Journal of Computer and System Sciences 82 (2016) 1161–1179

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Multidimensional range queries on hierarchical Voronoi overlays

L. Ferrucci [b,*], L. Ricci [a,b,*], M. Albano [d,*], R. Baraglia [b], M. Mordacchini [c]

[a] Department of Computer Science of the University of Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy
[b] Information Science and Technologies Institute of CNR (ISTI-CNR), Via Moruzzi 1, 56124 Pisa, Italy
[c] Institute of Informatics and Telematics of CNR (IIT-CNR), Via Moruzzi 1, 56124 Pisa, Italy
[d] CISTER, INESC-TEC/ISEP, Polytechnic Institute of Porto, Rua Dr. António Bernardino de Almeida 431, Porto, Portugal

ARTICLE INFO

Article history:
Received 13 March 2015
Received in revised form 18 March 2016
Accepted 8 April 2016
Available online 30 April 2016

Keywords:
Distributed systems
Range queries
Voronoi

ABSTRACT

The definition of a support for multi-attribute range queries is mandatory for highly distributed systems. Even if several solutions have been proposed in the last decade, most of them do not meet the requirements of recent platforms, like IoT or smart cities. The paper presents an approach that builds a multidimensional Voronoi graph by exploiting the attributes of the objects published by a node. Our solution overcomes the curse of dimensionality issue affecting Voronoi Tessellations in high dimensional spaces by defining a Voronoi hierarchy. The paper formally defines the structure, analysis the complexity of the operations and presents experimental results.

1. Introduction

The recent diffusion of smart-"things" such as smart-phones and RFID tags has produced the diffusion of new computing platforms like cyber-physical networks, smart-cities platforms, which can be generally grouped under the broader definition of the Internet of Things (IoT) [1]. In this context, an important building block for IoT infrastructures is the ability to sustain widely distributed, dynamic, and autonomic services [2], in particular an efficient, scalable and adaptable discovery service is essential.

Most IoT services share 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. A support for the resolution of multi-attribute range queries is therefore mandatory in these systems to support several higher level services. Consider, for instance, a scenario where mobile users issue location-based range queries, in order to discover points of interest in a geographical region. For instance, a user may submit a query to find out all the restaurants located in a given region with desired features such as average price of a meal, seating capacity etc. The corresponding query may include constraints on the spatial location of the restaurant and on an additional set of attributes.

Recent distributed computational platforms proposed for IoT, like those based on Fog computing [3], share several common traits with the peer-to-peer computational paradigm. As a matter of fact, these platforms require to run services throughout the network and the result is that intelligence is not localized on centralized cloud computing nodes, but spread throughout in the network resulting in a highly distributed platform. These platforms enable to improve user performance

* Corresponding authors.
E-mail address: laura.ricci@unipi.it (L. Ricci).

http://dx.doi.org/10.1016/j.jcss.2016.04.008
0022-0000/

and enhance security and privacy of users' data. For instance, next generation social networks can benefit from these new platforms, since increasing concerns about privacy are leading their architecture from the current centralized paradigm towards a more distributed one, where a user's information leaves its own computer on a need-only basis. Distributed platforms can be used to manage the connections between nodes to allow the user to interact efficiently with its own contacts, while still being guaranteed to reach every other node in the network.

Despite of the similarities with the peer-to-peer networks, not all the service discovery supports proposed in the last decade for these platforms can be directly ported to the new ones, since they do not meet all the distinguishing traits of the new computational paradigms.

Consider the problem of defining a distributed support for multi-dimensional range-queries. Several proposals have been presented in the last decade for peer-to-peer networks. Many of them are based on Distributed Hash Tables (DHTs) and exploit data delegation, i.e. the data published by a peer is generally stored on another peer chosen according to a mapping strategy guaranteeing efficient routing. Even if this solution offers several benefits, such as logarithmic bounds for routing and the possibility of employing uniform hashing to guarantee load balancing, delegation requires an high level of replication in a highly dynamic environment and may be not appropriate for platforms targeting data privacy. Furthermore, even if DHTs offer high performance in searching exact, single-attribute values, 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 [4–12]. The majority of these solutions has high maintenance costs, high replication costs, or need a high number of messages to solve range queries, especially when conditions over related attributes are not sufficiently selective. Other approaches to query resolution return results with probabilistic guarantees [13,14].

The problem of data delegation affects a further class of solutions based on the definition of peer-to-peer tree-based overlays [15–17,2]. As a matter of fact, these solutions partition the domain of values among the peers and delegate data to the node paired with the range including it. Furthermore, multi attributes queries are not always supported.

A widely-accepted class of solutions for placing and retrieving objects in highly distributed systems are those based on the definition of Voronoi Tessellations, these include VoroNet [18], SWAM-V [19], VoRaQue [20], GeoPeer [21]. In these solutions, each peer publishes and stores one or more objects characterized by a set of attributes, thus avoiding delegation. The attributes of the objects are exploited to pair the peer with a point of the multi-dimensional space. The attribute space is partitioned according to some notion of distance such that each peer is paired with all the points of the space that are closer to it with respect to any other peer. Since objects with similar attributes are close in the network, data locality is preserved and all the objects satisfying a multi-dimensional range-query can be localized within a region of the space. Moreover, locality constitutes an advantage for fault tolerance with respect to delegation-based approaches, since object insertion/removal causes changes only on the object neighbourhood. Finally, even if routing cannot be based on data mapping strategies, efficient routing algorithms can be defined by exploiting the mathematical properties of the Voronoi Tessellations [22], with additional long range links for fast searching to support query resolution.

The main drawback of these solutions is their poor scalability when the number of attributes and, correspondingly, of the dimensions of the space increases. While Voronoi-based approaches have been proposed on 2-dimensional spaces, higher dimensionality leads to an impractical runtime for the high average number of neighbours of a peer [18].

In order to take advantage of the Voronoi overlays' features and give a support for multi-attribute searches, 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 highly distributed systems. In the following we will use the term peer to refer a generic node of a highly distributed system. The proposed solution is based on a multilevel hierarchical structure that takes the form of a tree of Voronoi Tessellations; each level of the tree maps a different pair of attributes of the object, and each node of the tree is constituted by a 2-dimensional tessellation. In case a set of objects have the same coordinates in the multi-dimensional space, and thus a cluster of peers gets paired with the same point of the Voronoi space, a new tessellation comprising the cluster is created and inserted at a lower level of the hierarchy. Moreover, a clustering mechanism based on an absorption radius is exploited to gather peers that are close in the space. To have an upper limit on the number of neighbours, for each cluster a number (that depends on the cluster size) of peers are elected super-peers. They are the only peers that are visible (i.e. can interact) in the Voronoi Tessellation the cluster belongs to. They act as gateways between successive levels of the *Hivory* network. This mechanism allows to further limit the number of messages required to solve queries.

This paper improves the preliminary version of Hivory presented in [23] in several directions:

- We give a formal definition of Voronoi Tessellation Tree (VTT).
- We provide a formal proof of the soundness of the super-peer election mechanism.
- We give a detailed description of the Leave operation.
- We present a comprehensive evaluation of the complexity of all the operations defined by Hivory (Super-Peer election, join, leave and multi-range queries).
- We present a new set of experiments, including an experimentation on data taken from a real platform.

The remainder of this paper is organized as follows. Section 2 discusses some approaches proposed in the literature for supporting multi-attribute range queries. A preliminary discussion on the implementation of multi-attribute range queries on Voronoi networks is presented in Section 3. The architecture of *Hivory* is described in Section 4. Section 5 details Hivory's

operations, while Section 6 presents an analysis of the complexity of the operations. The experimental results are presented in Section 7. Finally, Section 8 reports some conclusions.

2. Related work

In the last decade a large set of supports for multidimensional range queries targeted to peer-to-peer systems have been presented, while only a few recent proposals target IoT environments.

