each object is mapped to a site of the Voronoi tessellation and the corresponding Delaunay Triangulation defines the P2P overlay. A hierarchy of Voronoi diagrams is defined by exploiting clusters of objects paired with the same site of the Voronoi diagram. A new Voronoi diagram including the peers of the cluster is created so that the query resolution may be refined by a top down visit of the Voronoi hierarchy. The paper presents the proposed solution, analyses its complexity, and provides a set of experimental results.**

I. Introduction

Peer-to-Peer (P2P) networks have been investigated in many scenarios. In particular, in recent years P2P applications have been used in new contexts such as online gaming, multimedia resource sharing and distributed computing. These types of applications have a set of common characteristics. In particular, they all use a representation of resources by multiple attributes, and need to provide resource finding methods through structured selection criteria that include range queries over more than one attribute.

Classical structured networks based on Distributed Hash Tables (DHTs) offer high performance in searching exact, single-attribute values, but they are not suitable for handling multi-attribute range queries. In the literature, several proposals try to address the problem of enhancing DHT overlays in order to support multidimensional range queries [1], [2], [3], [4], [5]. The majority of these solutions have high maintenance, high replication costs or need a high number of messages to solve range queries, especially when conditions over related attributes are not sufficiently selective.

Some other solutions for placing and retrieving objects in P2P networks are based on the Voronoi tessellations [6], [7], [8], [9] of the space. Examples of such networks are VoroNet[6], SWAM-V[8] and VoRaQue[7]. Peers use their attribute values as coordinates in the space. Each point of the space is assigned to the closest peer, according to some notion of distance. The set of points that are assigned to a peer P constitutes the *cell* of P. If two cells are adjacent, their related peers are linked. The resulting network is also called the Delaunay Triangulation of the peers. Since peers with similar attributes are close in the network, data locality is preserved, and it can be used to find all/some objects that are located into a region, also called Area of Interest (AoI). Moreover, locality constitutes an advantage for fault tolerance with respect to DHT-based approaches, since object insertion/removal causes changes only on the object neighborhood. Voronoi-based approaches have been proposed on 2-dimensional space, since higher dimensionality leads to an impractical runtime for the distributed node insertion algorithm [6].

In order to take advantage of the features of Voronoi networks and give a support for multi-attribute searches, in this paper we propose *Hivory* (HIerarchical VOronoi Range querY), a Voronoi-based solution able to efficiently perform multi-attribute (i.e. greater than two) range queries in P2P networks. The proposed solution is based on a multilevel hierarchical structure which takes the form of a tree of Voronoi planes; each level maps a different pair of attributes, and each node of the tree is constituted by a Voronoi two-dimensional plane. For each cell in a Voronoi plane, there exists another Voronoi plane in the underlying level, based on another pair of attributes. Using two-dimensional networks as nodes of the tree, at each level greedy and Compass Routing [7] can be exploited. Such algorithms were modified to support new procedures for publishing an object and to solve a multi-attribute range query. Moreover, a clustering mechanism based on the Voronoi space subdivision and the *absorption radius* mechanism are exploited at each Voronoi plane of the Hivory's levels. This mechanism allows to further limit the number of messages required to solve queries. To have an upper limit on the number of neighbors, for each cluster a number (that depends on the cluster size) of peers are elected Superpeers. They are the only peers that are "visible" (i.e. can interact) in the Voronoi network the cluster belongs to. They act as links between successive levels of the *Hivory* network.

We performed theoretical analysis and experimental simu-

978-1-4244-7141-6/10/$26.00 ©2010 IEEE

lations of the proposed architecture. They show that Hivory has very good scalability properties on both the number of nodes and the number of attributes. In particular, it exploits the combination of attribute selectivities to solve multi-attribute range queries, thus requiring a very limited number of messages. Moreover, it comes out that the mapping order of object attributes into the network level does not affect the system performance.

The remainder of this paper is organized as follows. Section II discusses some approaches proposed in the literature for supporting range queries. Section III describes the proposals based on Voronoi overlays. The problem of implementing multi-attribute range queries on Voronoi networks is discussed in Section IV. The architecture of *Hivory* is described in Section V, while Section VI shows its operations. The experimental results are presented in Section VII. Finally Section VIII reports some conclusions.

II. Related Work

DHTs have been the first solutions for indexing resources in P2P networks. *Chord, CAN, Pastry* ([10], [11], [12]) are among the most popular DHTs. Although very efficient, this kind of solutions supports only *single attribute exact match query*. Besides, the cost of maintaining the network in presence of high churn is higher than in the unstructured P2P networks, and the consistency of the network is not guaranteed in case of high churn rate.

Recently several proposals for the resolution of *multi-attribute range queries* have been proposed. Some proposals are based on a hierarchical subdivision of the space of the attributes. In [1] two methods are proposed, called $SCRAP$ and $MURK$, respectively. $SCRAP$ adopts a *space-filling curve* as an hash function to map multiple dimensions into a 1-dimensional index, split among peers. $MURK$ partitions the multi-dimensional space into hyper-rectangles that are assigned to peers. Routing is based on skip pointers to random nodes in the network. $ZNet$[13] partitions a multi-dimensional space using the *z-curve*, and then maps the partitioned zones to nodes organized in an overlay network based on *skip graphs*.

The *Multi-Attribute Addressable Network, MAAN* [2] is a structured P2P system targeted to the definition of a support for resource discovery in a large distributed systems that provides support for multi-attribute range queries. It modifies the Chord [2] hashing function by a *locality preserving hashing function*. In $MAAN$ resources are identified by a set of attribute-value pairs, and each attribute is mapped on a $Chord$ ring through the locality preserving function. The node target of the hashing function stores the full resource description so that a resource is stored as many times as its number of attributes. The resolution of a multi-attribute range query consists in executing a single 1-dimensional query on the dominant (i.e. most selective) attribute, while the other attributes are checked using the replicated data. The main drawbacks of *MAAN* are related to the large amount of memory required to store the resources, and to the cost to update the attribute values, which makes *MAAN* not suitable to support dynamic attributes.

In *SWORD* [3], nodes are divided into two categories: *reporting nodes* sending periodic updates of the resource values and *DHT Server nodes* that actually form the DHT overlay, and receive and store resource values and handle users queries. The possible values of each attribute $A_i \in A = \{A_1, \ldots, A_n\}$ are stored into contiguous regions of the DHT key space using a proper mapping function f_{A_i}. A range query on an interval $[x_{min}, x_{max}]$ of values of A_i can be solved by querying all the nodes holding the keys in the range $[f_{A_i}(x_{min}), \ldots f_{A_i}(x_{max})]$.