A first class of proposals for peer-to-peer systems extends DHTs [24–26] to support multi-dimensional range queries. The first fundamental work in this direction is *MAAN* (Multi-Attribute Addressable Network) [4]. In MAAN objects are identified by a set of attribute-value pairs, and each attribute is mapped on a Chord ring [24] through an hash function which computes, for each resource, a target node, storing the full resource description. This way 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 most selective attribute, while the other attributes are checked using the replicated data.

In *SWORD* [8], nodes are divided in *reporting nodes*, sending periodic updates of the object values, and *DHT Server nodes* that form the DHT overlay and receive and store resource values and handle users queries. The possible values of each attribute are stored into contiguous regions of the DHT key space using a proper mapping function. A range query on an interval of values of an attribute is solved by querying all the nodes holding the keys in the range defined by the query.

Mercury [6] is a multi-level DHT network that uses a different DHT (called *Hub*) for each attribute. Each resource registers *all* the values of its attributes in each corresponding Hub 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. [27] extends the CAN DHT [25] in order to support multidimensional range queries. CAN (Content Addressable Network) is a DHT based on a n-dimensional Cartesian Coordinates Space, where the entire space is partitioned amongst all the peers. Each peer owns a zone in the overall space and manages the objects that are mapped to that zone by the hashing function, thus exploiting object delegation. In [27], a proper CAN DHT layer is used separately for each attribute. In each layer, each node handle a subinterval of the corresponding attribute domain. Each sub-query 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 drawbacks: 1) the delegation of the objects published by a peer to other peers, which can be an issue for privacy; 2) the use of one DHT per attribute, which leads to high maintenance costs; 3) the exploitation of the most selective attributes to limit the diffusion of queries, which requires high replication costs in terms of number of messages exchanged. The works in [5,6] and more recent proposals [7–11] have similar drawbacks.

Another class of approaches studied in the literature [18,28,23,19] directly exploit a multidimensional graph to support multi-dimensional query resolution. These approaches overcome the delegation issue by storing the objects published by a peer on the peer itself and by exploiting the attributes of the objects to position the peer in a multidimensional space. Most of these approaches exploit a Voronoi Tessellation of the multi-dimensional space. Even if other space partitioning strategies can be exploited, the Voronoi approach presents several advantages, like the possibility of defining efficient AOI-cast algorithms, based on Voronoi properties, which permit an efficient propagation of a range-query within the area defined by it. For this reason, this paper focuses the attention on this structure, even if the idea of defining a hierarchy of low dimensional spaces can be applied to other multidimensional space partitioning approaches.

The *Voronet* P2P network [18] maps each peer to a bidimensional attribute Voronoi space where the values of two properly selected attributes of the object published by the peer specify its coordinates in the space. A overlay link is defined between peers that are Voronoi neighbours. These links correspond to those of the Delaunay Triangulation paired with the Voronoi Tessellation. Complexity bounds are given, both for routing and for the routing tables size. The main drawback of this approach is that it can be applied to bidimensional Voronoi space only, and that it does not support range queries.

SWAM (Small World Access Methods) [19] is a family of distributed access methods to efficiently execute range and k-nearest neighbours queries. They guarantee that finding an object defined by an exact query is logarithmic in the size of the network, and that all the similar objects would be located in neighbouring nodes. A range query is considered as an exact-match query for a point chosen in the AoI of the query, then flooding is adopted to propagate the query to all the nodes of the AoI. The main drawback of this approach is the high cost introduced by flooding.

The protocol in [29] builds a distributed Delaunay triangulation based on the equiangular property [30]. Periodically, each node checks whether it respects this property and whether its neighbours do too, otherwise they create new triangles to maintain a correct structure. [20] optimizes the notification of a query within its AoI by introducing Compass Routing [22], which has the characteristic to always find a finite path between two nodes of a Delaunay Triangulation, although it is not cycle free for general graphs. [29] suggests to exploit Compass Routing to define a Spanning Tree supporting an application layer multicast. [31–33,21,28] propose protocols to build and maintain Delaunay triangulation-based overlay networks. [32] describes an incremental algorithm to construct and manage bi-dimensional Delaunay overlay networks where nodes communicate only with neighbouring nodes and build the overlay network incrementally. Nodes communicate with each over a virtual collaborative space that exploits the overlay network, and employ multi-hop communication between dis-

tant nodes. This approach is applicable to geographical networks with large diameter and GIS (Geographical Information Systems). [33] proposes a distributed algorithm to build spherical Delaunay networks where nodes operate independently to generate incrementally a local area network according to the geometrical proximity of neighbour nodes. It can be used in the context of Collaborative Virtual Space. In [21] it is proposed a location-based query support, Geopeer, whose nodes arrange themselves to form a Delaunay triangulation augmented with long range links. [28] defines a peer sampling layer based on the Cyclon and Vicinity gossip protocols [34] to build a Delaunay network. The information returned by this layer is passed to an upper layer, which exploits the mathematical properties of Voronoi Tessellation [30] to detect the neighbours of a node.

It is worth noticing that, despite several Voronoi-based approaches have been proposed, they focus on low dimensional Voronoi Tessellation and do not face the problem of the curse of dimensionality.

Finally, [2,35] present proposals specifically designed for IoT scenarios. [2] supports multi-attribute range-queries and adopts a peer-to-peer approach for guaranteeing scalability, robustness, and maintainability of the overall system. The Discovery Service linearises multi-attributes through space filling curves and exploits an indexing structure built on top of the Kademlia DHT overlay network. [35] provides a simplified programming abstraction for IoT defining an API including functions for querying distributed data.

3. Preliminary discussion

The Voronoi approaches mentioned in Section 2 are based on the assumption that each peer publishes an object characterized by k attributes, mapping it into a point of a k-dimensional space, which is called the *site* of the object. The position of peers in this space defines the Voronoi Tessellation. We briefly recall the notion of Voronoi Tessellation:

Definition 1. Given M sites in a n-dimensional space, a Voronoi Tessellation is the partitioning of the space into M convex areas such that each area contains exactly one site and every point in a given area is closer to the site paired with that area than to any other site.

If the borders of two areas of a Voronoi Tessellation overlap, the corresponding sites are called *Voronoi neighbours*. A *Delaunay triangulation* is the graph obtained by connecting neighbours sites.

The main problem for the application of Voronoi-based solutions to real scenarios is the *curse of dimensionality* [36]. In fact, while the number of Voronoi neighbours of a site has a probabilistic bound of $O(1)$ for bidimensional Voronoi Tessellations, it grows as $O(n^{\lceil \frac{d}{2} \rceil - 1})$ for a number of dimensions $d > 2$ [36].

A simple solution to this problem is to partition the set of attributes so that each subset is paired with a different Voronoi Tessellation, obtaining a set of disjoint tessellations for each object. In this section we show that this solution, which is based on the reduction of the space dimensionality, overcomes the curse of dimensionality problem, but is not suitable for realising efficient distributed supports. This discussion is intended to justify the novel hierarchical approach introduced in the next section.

Consider a dimension reduction where each Tessellation is defined by considering a subset of the attributes of an object.

Definition 2. Given a set O of objects characterized by the attributes $A = \{a_1, ..., a_k\}$ and $A' \subseteq A$, $|A'| = t$, $V_{A'}$ is a t-dimensional Voronoi Tessellation such that

- each dimension of $V_{A'}$ corresponds to an attribute $a_i \in A'$
- $V_{A'}$ includes a set S of sites such that the coordinates of any $s \in S$ correspond to the values of the attributes $\in A'$ of at least an object $o \in O$.

Definition 3. Given an object $o \in O$, $map(o, V_{A'})$ denotes the site $s \in V_{A'}$ whose coordinates are defined by the values of the attributes of o in A'.

Based on this, given a partition P of the set of attributes of the objects, each object is paired with a set of Voronoi Tessellation, $VorSet(P)$, containing one tessellation for each element of the partition P. Note that this implies that each peer should join a set of overlays, one for each tessellation.

Definition 4. Given a partition P of the set of attributes A,

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

Let us consider now how range queries may be supported on these overlays. Given the partition P of the set of attributes, a range query q can be resolved according to two alternative strategies. We denote with $AoI(q, V_A)$ the region of V_A defined by the constraints of q referring the attributes in A. Given a partition P of the attributes, the first solution visits in parallel

all the $AoI(q, V_p), \forall p \in P$. The matches for the query belong to the intersection of the sets of objects returned by each visit. As discussed in [4], the peer submitting the query may be overwhelmed by a huge amount of unrequired notifications, since $AoI(q, V_p)$ may contain objects that do not match the query due to attributes not in p. The second strategy restricts the search to the most selective AoI. Each peer in this AoI checks if its object matches the constraints on attributes not in p and, in this case, it returns its object as a solution of the query. This approach reduces the number of unrequired notifications, but the number of peers involved in the query resolution may be much higher than the number of its matches also in this solution, especially when the selectivity of the AoI is not high and/or when the distribution of the objects is skewed so that very densely populated AoIs exist.

Furthermore, these strategies do not take into account the problem of clustering. A cluster C is a set of objects characterized by the same attributes values, leading to being paired with the same Voronoi site.

Definition 5. Consider a partition P of the attributes, a Voronoi Tessellation V_p paired with the set of attributes $p \in P$ and a site $s \in V_p$. The cluster of objects paired with s is defined as follows:

$$C(s, V_p) = \{o \in O \text{ such that } map(o, V_p) = s\}$$

The main problem of clustering is that the probabilistic bound in the number of neighbours of an object is not $O(1)$ anymore, even for bi-dimensional Voronoi Tessellations. Since all the objects in a cluster are paired with the same site, it is not possible to exploit the Delaunay triangulation edges efficiently for the resolution of a range query. A solution where a peer n sends the query to all or a subset of the peers of the cluster is not feasible, because the huge amount of needed connections, or the necessity of flooding to propagate the query to the other peers, which introduce a large amount of redundant messages. Clustering occurs often in real applicative scenarios. For instance, a large amount of restaurants can be characterized by the same amount of seating capacity and average price. Note that the reduction of a n-dimensional Voronoi Tessellation to a set of lower-dimensional Voronoi Tessellation increases the level of clustering, because the probability to coming across objects characterized by the same attribute values is inversely proportional to the number of attributes exploited to map an object on a Voronoi site.

4. Hierarchical Voronoi Tessellations

Hivory exploits Voronoi Tessellations in a different way with respect to what discussed in Section 3, in fact, it exploits object clustering to define *a hierarchy of 2D Voronoi Tessellations*.

In the rest of the paper we will consider, for the sake of simplicity, a one-to-one mapping between objects and peers, and we will use the terms *object* and *peer* interchangeably. 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. The obtained results can be easily extended to the case where a peer publishes several objects, by pairing a Voronoi site to each object, and relating multiple sites to each peer.

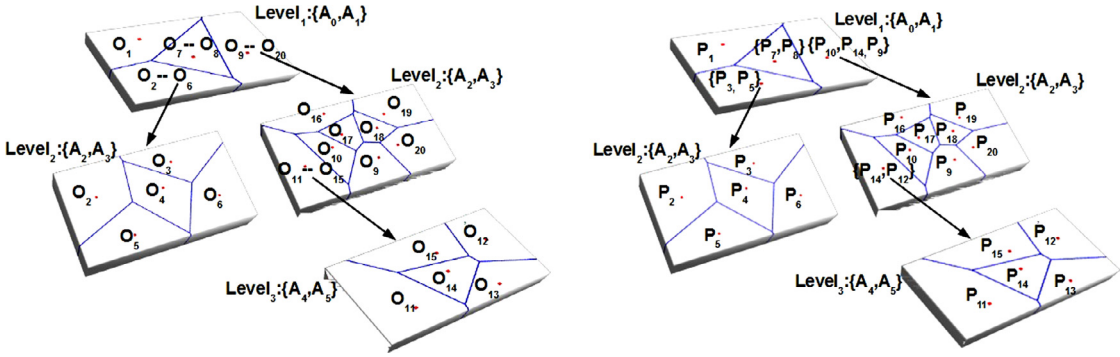
We first give an informal definition of the hierarchical structure we propose, afterwards we formalize it. Let us suppose, for the sake of simplicity, that each object is characterized by an even number k of attributes. Hivory defines a hierarchy of at most $l_{max} = \frac{k}{2}$ levels, each level corresponding to a pair of different attributes. Whenever a cluster C of objects is paired with a Voronoi site s at level $l \in 1, .., l_{max}$, and the size of the cluster exceeds a predefined threshold T, a new Voronoi Tessellation V is defined at level $l+1$, and all the objects of C are mapped to V by exploiting the attributes paired with the next level $l+1$. This implies that each cluster of level l exceeding the threshold, defines a different Voronoi Tessellation at level $l+1$ except at level l_{max}, because all the attributes have already been exploited to define the lower level Tessellations.

Example 1. Fig. 1a shows an example of Voronoi Tessellation hierarchy defined by Hivory. In the figure the notation O_i--O_k denotes the cluster including the objects O_j such that $i \leq j \leq k$. The threshold T is set equal to 3, i.e. a new Voronoi Tessellation is created if a cluster size is larger than 3. Therefore, the cluster O_7--O_8 is not expanded, while the clusters O_2--O_6 and O_9--O_{20} are expanded generating a further hierarchy level. Note that a subset of the objects of the cluster O_9--O_{20} at $Level_1$ is paired with the same site at $Level_2$ so that they define the cluster O_{11}--O_{15}, which is recursively expanded at $Level_3$. □

Let us now describe formally the hierarchical Voronoi Tessellation we propose. To reduce the overhead needed by a dynamic ordering of the attributes, we statically order them and define a static mapping between each pair of attributes and one level of the hierarchy.

Definition 6. Given a set of sites S, $V(S)$ is the Voronoi Tessellation defined by the sites $\in S$.

Definition 7. Let A be the set of $k = 2 * l_{max}$ attributes of a set O of objects. Let us consider an ordered sequence S of the attributes in A:



(a) A Voronoi Tessellations hierarchy

(b) Super Peer Election: same scenario of Figure 1a, showing SPs for each level of the hierarchy

Fig. 1. Hierarchical Voronoi Tessellations.

- Given a level $l \in [1, l_{max}]$, the attributes of O of level l, referred as A_l, are those whose rank in S is, respectively, $2*l-1$ and $2*l$.
- Given a level $l \in [1, l_{max}]$, the attributes of O of level less than l, referred as $A_{<l}$, are those whose rank in S is $< 2*l-1$.
- Given an object $o \in O$ and the attributes $A_{<l}$, $value_{<l}(o)$ is the sequence of the values of the first $2*l-2$ attributes of o.

Let us now formalize the notion of Voronoi Tessellation of level l labelled by a sequence of values val. It is worth noticing that, while all the objects in O are mapped onto the root Voronoi Tessellation by exploiting the attributes of level 1, A_1, only a subset of the objects are mapped onto the Tessellations of level $l > 1$. These objects are characterized by having the same values of the attributes $A_{<l}$, because they all belong to the same clusters of objects at the lower levels. We exploit the sequence of values of $A_{<l}$ to uniquely label a Tessellation of level $l > 1$.