Mercury [4] is a multi-level DHT network based on *Symphony*. It uses a different DHT (called *Hub*) for each attribute. Each resource registers *all* the values of its attributes in each corresponding Hub. This replication of information is required in order to reduce the messages needed to solve a multi-attribute query. In fact, only the Hub with the lowest requested range is queried. Since attribute values are stored in each DHT using locality preserving functions, range queries are solved proceeding from the lowest value in the range to the biggest one. The resources matching all the request attribute constrains are finally sent to the user.

Andrzejak et al.[5] extends the *CAN* architecture [11] in order to support multidimensional range queries. A proper *CAN DHT layer* is used separately for each attribute. In each layer, nodes (called $IntervalKeepers$) handle a subinterval of the corresponding attribute domain. Values are stored as *<attribute-value, resource ID>* pairs. Each subquery of a multiattribute query is resolved separately and results are intersected at the querying node, in order to obtain the final list of matching resources.

All the above approaches suffer of at least one of the following problems: some use one DHT per attribute, with related maintenance costs; some exploits the most selective attributes in order to limit the diffusion of queries, but require high replication costs; otherwise, queries are sent to all the DHTs, or to regions of a DHT corresponding to the attributes specified in a multi-attribute query. In the latter case, a higher number of messages is required to solve a query. The following section will survey a set of proposals, which are more close to our approach because they exploit Voronoi-based overlay.

III. Voronoi Based P2P Overlays

The first proposal of a Voronoi-based overlay is that of *Voronet* [6]. Instead of using hash functions to distribute data objects, the Voronet P2P network maps each object to a 2-dimensional attribute space where the values of two attributes of the object specify its coordinates in the space. A Voronoi-based tessellation of the plane is defined such that each site in the tessellation corresponds to an object. We recall that, given k sites, a Voronoi tessellation partitions the plane into k areas such that the area corresponding to a site s includes all the points of the plane which are closer to s with respect to any other site. Two sites are Voronoi neighbours iff the borders of their areas overlap. The connected graph defined by linking neighbour sites is the Delaunay Triangulation associated to the Voronoi tessellation. Voronet defines a P2P overlay where

a link is paired with each Delaunay link. Furthermore, a *Small World* [14] overlay is defined over the Delaunay overlay by introducing a set of Long Range Neighbours links providing the small-world characteristics. Finally, each peer maintains a set of close neighbours whose distance from the object is smaller than a predefined threshold depending on the number of peers in the overlay. These neighbours are required to ensure a poly-logarithmic routing cost, even in presence of the crowding phenomena, i.e. situations where a lot of objects are located in the same region of the space. In [6] the Voronet routing algorithm is described and complexity bounds both for the routing tables size and for the routing are given. The main drawbacks of this approach is that it can be applied to 2-dimensional Voronoi space only, and that do not support range queries.

The Small World Access Methods ($SWAM$) [8] is a family of distributed access methods to efficient execute different kinds of complex queries like range and k-nearest neighbors queries. These methods guarantee that the time necessary to find an object defined by an exact query is logarithmic with respect to the size of the network. Furthermore, all the similar objects would be located in neighboring nodes. In [8] a Voronoi-based SWAM to solve range queries is proposed. It operates in two steps. In the first one, the range query is considered as an exact-match query for a point chosen in the AoI of the query. In the second step flooding is adopted to propagate the query to all the nodes of the AoI. The related computational cost is $O(\log N)$ for exact-match queries, and $O(\log s + sN)$ and $O(\log N + k)$ for range queries (with N number of the network nodes, k number of objects on the AoI and s the object selectivity) and k-nearest neighbours, respectively. The main drawback of this approach is the high cost introduced by flooding.

[7] optimizes the notification of a query within its AoI by introducing Compass Routing [15], [7], which is based on the following simple observation. Consider a connected graph and assume of being located at one of its nodes n with the goal to reach a destination node d. [15] shows that the best strategy is to look at the edges incident in n and choose the edge whose slope is minimal with respect to the segment connecting n and the destination d. [15] also shows that while Compass Routing is not cycle free for general graphs, it can always find a finite path between two nodes of a Delaunay Triangulation. [16] suggests to exploit Compass Routing to define a Spanning Tree supporting an application level multicast. [7] exploits this strategy to propagate a query within its AoI.

IV. Multidimensional Range Queries on Voronoi Overlays

Here and in the following section we will consider, for the sake of simplicity, a one-to-one mapping between objects and peers so that the terms peer and object will be used in an interchangeable way. The results can be easily extended to the case where a peer publishes several objects. We will exploit the notation $Object(p)$ to refer the object published by the peer p, and $Peer(o)$ to refer the peer paired with the object o.

In Section III we have described a set of Voronoi based approaches for supporting range queries. The main problem for the application of these techniques to real scenarios is the *curse of dimensionality*, which affects high dimensional Voronoi diagrams. As a matter of fact, while the number of Voronoi neighbors is $O(1)$ for Voronoi tessellations in d=2 dimensions, it grows as $O(n^{\lceil \frac{d}{2} \rceil - 1})$ for $d > 2$.

A simple solution to the problem of the curse of dimensionality is to perform a dimension reduction by partitioning the set of the object attributes, and by pairing each subset with a different Voronoi diagram. A mapping based on subsets of the attributes is defined as follows:

Definition 1: Let us consider a set O of objects characterized by the attributes $A = \{a_1, ..., a_k\}$. Given $A' \subseteq A$, $|A'| = t$, $Voronoi_{A'}$ is a t-dimensional Voronoi diagram V such that

- each dimension of V is paired with an attribute of A'
- V includes a set S of sites such that $\forall s \in S\ \exists o \in O$ such that the coordinates of s are defined by the values of the attributes of $o \in A'$.

$map(o, Voronoi_{A'})$ denotes the site $s \in Voronoi_{A'}$ whose coordinates are defined by the values of the attributes in A' of o. Furthermore, we will exploit the notation $AoI(q, Voronoi_{A'})$ for the region of $Voronoi_{A'}$ defined by the constraints of the n-dimensional range query q, paired with attributes of A'.

Given a partition P of A, each object is paired with a site for each Voronoi diagram corresponding to the attributes in $p \in P$.

Definition 2: Given a partition P of the set A of the attributes

- $Voronoi(P)$=$\bigcup_{p \in P} Voronoi_p$
- $map(o, P) = \bigcup_{p \in P} map(o, Voronoi_p)$.

In the following we will be interested in *2D partitions*, i.e. partitions where each subset includes a single pair of attributes. Note that, since now each object o is paired with a set of sites belonging to distinct tessellations, $Peer(o)$ should join the overlay corresponding to each tessellation. This implies that a peer may receive messages from different overlays. As discussed in Section VI, this implies that each message should be uniquely tagged with the identifier of the overlay where it is propagated.

A range query on a set of partitioned Voronoi diagrams can be resolved according two alternative strategies. The first one visits in parallel all the $AoI(q, Voronoi_p)$ $\forall p \in P$. The matches for the query are computed by the *intersection* of the set of objects returned by each visit. As discussed in [2] each $AoI(q, Voronoi_p)$ may include a large amount of objects, which are not matches for q, because of attributes not belonging to p. As a consequence, the peer submitting

the query may be overwhelmed by a huge amount of useless notifications.

A different solution restricts the search of matches of q to the most selective AoI. Each peer in this AoI checks if its object matches the constraints on attributes not in partition p, and only in this case, it returns its object as solution of the query. Even if this approach reduces the number of useless notifications, the number of peers involved in the query resolution may be still much higher with respect to its matches. This is especially true when the selectivity of the AoI is not high and/or when the distribution of the objects is skewed so that a very densely populated AoI exists.

A further problem is that these strategies do not take into account the problem of *clustering*. A cluster of objects is paired with the same Voronoi site when a set of objects are characterized by the same values of the attributes corresponding to a Voronoi tessellation.

Definition 3: Let us consider a Voronoi diagram $Voronoi_p$ paired with a set of attribute $p \in P$ and a site s $\in Voronoi_p$
$Cluster(s, Voronoi_p) = \{o \in O \text{ such that } map(o, Voronoi_p) = s\}$

The problem of clustering often occurs in real applicative scenarios. For instance in a resource discovery framework, each peer publishes an object describing its characterizing features, like operating system, memory and disk size and so on. A large amount of peers characterized, for instance, by the same operating system and CPU types are paired with a single site of the tessellation. Note that the reduction of a n-dimensional Voronoi diagram to a set of 2D Voronoi diagrams increases the level of clustering, because the probability to find out objects characterized by the same attribute values is inversely proportional to the number of attributes exploited to map an object on a Voronoi site.

The main problem of clustering is that the number of neighbors of an object is not $O(1)$, even when 2D Voronoi diagrams are considered. This implies that both the size of the routing tables and the number of connections between neighboring peers are not bounded. Furthermore, since all the objects in a cluster are paired with the same site, it is not possible to exploit the Delaunay triangulation edges for the definition of a compass routing based construction of the multicast tree. A solution where a neighbor peer n sends the query to all the peers of the cluster is not feasible, because the huge amount of connections may exhaust its bandwidth. If n sends a query to a subset of peers, flooding is required to propagate the query to the other peers. This may introduce a large amount of redundant messages.

The following section describes an alternative solution, which tries to solve both the problem of useless notifications and of clustering.

V. Hivory: the Architecture

In this section we describe the architecture of Hivory. As shown in the previous section, a solution of both the clustering problem and of useless notifications problem is mandatory for the Voronoi approach to be applied in real scenarios.

In Hivory, a pair of attributes is associated with each level of the Voronoi hierarchy, and clustering is exploited to define the hierarchy of Voronoi diagrams.

Let us suppose each object being characterized by k attributes, with, for the sake of simplicity, k is fixed even. Moreover, let us define $max = \frac{k}{2}$. Hivory defines a hierarchy of at most l_i levels, with $i \in 1, .., max$, where a pair of different attributes is associated at each level. Whenever a cluster C of objects is paired with a Voronoi site s at level l_i, and the size of the cluster exceeds a predefined threshold T, a new Voronoi diagram V is defined at level l_{i+1}, and all the objects of C are mapped to V by exploiting the attributes paired with l_{i+1}. This implies that a set of Voronoi diagrams may be defined at level l_{i+1}, one for each cluster of level l_i. Note that a cluster paired with a site at level l_{max} does not generate a further Voronoi diagram because all the attributes have already been exploited to define the higher level diagrams.

In Hivory the query resolution process is refined step by step visiting in top down fashion the hierarchy of Voronoi diagrams. Whenever a query is forwarded to a peer of a cluster, it does not broadcast the query to the other peers, instead it switches the query to the Voronoi diagram at the next level of the hierarchy. This way, at each step, the AoI paired with the next level is exploited to reduce the number of candidate matches for the query so that this number decreases step by step. Note that the cost of the creation of a new Voronoi diagram may be balanced by the reduction of the candidate matches by choosing a proper threshold T.

Hivory exploits clustering to distribute the load of query forwarding as well. When a query reaches the neighbor of a cluster, it chooses at random a peer of the cluster to forward the query. In this way, load is balanced because different queries are statistically distributed to different peers.

Let us formally describe the hierarchy of Voronoi diagrams. The first step is the definition of a proper ordering of the attributes so that each level of the hierarchy corresponds to a pair of attributes.

Definition 4: Let us consider an ordered sequence S of the attributes $A = \{a_1, ..., a_k\}$ of a set O of objects

- the *attributes* A_{li} of *level* l_i of O, with $i \in 1, .., max$, are those whose rank in S is, respectively, $2 * i - 1$ and $2 * i$. For each level l_i, the attribute ranked $2 * i - 1$ is paired with the x coordinate of the Voronoi diagram of level l_i, that ranked $2 * i$ with the y coordinate.
- given a level l_i, with $i \in 1, .., max$, and an object $o \in O$
  - if $i \neq 1$ $Label(o, l_i)$ is the ordered sequence of the values of the attributes A_{lj} of o, for $j \in 1..i-1$
  - if $i = 1$ $Label(o, l_i)$ is an empty sequence.

A Voronoi diagram V at level l_i corresponds to a cluster of objects paired with a site of level l_{i-1}. By recursively applying this property, we can note that all the objects mapped on a Voronoi diagram at level l_i share the same value for the attributes paired with the diagrams at higher levels of the

hierarchy. As a consequence, it is possible to uniquely identify a Voronoi diagram at level l_i by a label id corresponding to the values of the attributes of the upper level of the hierarchy which characterize all its objects. The following definition formally describes $Voronoi(l_i, id)$, the Voronoi diagram of level l_i labeled id.