Definition 8. Given a level $l \in [1, l_{max}]$ and a sequence val of $(2*l)-2$ attribute values, we define a Voronoi Tessellations $V_{l,val}$ of level l labelled by val as follows:

$$V_{l,val} = \begin{cases} V(\{s : \exists o \in O, s = map(o, A_1)\}) & \text{if } l = 1 \wedge val = \epsilon \\ V(\{s : \exists o, value(o, A_{<l}) = val \wedge s = map(o, A_l)\}) & \text{if } l \neq 1 \wedge val \neq \epsilon \end{cases}$$

Example 2. Consider the Voronoi Tessellation in Fig. 1a. Even if there are two Voronoi Tessellations at the level 2 of the hierarchy, both paired with the attributes $\{A_2, A_3\}$, they correspond to two different clusters at level 1 of the hierarchy. Each Tessellation at level 2 is uniquely identified by the coordinates of the corresponding cluster at level 1. □

We can now define formally a *Voronoi Tessellation Tree, VTT*.

Definition 9. Given an integer threshold T, a Voronoi Tessellation Tree (VTT) is a hierarchical structure defined recursively as follows. $V_{l,val}$ is a VTT if one of the following conditions holds:

- $(\forall s \in V_{l,val}, \mid C(s, V_{l,val}) \mid < T) \vee (\exists s$ such that $\mid C(s, V_{l,val}) \mid > T \wedge l = l_{max})$.
- If $\exists$ a set $S = \{s_1, \ldots s_k\}$ of sites of $V_{l,val}$ such that $\mid C(s_i, V_{l,val}) \mid > T$, the structure obtained by connecting each site $s_i \in S$ of coordinates x_i, y_i to V_{l+1,val^i}, where $val^i = val \cdot x_i \cdot y_i$ is a VTT, if V_{l+1,val^i} is a VTT.

Example 3. Consider a Voronoi Tessellation $V_{1,\epsilon}$, and its site s whose coordinates are $x = 8, y = 7$ and suppose that a set S of more than T objects is paired with s. Then a Voronoi Tessellation $V_{2,8\cdot7}$ is created, i.e. a tessellation 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. The sequence $val = 8 \cdot 7$ uniquely identifies the tessellation among those of the same hierarchy's level. □

The Voronoi tree is exploited to define a set of distributed overlays where the nodes correspond to the peers and the overlay connections to the Delaunay graph paired with each tessellation. Each peer joins all the overlays where the object it published is mapped. In the rest of this paper, we will mainly focus on the management of the hierarchy, while we refer to [28,18] for the techniques defined to manage a single overlay, and for the routing algorithms that exploit the Delaunay graph properties.

Assuming that each peer is visible at each level of the hierarchy where its object is mapped, the hierarchy may be exploited to optimize the query resolution process, refining it recursively by visiting in top down fashion the hierarchy

itself. Whenever a query is forwarded to a peer of a cluster, it does not broadcast the query to the other peers of the same cluster, but instead it switches the query to 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, which decreases step by step. Note that the cost of the creation of a new Voronoi Tessellation may be balanced by the reduction of the candidate matches by choosing a proper threshold T. Clustering is also exploited to distribute the load of query forwarding as well. When a query reaches the neighbour of a cluster, it chooses at random a peer of the cluster to forward the query to. This way, load is balanced because different queries are statistically distributed to different peers.

Despite its advantages, clustering may introduce a high number of connections between neighbour peers and reduce the advantages of considering 2D-dimensional Voronoi Tessellations. In fact, a single edge between two Voronoi neighbours may correspond to a set of links in the overlay, in the case the sites connected by that edge are paired with clusters of objects and, correspondingly to a set of peers. In the following section we will describe a *super-peer election* mechanism able to reduce the number of peers paired to the same site.

4.1. Super-peer election

Let us suppose that a cluster C occurs at a site s at level l, Hivory elects a subset m of the peers belonging to C, with $m = \lfloor log(|C|) \rfloor$, as the representative peers of the cluster C. The operation considers to maintain the correct number of these representative peers in the overlay at any time; still, the elected peers are currently chosen at random among the eligible peers. Only such peers belong to the overlay at level l, i.e. are *visible* at level l. This strategy guarantees a logarithmic bound on the number of neighbours of each peer [37]. Each peer elected as representative of a cluster, called *super-peer* (SP), acts as a gateway between different levels of the hierarchy, i.e. it is responsible of propagating the query from the level l to the level $l+1$ where it has been elected. Among the peers of the same cluster at level l we aim at electing $log(k)$ only as SPs, with the rationale that they will be the only ones visible at level $l+1$. Current approach considers exploiting a bottom up SP election algorithm to collect candidate super-peers, and then choose the super-peers at random. Future work will leverage heuristics to select SPs that are characterized by high computational power and bandwidth, to facilitate support of a higher traffic load since they are visible on more overlays. All the other peers of the same cluster are not visible at level l. The $log(k)$ SPs elected at level l are guaranteed to be visible at level $l+1$ exploiting a bottom up SP election algorithm.

Example 4. Fig. 1b shows the result of SPs election with reference to the Voronoi Tessellations hierarchy shown in Fig. 1a. Let us suppose peers P_3 and P_5 have been elected SPs of the cluster P_2--P_6 ($\lfloor log(5) \rfloor = 2$). Even if all the peers of the cluster P_2--P_6 are logically mapped at a $Level_1$'s site, only two of them are visible at such level. Note also that P_{14}, which is a SP of the cluster P_9--P_{20}, is visible at different hierarchy levels, since it has been recursively elected SP at those levels. □

The following theorem states that, starting from a level l with $l \neq 1$, it is always possible to elect a number of SPs equal to the logarithm of a cluster size, thus proving the soundness of the SP election mechanism.

Theorem 1. *Let us consider a Voronoi diagram $V_{l,val}$ and a site $s \in V_{l,val}$ such that $|C(s, V_{l,val})| > T$, where $T \geq 2$ is a fixed threshold. At least* $\log(|C(s, V_{l,val})|)$ *peers are visible from the Voronoi diagram of level $l+1$ rooted at s.*

Proof. By induction.
Base case: Consider a site $s \in V_{l_{max}-1,val}$ such that $|C(s, V_{l_{max}-1,val})| > T$. Since the underlying Voronoi Diagram at level l_{max} includes all the peers in $C(s, V_{l_{max}-1,val})$, their number is greater than $log(|C(s, Voronoi_{l,val})|)$.

Induction hypothesis: Given a $V_{l,val}$ rooted at a site s of a Voronoi diagram V' of level l, with $|C(s, V')| = N > T$, for each site $t \in V_{l,val}$ such that $|C(t, V_{l,val})| = k > T$, at least $\log(k)$ peers are visible from the Voronoi diagram at level $l+1$ rooted in t.

Inductive step: Suppose that $V_{l,val}$ includes a set S of sites, with a total number N of associated peers. Moreover, for each site $t \in S$, let $|\ C(t, V_{l,val})\ | = N_t$. Let us consider the sets $A = \{t \in S : N_t > T\}$ and $B = \{t' \in S : N_{t'} \leq T\}$. Using this partition, we can express the number of peers paired with the sites of $V_{l,val}$ as:

$$N = \sum_{t \in A} N_t + \sum_{t' \in B} N_{t'} \tag{1}$$

However, only a subset of these peers are visible in $V_{l,val}$. By the inductive hypothesis, for each $t \in A$, $\log(N_t)$ peers are visible at level l from the Voronoi diagram at level $l+1$ rooted in t. On the other hand, all the peers paired with each site in B are visible in $V_{l,val}$. Then, the total number of peers that is visible in $V_{l,val}$ is:

$$\sum_{t \in A} \log(N_t) + \sum_{t' \in B} N_{t'}$$

We have to prove that this value is greater than or equal $\log(N)$, that is:

$$\log(N) \leq \sum_{t \in A} \log(N_t) + \sum_{t' \in B} N_{t'}$$

We can re-write the previous expression as:

$$\log(N) \leq \sum_{t \in A} \log(N_t) + \sum_{t' \in B} \log(2^{N_{t'}})$$

and, by applying the properties of logarithms, we obtain:

$$\log(N) \leq \log(\prod_{t \in A} N_t \times \prod_{t' \in B} 2^{N_{t'}}) \tag{2}$$