Definition 5: Given a set O of objects with k attributes, a level l_i, with $i \in 1, .., max$ and a sequence id of $(i-1)*2$ values, $Voronoi(l_i, id)$ is a 2D Voronoi diagram such that

- the dimensions of $Voronoi(l_i, id)$ are paired with the attributes of O of level l_i
- $Voronoi(l_i, id)$ includes a set S of sites such that $\forall s \in S \ \exists o \in O$ such that
  - $s = map(o, Voronoi(l_i, id))$
  - $Label(o, l_i) = id$
  - $\forall j \in (1, .., i-1)$, $|Cluster(s', Voronoi(l_j, Label(o, l_j)))| > T$ where $s' = map(o, Voronoi(l_j, Label(o, l_j))$

Note that, if $|Cluster(s, Voronoi(l_i, id))| > T$ and the value of the coordinates of s are, respectively, x_s and y_s then $Voronoi(l_{i+1}, id')$ is created, where $id' = id \cdot x_s \cdot y_s$ The resulting hierarchy may be described by a tree whose nodes correspond to Voronoi tessellations and each node n paired with the Voronoi Diagram V has a set of sons corresponding to the sites of V where a cluster of size greater than T occurs.

Example 1: Consider a Voronoi Diagram at level 1 $Voronoi(1, \emptyset)$, and one of its site s whose coordinate are $x = 8, y = 7$ and suppose that a set S of more than T objects is paired with s. Then a Voronoi diagram $Voronoi(2, 8 \cdot 7)$ is created, i.e. a diagram of level 2 whose sites are all characterized by having at level 1 the x-coordinate equal to 8 and the y-coordinate equal to 7. $8 \cdot 7$ is the id which uniquely identifies the diagram among those of the same level of the hierarchy.

The Voronoi hierarchy is exploited to define the *Hivory P2P overlay*. Each Voronoi diagram defines an overlay whose links correspond to the edges of the corresponding Delaunay triangulation. Note that because of clustering, a single edge of the Delaunay Triangulation may correspond to a set of links in the overlay, because the sites connected by that edge are paired with clusters of objects. To reduce the number of neighbor peers Hivory exploits an *election mechanism*. Suppose that a cluster C occurs at a site of $Voronoi(l_i, id)$. Hivory elects a subset of m *representative peers* of the cluster C, $m = log(|C|)$. Only these peers belong to the overlay at level l_i, i.e. are *visible* at level l_i. This strategy guarantees a logarithmic bound on the number of neighbours of each peer, as shown by the following theorem.

Theorem 1: Let us consider a given peer p paired with a site s at any level of the hierarchy. If k is the size of the larger cluster paired with a Voronoi neighbor site of s, then the number of neighbours of p is O(log(k)).

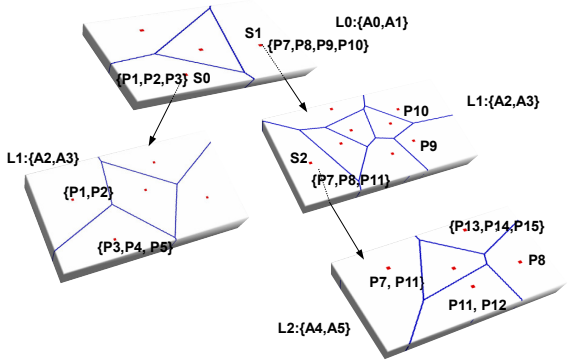

Fig. 1. Network organization and SuperPeer Election in a Hivory Network

The proof of the theorem is given in [17].

Each peer which is elected as representative of a cluster acts as a *gateway* between different levels of the hierarchy. i.e. it is responsible of propagating the query to the lower level of the hierarchy. The search for candidate matches for the query is carried on at this level by considering only the peers which belong to the AoI defined by its attributes. Peers which are chosen as representative of the cluster, i.e. the SuperPeer of the cluster, are chosen among those characterized by high computational power and bandwidth because they should support a high load of traffic since they are visible on a set of overlays.

Note that if a peer has been elected as Superpeer at level l_i, it must be visible at level l_{i+1} as well so that it can act as a gateway between these levels. For this reason, as discussed in more details in section VI, a peer should be elected Superpeer at level l_i if and only if it visible at level l_{i+1}. This property is guaranteed by Hivory by a bottom up election of the SuperPeers. [17] shows that is always possible to elect from an underlying level, a number of SuperPeers equal to the logarithm of the size of the cluster.

Figure 1 shows an example of Superpeer election. In the example, we consider objects characterized by 6 attributes, so that a hierarchy of 3 levels is defined. The value of the threshold is 4, i.e. a new Voronoi Diagram is created when a cluster includes more than 4 peers. For each site we show the peers paired with it. For the sake of simplicity, we do not report the peers associated to some sites which are paired with a single object. Note that since the size of the cluster paired with the site S_1 is 16, because it includes both the objects mapped at level L_1 and those mapped at level L_2, 4 SuperPeers have been elected and paired with S_1. Instead only the objects mapped at level L_2 must be considered when electing the SuperPeers paired with S_2.

Finally, it is worth noticing that clustering is exploited by Hivory also to decrease the cost of query resolution in regions including a huge amount of close sites, i.e. in crowded regions.

In this case, a the resolution of a query may require a large amount of routing hops, even when the AoI is highly selective. Hivory increases clustering of objects in those regions in order to factorize the query resolution process. The strategy adopted to increase clustering is the following. Whenever a new object o is inserted into a Voronoi tessellation at any level, Hivory checks the distance of the object o to the closer Voronoi site s and if it is smaller than a predefined threshold, the mapping of the object is forced so that o is paired with s. In the following, such a threshold will be referred as absorption radius of a site. Note that the size of the cluster paired with s increases, but since the number of peers visible at s is logarithmic with respect to the number of peers in the cluster, the amount of peers in the crowded region decreases.

VI. Hivory: The Operations

This section describes the join/leave operation exploited by a peer to join/leave the *Hivory* hierarchy and the support defined for the *multi-attribute range query*. First we briefly define the main data structures exploited by a peer p for managing the overlays it is part of. p maintains a data structures for each Voronoi network of every level in which p is visible, ranging from the *insertion*(lower) *level* to the upper level where p is visible. Each element of these structure stores the neighbours of p at the corresponding level, i.e. its *Voronoi, close and long range neighbours*. Furthermore, for each level where a cluster occurs p stores a reference to all the other peers of the cluster. Finally, p stores a reference to the *SuperPeers* corresponding to the upper level where it is visible.

A. The Join/Leave Operation

The first step performed by a peer p joining a *Hivory* Voronoi diagram at level l_i, $Voronoi(l_i, id)$, is the detection of the insertion point of the peer. This point falls into the Voronoi region of an already existing site s and its coordinates may or not be within the *absorption radius* from s. In the former case, a cluster is created at s if a single peer is associated with s; otherwise p joins the existing cluster in s. In the latter case a new Voronoi region is paired with a new site corresponding to p_i. In both cases, a *bootstrap peer* is exploited to propagate a *join message* in the overlay at level l_i. The target of this message is one of the peers paired with s. Let this peer be p_m, the manager of the insertion of p in the overlay. The procedure executed by p_m when the coordinates of the insertion point is not within the absorption radius of s is the same exploited in [6] and will not be discussed here.

Let us now continue the discussion with the analysis of the scenario where the coordinates of the insertion point are within the absorption radius of s. In this case, the following cases may occur:

- $|Cluster(s, Voronoi(l_i, id))| < T$. p joins the existing cluster. No further level of the Voronoi hierarchy is created.
- $|Cluster(s, Voronoi(l_i, id))| = T$. A new Voronoi diagram of level l_{i+1} is created where all the peers belonging to $Cluster(s, Voronoi(l_i, id)$ are mapped.
- $|Cluster(s, Voronoi(l_i, id))| > T$. The join request of p is propagated on the Voronoi overlay already existing at level l_{i+1}.

Let us now discuss each of the previous cases in more details.

In the first case, the only task of p_m is to notify the identity of p to all the peers of the cluster and the other way round.

In the second case, the threshold T has been reached because of the insertion of p so that a new level of the hierarchy should be created. p_m acts as a bootstrap peer for the creation of the new Voronoi Diagram V at the lower level: first it joins V, then it propagates within V a join request tagged by the new level l_{i+1} for each peer q of the cluster. The request is propagated till it reaches the insertion point of q at level l_{i+1}. Note that if a new cluster is created at level l_{i+1} then the procedure ia recursively applied. The last step is the election of the *SuperPeers* for the level just created. To reduce the cost of the bottom up election, *Hivory* elects the $T-1$ peers which belonged to the cluster before the insertion of the new peer as *SuperPeers* of the Voronoi overlay just created. Note that, in this case, the underlying level includes exactly T peers and $T \approx log(T)$ whenever T is small.

Finally, consider the case where $|Cluster(s, Voronoi(l_i, id))|>T$. In this case, a Voronoi diagram already exists at level l_{i+1}, therefore the join message is switched by p_m to this level and it is recursively propagated until the *insertion point* of the peer in l_{i+1} is reached. Note that the level tag in the join message is incremented each time the message is switched to a Voronoi overlay at a deeper level. At the end of the recursive insertion, if the the insertion level of the new peer p is $\neq 1$, p has joined a set of clusters whose size was larger than the threshold before its insertion, one for each level between 1 and its insertion level. The condition on the logarithmic number of SuperPeers may be non more satisfied for some cluster because of the insertion of p so that the a bottom up election may be required. The election starts at the insertion level of p and performs a backward visit of all the clusters in order to check the need of new SuperPeers.

Let gw_{l_i} be the *gateway peer* which has propagated the join message of p form level i to level $i+1$. The identity of gw_{l_i} is stored in the join message sent to the overlay of level $i+1$ so that a further gateway peer $gw_{l_{i+1}}$ may store the identity of gw_{l_i} before propagating the join message to a further level. In this way, a *distributed chain of backward links* is defined to implement the bottom up election. When the insertion of the new peer p finishes, a message notifying its completion is sent back over this chain so that each gw_{l_i} is able to check the logarithmic bound on the peer of its cluster and, if necessary, to provide for the election of a new SuperPeer. This should be chosen among the peers visible at level l_{i+1} which are not already SuperPeers of the cluster at level l_i. The new peer p is elected SuperPeer if it belongs to l_{i+1} because this is its insertion level or because it has been elected *SuperPeer* for this level. Otherwise, a new *SupePeer* is found out by a *distributed algorithm*. gw_{l_i} contacts its *Voronoi*

neighbours in the underlying level to check if one of them is not a *SuperPeer* of the cluster at level l_i. Each Voronoi neighbour which is already a *SuperPeer* for level i contacts in turn its neighbours, and the message is propagated until an eligible peer is discovered.

Let us now give an analysis of the cost of finding the proper point for p in the recursive insertion process. Suppose this point is placed in a Voronoi network at the last of M levels. At every level, a *greedy routing* is required to find the cell on that level that is associated with p. Suppose that we are in the case of a uniform distribution of both the attribute values and of the nodes among clusters at all levels. Thus, if there are N nodes, and the first level has N_1 clusters (corresponding to N_1 cells), the subtree rooted at each cluster contains N/N_1 peers. The same situation is replicated in every level, till the last one, where every node has its own cell. Supposing that there are M levels, we have that $N = \prod_{i=1}^{M} N_i$.

Remember that a routing on a 2D Voronoi plane requires $O(\log^2 N)$ messages [6], the cost of finding the right cells across all the levels is then

$$\log^2 N_1 + \log^2 N_2 + \ldots + \log^2 N_M \tag{1}$$

With the given assumptions, we have that

$$\log^2 N = (\log(\prod_{i=1}^{M} N_i))^2 = (\log N_1 + \ldots + \log N_M)^2 \geq$$

$$\geq \log^2 N_1 + \log^2 N_2 + \ldots + \log^2 N_M \tag{2}$$

Thus, in this situation the complexity is even lower than the 2D case. Let us now consider the case of an unbalanced *Hivory* system, where there is only one network x at a level k that has a number of cells $N_x \approx N$. If we are in a case where $N_i \ll N, \forall i \neq x$, the complexity is

$$\log^2 N_1 + \ldots + \log^2 N_x + \ldots + \log^2 N_M \rightarrow \log^2 N_x \approx \log^2 N \tag{3}$$

In this case the complexity is approximately the same of a 2D network.