Using Formula (1), Formula (2) becomes:

$$\log(\sum_{t \in A} N_t + \sum_{t' \in B} N_{t'}) \leq \log(\prod_{t \in A} N_t \times \prod_{t' \in B} 2^{N_{t'}}) \tag{3}$$

Inequality (3) is always true, thus demonstrating the theorem. This fact follows by noting that:

$$\sum_{t \in A} N_t \leq \prod_{t \in A} N_t \; since \; \mid N_t \mid > 2, \forall N_t \in A$$

and

$$\sum_{t' \in B} N_{t'} \leq \prod_{t' \in B} 2^{N_{t'}}$$

Hence, the right-side of Inequality (3) is always greater or equal to the left side of the expression. As a consequence, more than (or at least) $\log N$ peers are visible at $V_{l,val}$ at any level $l > 1$, so that the required number of SPs can be elected to be visible at level $l - 1$. □

Finally, Hivory adopts a strategy to increase clustering in regions that include a huge amount of close sites, to decrease the query resolution cost. 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 – called the *absorption radius* – the mapping of the object is forced so that o is paired with s.

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 visible peers in the crowded region decreases.

5. Hivory: the operations

This section describes the *join*, *leave* and *super-peer* election operations, and the algorithm adopted to resolve *multi-attribute range queries*. To simplify the description, we consider these operations as executed sequentially.

First of all, a peer p maintains a set of data structures for each Voronoi tessellation of every level where p is visible, which store p's Voronoi neighbours at the corresponding level. For each level l where p is visible and a cluster occurs, p stores a reference to all the other peers of the cluster. Finally, p maintains a reference to the SPs within the upper level $l - 1$, if any.

5.1. The Join operation

The insertion of a peer p always starts from the root level $l = 0$ and goes down the hierarchy until it reaches the insertion level l, i.e. a level where either:

- the attributes of p place it far away from any other site in l, generating a new site, or
- a cluster C absorbs p at level l, and C size is smaller than the threshold T or $l = l_{max}$, so there is no corresponding Voronoi tessellation at level $l + 1$.

Note that, while p results visible at level l, it is visible on upper levels only if it is elected SP also at these levels.

The insertion procedure is executed recursively for each level $m \leq l$, where l is the insertion level of p. For each of such level, p joins an existing cluster, while it joins the corresponding overlay only if it is elected SP at that level. The SP election procedure, when required, is executed bottom–up after the joining peer has completed the insertion procedure; it is described in Section 4.1, while in this section we describe the insertion procedure at a generic level l.

Let us consider a peer p logically joining $V_{l,val}$ where l is a generic level and val is the sequence of values of the first $(2 * l) - 2$ attributes of p when $l \neq 1$, and the empty sequence ϵ when $l = 1$. First of all, a peer p detects the site s of $V_{l,val}$ closest to the values of its attributes at level l. Afterwards, p issues a *join message*, targeted to one of the peers paired with s

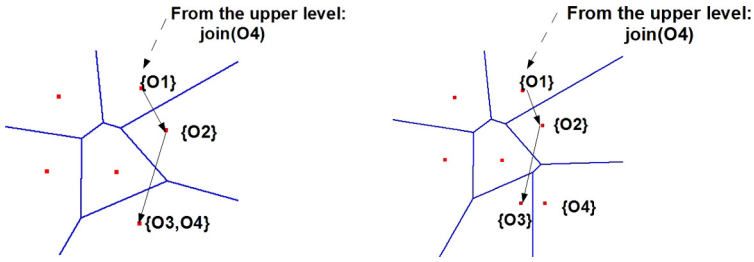


Fig. 2. The Join operation.

called m_l, and propagates it into $V_{l,val}$; m_l is the manager of the insertion of p at level l. The join message is propagated to the other levels by a bootstrap peer at the root level, i.e. retrieved by an off-line procedure, or by a SP acting as a gateway between the levels $l-1$ and l.

When m_l receives the join message, it checks whether the coordinates of p are within the *absorption radius* of s, otherwise it must update the Voronoi tessellation at level l as in [18], so we will not discuss this case here.

Recalling Definition 5, if the coordinates of the insertion point are within the absorption radius of s and the threshold value is T, the following cases may occur:

- $|C(s, V_{l,val})| < T$. In this case, p joins the existing cluster, avoiding the generation of a new level.
- $|C(s, V_{l,val})| = T$. A new Voronoi Tessellation at level $l+1$ is created where all the peers belonging to $C(s, V_{l,val})$ are mapped.
- $|C(s, V_{l,val})| > T$. The join request of p is propagated on the Voronoi Tessellation already existing at level $l+1$.

In the first case, m_l notifies the identity of p to all the cluster's peers and the neighbours, and vice versa, so that it may define its local view of the Voronoi overlay.

In the second case, m_l acts as a bootstrap peer to build the new Voronoi Tessellation V at level $l+1$. After creating V, it propagates within V the join message tagged by the new level $l+1$ for each peer $p \in C(s, V_{l,val})$ until it reaches the insertion point of p in the new level.

Finally, in the third case, a Voronoi Tessellation already exists at level $l+1$, so the join message is switched by m_l to this level and it is recursively propagated until the insertion point of p in $l+1$ is reached. At the end of the procedure, p has logically joined a set of clusters whose size was $\geq T$, one for each level between 1 and its insertion level. The identity of the super-peer from level l to level $l+1$, called gw_l, is stored in the join message sent to the level $l+1$, such that a further super-peer gw_{l+1} may store the identity of gw_l before eventually propagating the join message to a level $l+2$. In this way, a *distributed chain of backward links* is defined to implement the bottom up SP election. When the insertion of p is completed, a notification message is sent back over this chain so that each super-peer is able to verify the logarithmic bound on the number of peers of its cluster, which may be violated due to the insertion of the new peer, triggering a new SP election.

Example 5. Fig. 2 shows an example of the Join operation of a peer O_4 in an existing overlay, starting by its insertion level. O_4 issues the join message $join(O_4)$, which is propagated to O_1 from the upper level. Let us suppose that O_4 is mapped to the Voronoi region of O_3, at this level, so the join message is propagated to O_3. The figure shows the cases where the position of O_4 is within the absorption radius of O_3 – the left part of the figure – or not – the right part of the figure; in the latter case, a new site is paired with O_4 and the Voronoi tessellation is modified accordingly. □

5.2. The Leave operation

In Hivory each peer is responsible only for the objects it has published, so its leaving does not affect objects published by other nodes, but it may require a rearrangement of more than one level of the Hivory hierarchy. In fact, due to the departure of a peer p, the condition $m = \lfloor log(|C|) \rfloor$, where m is the number of SPs paired with a cluster C including p, may be violated.

The algorithm implementing the leave of a peer p includes the following phases:

- removal of p from the Voronoi Tessellation corresponding to its insertion level;
- removal of p from every level where it is visible;
- starting from the insertion level, for each cluster C including p, the condition $m = \lfloor log(|C|) \rfloor$ is checked, involving the revoke/election of further SPs.

The first phase involves the update of the Voronoi Tessellation at the insertion level of p, and, if p belongs to a cluster at that level, the notification of the leave of p to the other peers of the cluster. A distributed algorithm to update a Voronoi network due to a peer leave is described in [37] and we will not discuss it here.