As pointed out in[6], a *leave* operation on a 2D Voronoi space perturbs only the neighborhood of a node. In *Hivory*, the majority of the nodes are inserted in one network only. As pointed out above, the only nodes allowed to be visible in more than a network are the SuperPeers. Moreover, nodes are grouped into clusters. Thus, usually, when a node leave the network, this communication is sent the other nodes of its cluster. In the case that its departure involves the removal of its Voronoi cell (i.e. it was the only node in that cell), the neighbors of the cell have to be informed. Those operations are performed locally, in the immediate surroundings of the leaving node. Thus they have only a limited impact and their complexity depends only on the number of nodes in a cluster and on the number of neighbors. In the case of a SuperPeer, the previous operations are repeated in all the networks the SuperPeer belongs to. Moreover, an election process like the one described above is needed in each of such overlays. In any case, only few nodes are SuperPeers and the number of SuperPeers that are visible in more than a network decreases accordingly with the number of levels involved.

B. Multidimensional Range Query Resolution

A range query may be submitted by any peer belonging to the *Hivory* hierarchy. Note that this peer could not necessarily be visible at level l_1 of the hierarchy. In this case, the first step of the query resolution is its propagation to a peer belonging to l_1. Since each peer stores a reference to its SuperPeer of the upper level, these references may be exploited for a bottom up forwarding of the query to the level l_1. When the query has reached a peer at level l_1, *Hivory* exploits *greedy routing* to reach any peer belonging to the AoI defined by the constraints on the attributes corresponding to this level and *compass routing* to propagate the query within the AoI.

When a peer belonging to the AoI and to a cluster whose size is smaller than the threshold (i.e. it has no underlying networks) receives the query, it matches the attributes not mapped on the lower levels of the hierarchy against the corresponding constraints of the query and, if all the matches are successful, it sends a positive reply to the querying peer. The peer propagates the query to the other peers of the cluster as well. Instead, if a peer p belonging to the AoI and to a cluster whose size is larger than the threshold receives the query, it makes it own local check and then it switches the query to the lower level of the hierarchy. Note that the query is not propagated to the other peers of the cluster, because they will receive the query at that level only if they belong to the AoI defined by the constraints of the attributes of this level. In both previous cases, the query is forwarded to the peers paired with the neighbour sites, through compass routing. This process is iterated in all the underlying networks of the contacted clusters. Thus, messages are propagated toward the related $AoIs$, forwarded inside them, and propagated down to the other levels. The main points of this mechanism is illustrated in Algs. 1 and 2, that describes the $P.Forward$ method of the previous algorithm.

Algorithm 1 Query resolution process

```
Let Q be a query and P a peer that receives it al level L_i
if P.UpperLevel ≠ 1 then
    Get peer S from P.SuperPeers
    Send Q to S
else
    P.Forward(Q, L_i)
end if
```

In order to better understand the behavior of *Hivory* when solving multi-attribute queries, we give an analysis of the complexity of the mechanism described above.

We want to give an estimation of the global number of nodes that has to be contacted inside all the AoIs defined by a query Q. Let us then consider the number of nodes contacted by the *compass routing* algorithm. As for the join operation, let us

Algorithm 2 Query forwarding process at level L_i

```
if P.Cell(L_i) ∩ Q.AoI(L_i) ≠ ∅ then
  if ¬P.HasProcessed(Q) then
    P.LocalMatch(Q)
  end if
  P.CompassRouting(Q, L_i)
  P.SendToCluster(Q, L_i)
  if ∃P.Cell(L_{i+1}) then
    P.Forward(Q, L_{i+1})
  end if
else
  P.GreedyRouting(Q, L_i)
end if
```

consider the case of a uniform distribution of both the attribute values and of the nodes among clusters at all levels. Hence, if we suppose that there are M levels, we have that $N = \prod_{i=1}^{M} N_i$. If a plane at a level L_i has a combined selectivity s over its attributes, $s * N_i$ nodes will be involved by the query. The total number of requested nodes is

$$\sum_{i=1}^{M} \prod_{j=1}^{i} s_j N_j \tag{4}$$

Note that each N_i can be written as $N_i = \frac{N}{\prod_{j=1, j\neq i}^{M} N_j}$. Thus, we can express Form.4 as:

$$N * (\prod_{i=1}^{M} s_i + \sum_{i=1}^{M-1} \frac{\prod_{j=1}^{i} s_j}{\prod_{k=i+1}^{M} N_k}) \tag{5}$$

Please note that, $\forall i \in [1, M-1]$, we have that $\frac{\prod_{j=1}^{i} s_j}{\prod_{k=i+1}^{M} N_k} \leq \prod_{j=1}^{M} s_j$. In fact, we can re-write the inequality as $\prod_{j=1}^{i} s_j \leq \prod_{j=1}^{M} s_j \prod_{k=i+1}^{M} N_k$. We have that $\prod_{j=1}^{i} s_j \leq \prod_{j=1}^{i} s_j \prod_{k=i+1}^{M} s_k \prod_{k=i+1}^{M} N_k$, i.e. $\prod_{j=1}^{i} s_j \leq \prod_{j=1}^{i} s_j \prod_{k=i+1}^{M} s_k N_k$, that is tautologically true. Thus, we can write that

$$N * (\prod_{i=1}^{M} s_i + \sum_{i=1}^{M-1} \frac{\prod_{j=1}^{i} s_j}{\prod_{k=i+1}^{M} N_k}) \leq N * M * \prod_{i=1}^{M} s_i$$

$$= M * s_1 \dots s_M * N \tag{6}$$

In a uniform distribution of values and peers in the space, the complexity is the proportional to the number of levels and the *combined selectivity* of all the attributes.

Let us now consider the case of an unbalanced *Hivory* system, where there is only one network x at a level k that has a number of cells $N_x \approx N$. If the query is directed to x, the selectivities expressed at the previous levels do not have a great impact on the final global selectivity, since the most populated network is included in the range. Thus, since the majority of nodes is concentrated in x, the highest cost in term of visited nodes is paid in x. Hence, we can assume that the cost of such a query is $O(s_k * N_x)$, where s_k is the combined selectivity of the attributes at level k, the same of x.

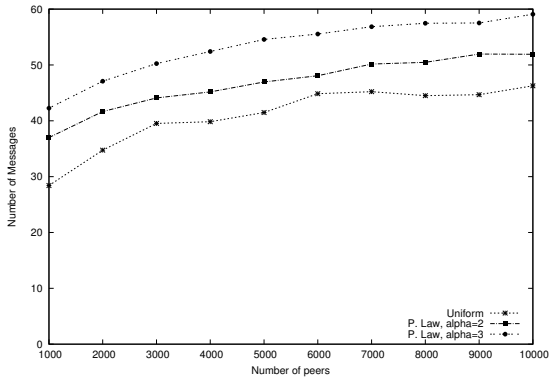

Fig. 2. Insertion Costs with 3 different data distributions when varying the network size

VII. Experimental Results

In this section we provide the experimental findings of the system described in the previous sections.

The evaluation of the proposed system was carried out using the PeerSim [18] simulator. In all the following experiments, we used the number of exchanged messages as the performance metric.

The system was tested under different conditions. We present the results related with the multi-attribute, range queries resolution and the maintenance cost of the network. In all those experiments we used the following assumptions and settings:

- each peer is associated to a multi-attribute vector of values; it can be regarded as a representative of the peer's resources, like the mean value or the sum of the values for each attribute of the peer owned resources;
- we used three different data distributions over the attributes' domains associated to the above values: a uniform distribution and two power-law distributions, one with $\alpha = 2$ and one with $\alpha = 3$. In these two last cases, values (and peers) are more concentrated in one end of each attribute domain than to the other, thus creating an unbalanced tree-shaped network.
- we varied the network size from 1,000 to 50,000 nodes

First of all, we provide an analysis of the cost of inserting new nodes into the network. The results are shown in Fig. 2.

In this experiment we evaluated the number of messages required to do a node insertion in a Hivory network of the given size. We performed 100 different insertions for each network size and took the final mean value. We can see that the number of messages scales well with the number of nodes and remains always very low, requiring less than 60 messages in the worst case, with the largest network size. The best

performance is obtained with a uniform distribution of the values among the peers. This happens because, in this case, nodes are also uniformly distributed in each Voronoi layer they belong to. Thus, they are more likely to have (i) less neighbors and (ii) less nodes falling in the same Voronoi cell. Hence, less communications are required to insert a node in this situation, while more messages are needed when dealing with more concentrated distributions, like the two power-law distributions in the figure.

The main focus of our experiments were the system performance in solving multi-attribute, range queries. In the following, we used the three value distributions described above and three different query selectivities. These selectivities are defined as a percentage of the area of a Voronoi plane obtained by a couple of attribute. Hence, if we have a 0.05 query selectivity over a couple of attributes, it means that the query covers an area that correspond to the 5% of the whole Voronoi plane formed by the attributes related to that plane. If the query has the same selectivity on both the attributes, thus each attribute will have approximately 22.361% ($\sqrt{0.05}$) of selectivity in its domain. Otherwise, we have that the lowest selectivity (highest percentage) will range in $[0.22361, 1]$, while the highest selectivity will range in $[0.05, 0.22361]$, accordingly. In the following, we collected results obtained by executing, for each network size, 100 random queries of different selectivies. Each query contains all the attributes and the selectivity on each pair of attributes is determined by randomly chosing the selectivity of one attribute of the couple and the by adjusting the second one in accordance with the global selectivity constrain. Queries are injected in randomly chosen nodes, at any level of the network. All the experiments are confronted with MAAN[2], a well-known P2P system for handling multi-attribute, range queries, described in Sec.II. In the experiment showed in Fig. 3, we measured the number of messages required for solving queries with the given selectivities, varying the number of the involved attributes. The network size is fixed to 50,000 nodes. The messages are the sum of the messages needed for routing a query toward related areas (i.e. Greedy Routing messages) and messages used to contact all the nodes within a requested zone (i.e. Compass Routing messages)

The results are presented in a log scale along the y axis in order to allow to see the differences in our system between the different selectivities, that were otherwise hard to see. MAAN uses the a DHT with the highest selectivity (i.e. smallest area) for each query. Thus, its results are almost independent from the number of attributes. On the contrary, as explained in the system architecture section, our system combines the selectivities of all the involved attributes, thus obtaining a more discriminating power. As a consequence, when more attributes are involved, less messages are required. The final result is that our system outperforms MAAN from one to two orders of magnitude.

After having measured the performance of the network when varying the number of attributes, we studied the system behaviour when the network size changes. In order to have a good number of attributes and levels we fixed the number of attributes to 6. The results, using the different data distributions with both Hivory and MAAN, are shown from Fig.4 to Fig.6.

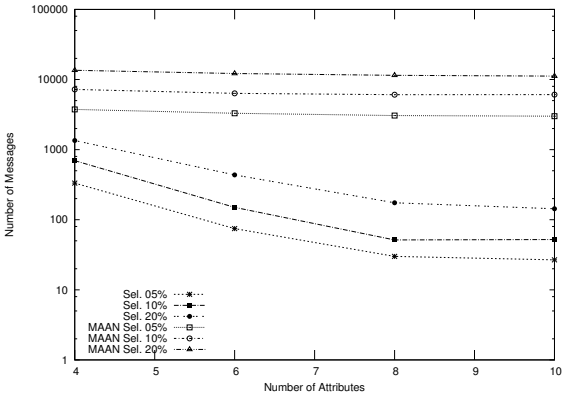

Fig. 3. (Logscale on y axis) Comparison between the proposed approach and MAAN when varying the number of attributes and using 3 different query selectivities (uniform distribution)

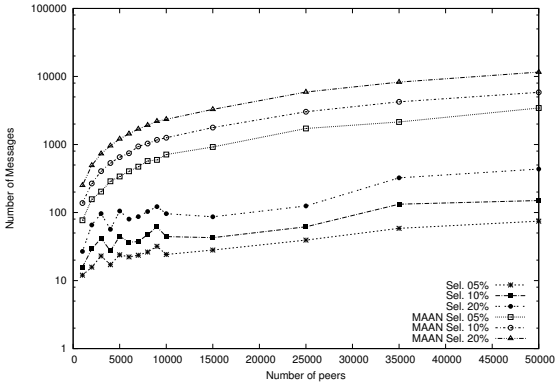

Fig. 4. (Logscale on y axis) System performance in solving multi-attribute, range queries when varying the network size and using 3 different query selectivities (uniform distribution)

We can see that the network scales well with the number of peers, maintaining a controlled number of messages with all the selectivities. The best performances are achieved when using power-law data distributions. In contrast with the insertion costs, since nodes are more concentrated in some areas they may collapse in one single cell, due to the absorption radius. Thus, it is more easy to contact all the nodes when a query falls into heavily concentrated areas. On the other hand, if a query requests less populated zones, we will have also few nodes falling into them, thus requiring less messages to solve the whole query. Also in this case, the combined selectivity used by *Hivory* allows to obtain a considerable gain with respect to MAAN.