The second and third phases are executed bottom up, starting from the insertion level l of p. If p is not a SP, it contacts one of the elected SP of l. For this reason, each SP stores and keeps updated the list of the other SPs for each level where it is visible, while the other peers exploit a lazy policy whose main goal is to minimize both the number of messages exchanged and the probability that all the SPs in the list are not up to date. The complete description of the algorithm is reported in [37]. Here we only outline some scenarios that may occur after removing p:

- $m = \lfloor log(|C|) \rfloor$. This may occur, for instance, if p is a SP and $log(|C|) = \lfloor log(|C|) \rfloor$.
- $m > \lfloor log(|C|) \rfloor$. In this case, our algorithm revokes one of the SPs paired with the cluster, repeating it at every level where it is visible.
- $m < \lfloor log(|C|) \rfloor$. This may occur because the leaving peer is a SP or because a SP has been revoked from an upper level. In this case a new SP must be elected.

Example 6. As examples of the three different scenarios above, we refer to Fig. 1b, in particular to the Tessellation labelled $V_{2,s}$, where s is the site at Level_1 that is the root of the Tessellation at Level_2 on the left. The first scenario happens if, for instance, P_6 leaves the network, since, in this case, $m = 2 = \lfloor log(|C|) \rfloor$, where $|C| = 4$. The second scenario happens if both P_2 and P_4 leave the network, since $m > \lfloor log(|C|) \rfloor$, where $|C| = 3$, $m = 2$ and $\lfloor log(|3|) \rfloor = 1$. One of the SPs should be revoked. The third scenario happens if P_5, visible on $V_{1,\epsilon}$ as SP of $V_{2,s}$, leaves the network since $m < \lfloor log(|C|) \rfloor$, where $|C| = 4$, $m = 1$ and $\lfloor log(|4|) \rfloor = 2$. Thus, a new SP has to be elected on Level_2 to be visible on Level_1. □

5.3. The super-peer election operation

The election of a new super-peer takes place when the number of SPs paired with a cluster, at any level, is not larger or equal to the logarithm of its size any more. Theorem 1 guarantees the existence of a new eligible peer, provided that any cluster of the underlying levels respects the logarithmic bound. For this reason, the election must be executed bottom–up starting from the insertion level of the joining/leaving peer. Current approach is that we collect eligible super-peers, and then we choose randomly the super-peers, with the constraint of maintaining the correct number of super-peers at any time.

In case a new peer joins in the overlay, the election is started by the super-peer that has relayed the join message to the insertion level of the joining peer, then an insertion message is propagated backward to the super-peers at the upper levels. In case of the leave of a peer, the election is started by the leaving peer itself.

The algorithm executed to elect a SP is structured according to two phases: 1) Selection of the peer to be elected, 2) Election of the selected peer.

Peer selection. A new SP to be elected into a cluster C paired with the site s of the level l may be any peer of the Voronoi tessellation V of the level $l+1$ rooted at s that was not a SP. The algorithm we propose tries to reduce the complexity of the peer selection phase. First, if a new peer p has become visible in V because the level $l+1$ is its insertion level or it has been elected SP from the underlying levels, p is selected to become SP at s, avoiding a search for a new SP.

Another scenario occurs when a new Voronoi tessellation is created at level $l+1$ because the size of a cluster of the level l has reached the threshold value T. To reduce the cost of the bottom up election, Hivory elects $T-1$ peers belonging to the cluster before the insertion of the new peer, as SPs of the Voronoi tessellation just created. Note that, in this case, the underlying level includes exactly T peers and $T \approx log(T)$ whenever T is small.

When no one of previous conditions occurs, a distributed algorithm is executed, which is based on Theorem 1 that guarantees the existence of an eligible peer at the level $l+1$. It is easy to verify that at least one of the peers that are eligible as SPs is a Voronoi neighbour of one of the current SPs. Let us suppose, by contradiction, that no one of the eligible peers is a Voronoi neighbour of a SP, then the Voronoi neighbours of any SP are all SP themselves. This implies that the Voronoi tessellation at the level $l+1$ is partitioned into two sets including, respectively, the peers elected SPs at level l and those that are eligible. This implies a contradiction, since any Voronoi tessellation is connected.

The super-peer performs the following steps:

- Verifies if among all its Voronoi neighbours there is an eligible peer, which would then get elected SP.
- If all the neighbours are already SPs, a broadcast message is sent to them to inquire whether among their Voronoi neighbours there is an eligible peer. Then, its identifier is sent to the super-peer. The peer included in the first received message will be elected SP, and the subsequent response messages are ignored.

Peer election. The super-peer adds the selected peer p in its list of SPs, and sends a notification message to the other SPs, such that they can add it to their list. Then, it notifies p to its Voronoi neighbours, so that they can update the list of their Voronoi neighbours too. Finally, the super-peer sends to p an election message containing all the information required for its insertion in the SP list and in the Voronoi tessellation of the level $l-1$. The super-peer then propagates the notification election message to the super-peer of the upper level.

5.4. Multidimensional range query resolution

A range query may be submitted by any peer p belonging to the overlay hierarchy. The first step of a query resolution is its propagation to a peer belonging to the first level, if p is not visible in it. Since each peer stores a reference to its SPs, these references may be exploited for a bottom up forwarding of the query up to the level 1. When the query has reached any peer at this level, a peer belonging to the AoI defined by the query at level 1 is reached through a *greedy routing*, then *compass routing* [29], an efficient routing algorithm for Delaunay overlays, is exploited to propagate the query to any peer located within the AoI.

When a peer belonging to a cluster whose size is smaller than the threshold T, receives a 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 and propagates it to the other peers of the cluster. On the other hand, if the peer belongs to a cluster whose size is larger than the threshold T, it makes its own local check and then it switches the query to the lower level, without propagating it to the other peers of the cluster; this way, peers will receive the query at any lower level of the hierarchy only if they belong to the AoI defined by the constraints on the attributes of that level.

6. The operations complexity

In this section, we show an analysis of the theoretical complexity of the operations described above, aimed at computing complexity upper bounds for two critical statistic distributions of the peers. These distributions have to be considered as critical, since each of them produces a challenging scenario for one of the operations described in this section. The complexity is computed as a function of the number of exchanged messages with respect to the total number of peers in the Voronoi network. The validity of this complexity function is due to the fact that the computation time on each peer is negligible with respect to the data transmission time and the data traffic on the network. The chosen critical statistic distributions cited above are the following, which lead to two different but opposite scenarios:

- an uniform distribution over the attribute values, implying a uniform distribution of nodes among the clusters at all levels;
- an unbalanced distribution – a power-law one – where a Voronoi diagram x at a level l is characterized by a number of cells N_x that is almost equal to the number of peers N in the whole overlay network.

6.1. The Join operation

To compute an upper bound of the number of messages exchanged for the Join operation, we suppose that a new peer p must be inserted in a Voronoi diagram at the last level, denoted with l_{max}. We must recall that, since the number of neighbouring sites is statistically constant in a 2D tessellation [18], the complexity of the creation of a new Voronoi region or the insertion in a cluster with size smaller than the threshold T at l_{max} is $O(1)$ respect to N, which is the number of nodes in the overlay network. At every level $l < l_{max}$, these operations take $O(\log N)$ due to the logarithmic bound on the number of SPs visible at l in the neighbouring sites.

For the above reasons, the more complex operation is to find the proper insertion point for p. At every level $l < l_{max}$, a *greedy routing* is required to find the site on l that is associated with p. Concerning the uniform distribution case, if in the first level there are N_1 clusters (corresponding to N_1 sites), the subtree rooted at each cluster contains N/N_1 peers. The same situation is replicated at every level until l_{max}, excluded. We have that $N = \prod_{l=1}^{l_{max}} N_l$. Given that a routing on a 2D Voronoi plane requires $O(\log^2 N)$ messages [18], the cost of finding the right cells across all the levels is:

$$\log^2 N_1 + \log^2 N_2 + \ldots + \log^2 N_{l_{max}} \tag{4}$$

With the given assumptions, we have that:

$$\log^2 N = \left(\log\left(\prod_{l=1}^{l_{max}} N_l\right)\right)^2 = \left(\log N_1 + \ldots + \log N_{l_{max}}\right)^2 \geq$$
$$\geq \log^2 N_1 + \log^2 N_2 + \ldots + \log^2 N_{i_{max}} \tag{5}$$

Thus, in this situation the complexity is potentially lower than in the 2D case.

Concerning the unbalanced case, where there is only one network x at a level l that has a number of cells $N_x \approx N$, and where $N_i \ll N, \forall l \neq x$, the complexity is:

$$\log^2 N_1 + \ldots + \log^2 N_x + \ldots + \log^2 N_{l_{max}} \to \log^2 N_x \approx \log^2 N \tag{6}$$

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

6.2. The super-peer election operation

Since the election of a peer is not executed any time a peer is inserted/removed, we do an amortized of the complexity of this operation. Currently, we consider that the super-peer is chosen at random between the eligible peers; in future work we will use high-performance/lightweight heuristics to select the best peer as super-peer according to given criteria, without impacting the performance of the system. As shown in section 5.3, searching for an eligible peer requires the SP managing the election to send at most a message to its Voronoi neighbours, except if the new inserted peer becomes visible as a new SP of the underlying levels. Then, it sends to the other SPs in the cluster and to the Voronoi neighbours of the tessellation at level $l-1$ a message to notify the election of the new SP so that they can update their lists of SPs and Voronoi neighbours, respectively.

Concerning the balanced case, all the three phases have the same cost. Supposing to insert a peer at level l, all the Voronoi Tessellations at level $l+1$ have size equal to $\frac{N}{\prod_{k=0}^{l} N_k}$. The number of peers belonging to the clusters at level l is equal to $\log\left(\frac{N}{\prod_{k=0}^{l} N_k}\right)$, since the peers in such cluster were previous elected SPs. Since each site on this tessellation has a statistically constant number of neighbouring sites, then the total election cost is $\log\left(\frac{N}{\prod_{k=0}^{l} N_k}\right)$. The number of elections carried out by a network at level $l+1$ is equal to the current number of SPs of the network, i.e. $\log\left(\frac{N}{\prod_{k=0}^{l} N_k}\right)$. Then, the total cost to perform all of the elections in a network at level l is $log^2\left(\frac{N}{\prod_{k=0}^{l} N_k}\right)$. Since the number of tessellations at level $l+1$ is equal to the number of sites in the tessellation at level l, the cost due to the election made by the networks at level $l+1$ is equal to $N_k * log^2\left(\frac{N}{\prod_{k=0}^{l} N_k}\right)$.

Generalizing the formula for any tessellation, we obtain that the total cost of all elections at each level is:

$$N_1 * \log^2 \frac{N}{N_1} + N_2 * \log^2 \frac{N}{N_0 N_1} + \ldots$$

$$\ldots + N_{l_{max}} * \log^2 \frac{N}{\prod_{k=1}^{l_{max}} N_k} \leq \log^2 \left(N * \sum_{k=1}^{l_{max}} N_k \right) \tag{7}$$

and the amortized cost is equal to $\sum_{k=1}^{l_{max}} N_k * \frac{log^2 N}{N}$.

Concerning the unbalanced case, there is only one tessellation at level l_{max} that has a number of cells $N_x \approx N$, so in each level $l < l_{max}$ there is only one cluster containing $logN_x$ SPs. Therefore, the total number of elections is equal to $l_{max} * \log N_x$. Since all the tessellations at all level have approximatively only a site, the costs of the other phases except the notification to the other SPs of the same cluster are negligible. Therefore, the amortized cost of such operation is equal to $l_{max} * \frac{\log N}{N}$, since $N_x \approx N$.

In conclusion, the amortized cost of the elections is in all cases less than $O(\log N)$, thus negligible compared to the complexity of the other operations.

6.3. The Leave operation

To perform an analysis of the complexity of the Leave operation in terms of upper bounds for the exchanged messages, we suppose that the peer p to be deleted is SP at all levels of the hierarchy. Since, for this operation, there are no routing algorithms involved, the complexity is proportional to the deletion of p in the tessellation at level l_{max}, its revocation as SP at all level and the election of new SPs, if needed. We can note that the deletion of p at level l_{max} costs $O(1)$, in every case, since the number of neighbouring sites is statistically constant.

The operations needed to perform the revocation of p at a generic level l are similar to the election ones, except for the absence of the peer selection phase, so its cost is $O(logN-1)$ at every level $l \leq l_{max}$. Concerning the unbalanced case, where there is only a tessellation x at a level k that has a number of cells $N_x \approx N$, we can note that an election never occurs since the number of SPs remains valid also after the deletion of p and corresponds to $logN-1$.

Concerning the balanced case, we have at the level l_{max} a number of peers equal to $\frac{N}{\prod_{i=1}^{l_{max}} N_i}$, and, consequently, a number of SPs equal to $log\frac{N}{\prod_{i=1}^{l_{max}} N_i}$. So, revocation of p as SP at level l_{max} costs $O\left(log\frac{N}{\prod_{i=1}^{l_{max}} N_i}\right)$, which is the same complexity of the election of a new peer, if needed. At every level $l < l_{max}$, this operation takes $O(\log N)$ due to the logarithmic bound on the number of SPs visible at l in the neighbouring sites. So, the Leave operation, in both cases, has complexity equal to $O(l_{max} * (logN-1)) = O(logN)$.

6.4. Multidimensional range query resolution

To compute an upper bound for the complexity of the resolution of multi-attribute range queries, we estimate the total number of nodes that have to be contacted when performing a query Q, i.e. all the nodes contained in all AoIs defined by Q. Let us then consider the number of nodes contacted by the *compass routing* algorithm.

Concerning the balanced case, we have that $N = \prod_{i=1}^{l_{max}} N_i$. If a tessellation at a level l has a combined selectivity $s \in (0, 1]$ over its attributes, $s * N_l$ nodes will be involved by the query. The total number of requested nodes is

$$\sum_{i=1}^{l_{max}} \prod_{j=1}^{i} s_j N_j \tag{8}$$

where we consider that $\forall . j s_j N_j \geq 1$. In fact, even a small s_j represents a fraction of the area of a Voronoi tessellation at level j in the hierarchy. Thus, this fraction is comprised in at least a Voronoi cell. Hence, at least one node will be in charge of handling the query at that level. Moreover, note that each N_i can be written as $N_i = \frac{N}{\prod_{j=1, j\neq i}^{l_{max}} N_j}$. Thus, we can express formula (8) as:

$$N * \left(\prod_{i=1}^{l_{max}} s_i + \sum_{i=1}^{l_{max}-1} \frac{\prod_{j=1}^{i} s_j}{\prod_{k=i+1}^{l_{max}} N_k} \right) \tag{9}$$

Please note that, $\forall i \in [1, l_{max} - 1]$, we have that

$$\frac{\prod_{j=1}^{i} s_j}{\prod_{k=i+1}^{l_{max}} N_k} \leq \prod_{j=1}^{l_{max}} s_j \tag{10}$$

In fact, we can re-write the inequality above as

$$\prod_{j=1}^{i} s_j \leq \prod_{j=1}^{l_{max}} s_j \prod_{k=i+1}^{l_{max}} N_k$$

Thus

$$\prod_{j=1}^{i} s_j \leq \left(\prod_{j=1}^{i} s_j \prod_{k=i+1}^{l_{max}} s_k \right) \prod_{k=i+1}^{l_{max}} N_k$$

With simple algebraic manipulations, we obtain

$$\prod_{j=1}^{i} s_j \leq \prod_{j=1}^{i} s_j \prod_{k=i+1}^{l_{max}} s_k N_k$$

that, with the given assumptions ($s_k N_k \geq 1$, $\forall k$), is always true. Thus, using inequality (10), we can write that