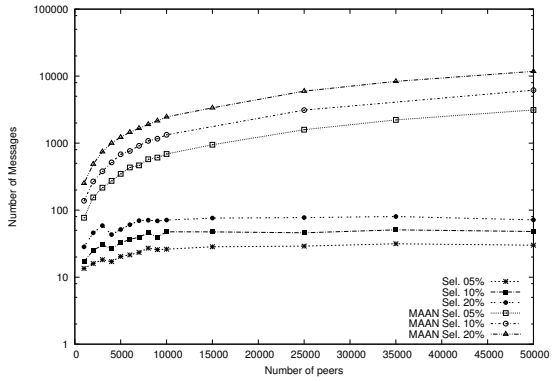

Fig. 5. (Logscale on y axis) System performance in solving multi-attribute, range queries when varying the network size and using 3 different query selectivities (power-low distribution, $\alpha = 2$)

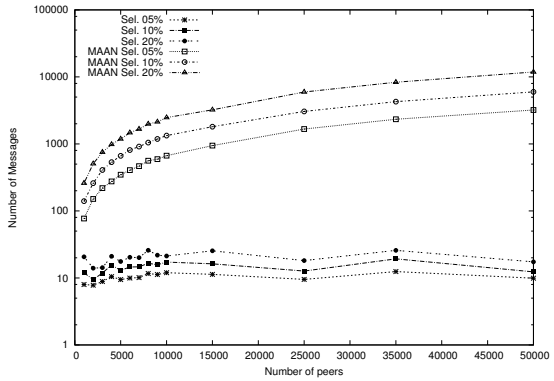

Fig. 6. (Logscale on y axis) System performance in solving multi-attribute, range queries when varying the network size and using 3 different query selectivities (power-low distribution, $\alpha = 3$)

VIII. Conclusions

This paper has presented *Hivory* a P2P support for multi-dimensional range query based on a hierarchy of Voronoi tessellations. The search of the matches of a query are performed by a top down visit of the hierarchy where the search space is restricted at each level. The analysis of the complexity of the main operations and the simulations results of the proposed approach show that the system requires a very limited number of messages to perform multi-attribute, range queries. The system combines the good properties of classical Voronoi-based networks and extends them to support high numbers of attributes. The analytical observations and the experimental results show a great scalability with both the number of nodes and the number of attributes. With respect to this last point, the ability of the system to combine the selectivity of all the attributes, allows *Hivory* to even decrease the number of messages when dealing when multiple attributes. As future works, we plan to develop a a set of experiments on a real network. Moreover, we plan to confront it with further systems and investigate the system performance in some applicative scenarios.

References

[1] P. Ganesan, B. Yang, and H. Garcia-Molina, "One Torus to Rule Them All: Multi-Dimensional Queries in P2P Systems," in *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*. New York, NY, USA: ACM Press, 2004, pp. 19–24.

[2] M. Cai, M. Frank, J. Chen, and P. Szekely, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services," in *GRID '03: Proc. of the 4th Int. Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 184.

[3] J. Albrecht, D. Oppenheimer, A. Vahdat, and D. A. Patterson, "Design and implementation trade-offs for wide-area resource discovery," *ACM Trans. Internet Technol.*, vol. 8, no. 4, pp. 1–44, 2008.

[4] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: Supporting Scalable Multi-Attribute Range Queries," in *Proc. ACM SIGCOMM 2004 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*. ACM Press, 2004, pp. 353–366.

[5] A. Andrzejak and Z. Xu, "Scalable, efficient range queries for grid information services," in *Peer-to-Peer Computing*, R. L. Graham and N. Shahmehri, Eds. IEEE Computer Society, 2002, pp. 33–40.

[6] O. Beaumont, A. marie Kermarrec, L. Marchal, tienne Rivire, and E. Lyon, "Voronet: A scalable object network based on voronoi tessellations," in *In Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007*. Society Press, 2007.

[7] M.Albano, M.Baldanzi, R.Baraglia, and L.Ricci, "Voraque: Range queries on voronoi overlays," in *In Proceedings of 13th IEEE Symposium on Computers and Communications*, July 2008.

[8] F. Banaei-Kashani and C. Shahabi, "SWAM: a family of access methods for similarity-search in peer-to-peer data networks," in *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*. New York, NY, USA: ACM, 2004, pp. 304–313.

[9] M. Albano, R. Baraglia, M. Mordacchini, and L. Ricci, "Efficient broadcast on area of interest in voronoi overlays," in *Proceedings of IEEE CSE'09, 12th IEEE International Conference on Computational Science and Engineering, August 29-31, 2009, Vancouver, BC, Canada*. IEEE Computer Society, 2009, pp. 224–231.

[10] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," in *Proc. SIGCOMM '01*. New York, NY, USA: ACM Press, 2001, pp. 161–172.

[12] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Middleware 2001: Proc. of the IFIP/ACM Int. Conf. on Distributed Systems Platforms, Heidelberg*. London, UK: Springer-Verlag, 2001, pp. 329–350.

[13] Y. Shu, B. C. Ooi, K.-L. Tan, and A. Zhou, "Supporting multi-dimensional range queries in peer-to-peer systems," *IEEE International Conference on Peer-to-Peer Computing*, vol. 0, pp. 173–180, 2005.

[14] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000, pp. 163–170.

[15] E.Kranakis, H.Singh, and J.Urrutia, "Compass routing on geometric networks," in *In Proceedings of 11th Can. Conf. on Computational Geometry, CCCG*, August 1999.

[16] J.Liebeherr and M.Nahas, "Application layer multicast with delaunay triangulations," *IEEE Journal on Selected Areas in Communications*, vol. 40(8), October 2002.

[17] L. Ferrucci, "Un sistema gerarchico basato su Voronoi per la risoluzione di query multiattributo," Master Thesis, University of Pisa etd-11102009-151747, Dec. 2009.

[18] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim simulator," `http://peersim.sf.net`.