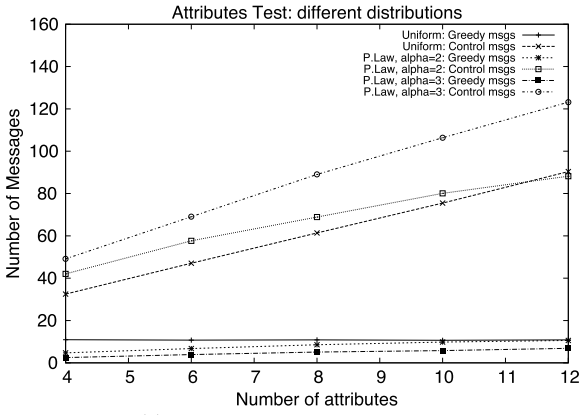
$$N * \left(\prod_{i=1}^{l_{max}} s_i + \sum_{i=1}^{l_{max}-1} \frac{\prod_{j=1}^{i} s_j}{\prod_{k=i+1}^{l_{max}} N_k} \right) \leq N * l_{max} * \prod_{i=1}^{l_{max}} s_i = l_{max} * s_1 \ldots s_{l_{max}} * N$$

So, the *upper bound* for the complexity of the multidimensional range query resolution process is proportional to the number of levels and the *combined selectivity* of all the attributes.

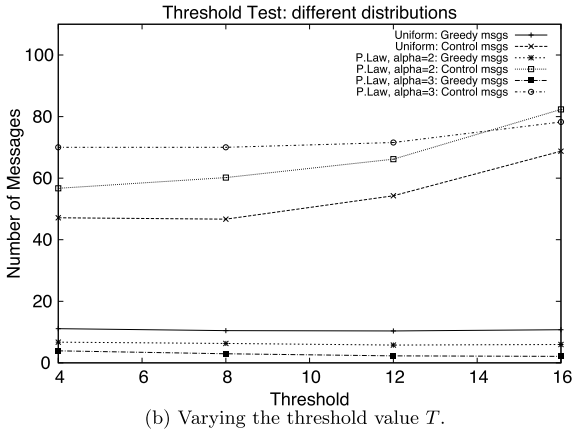
Concerning the unbalanced case, where there is only one tessellation x at a level k that has a number of cells $N_x \approx N$, if Q is directed to x, the selectivity expressed at the previous levels do not have a great impact on the final global selectivity, since the most populated tessellation is 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, being it the same of x.

7. Experimental results

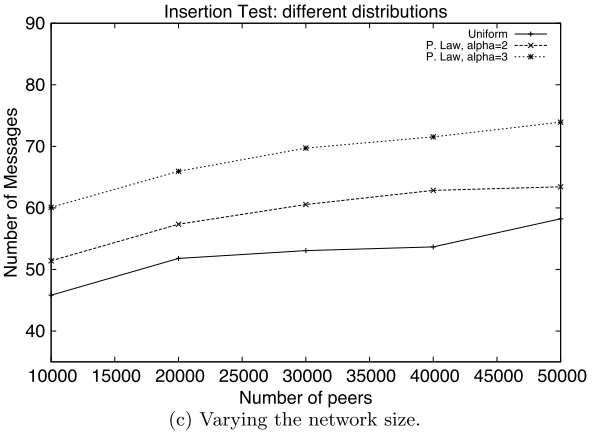
In this section we provide experimental findings regarding the proposed system. The evaluation was conducted by simulation using PeerSim [38] with three different data distributions. We used the average number of messages exchanged as a metric of performance, to validate the theoretical complexity analysis described in section 6. A further test was conducted by using a dataset with real-life measurements taken on PlanetLab [39]. Finally, we conducted a set of tests to compare Hivory with the MAAN, a multi-attribute range query support briefly presented in Section 2.



(a) Varying the number of attributes.



(b) Varying the threshold value T.



(c) Varying the network size.

Fig. 3. Join operation costs with three different data distributions.

Messages are divided in Greedy, Control, and Compass. The Greedy ones are sent during the execution of greedy routing, the Control ones are sent during insertion/removal of a node in/from a specific site, and the Compass ones are sent during the execution of compass routing. The system was tested under different conditions. We present the results related with the multi-attribute range queries resolution, and the maintenance cost (i.e. the cost to run the Join and Leave operations) of the network. In all the experiments we used the following assumptions and settings:

- Each peer is associated to an integer type multi-attribute vector. Every position of the vector represents the value, after algebraic transformations if needed, of the peer's owned resources.
- The size of all the two-dimensional networks that make up the various overlay network levels was set, for simplicity, to a fixed number of $1,000 \cdot 800 = 800,000$ points. This means that the odd and even attributes have domains $[0; 1,000]$ and $[0; 800]$, respectively.
- The network size varies from $10,000$ to $50,000$ nodes, with an absorption radius equal to 1.
- In order to obtain stable values, all simulations were repeated 100 times with different random peer attribute values, except those conducted on PlanetLab.

Fig. 3 shows the performance results obtained in the tests conducted by using three different data distributions: a uniform and two power-law distributions, one with $\alpha = 2$ and one with $\alpha = 3$. In the power-law distributions, values (and peers) are more concentrated at one end of each attribute domain than to the other, thus creating an unbalanced tree-shaped network.

Figs. 3a and 3b show the average number of greedy and control messages required to execute a Join operation in an overlay network fixed to $50,000$ nodes varying respectively the number of attributes and the threshold T. In Fig. 3a we fixed a threshold value $T = 4$, while in Fig. 3b we fixed a number of attributes equal to 6, which leads to an overlay network with $l_{max} = 3$. It can be seen that, for all the data distributions, changing the number of attributes and the value of T does not affect the number of exchanged greedy messages, while the number of control messages grows almost linearly. This is due to the fact that the number of exchanged greedy messages is a function of the number of sites in each tessellation while, on the contrary, the number of control messages rises with the number of attributes and the threshold value. In fact, control messages are sent only to Voronoi neighbours of the level in which a peer p is inserted. For example, adopting a

threshold value $T = 16$, a site s can have up to 16 peers visible as SPs before a lower level Voronoi tessellation rooted in s is built or, for a lower threshold value, to have 16 SPs, the number of peers in s must be at least 2^{16}. Therefore, if the number of peers in the network is fixed, the number of exchanged control messages is a linear function of the threshold.

The rest of the experiments with simulated data have been performed with a fixed number of attributes, equal to 6 and a threshold value $T = 4$. These parameters allow us to keep low the complexity of the tests, and therefore a lower test execution time. Fig. 3c shows the average number of messages required to execute a Join operation where the values shown in the graph were obtained averaging the sum of the Greedy and Control messages needed to manage a peer insertion. It can be seen that the number of messages scales well with the number of nodes, requiring less than 80 messages in the case featuring the largest network size. The best performance is obtained with the uniform distribution. It happens because nodes are also uniformly distributed in each Voronoi level they belong to. Thus, they are more likely to have (i) less neighbours and (ii) less nodes falling in the same Voronoi cell. Hence, less communications are required to insert a node, while more messages are needed when dealing with more concentrated distributions, like the two power-law ones.

For example, in the case of $N = 1,000$ and the uniform distribution, according to the theoretical complexity, we should have $log 1,000 = 10$ messages exchanged, while we obtained approximately 3.6, following far below the theoretical limit of complexity. The same phenomenon is repeated for all other values of N thus showing that the algorithm adopted for the Join operation achieves a good level of scalability.

The tests to evaluate the system performance in solving multi-attribute range queries were conducted by using the three node distributions and three different query selectivities, in an overlay network fixed to 50,000 nodes. Query selectivity is defined as a percentage of the area of a Voronoi tessellation defined by the restraint on each pair of attributes. Hence, a 0.05 query selectivity over a pair of attributes means that the query covers an area that corresponds to 5% of the whole Voronoi tessellation formed by the attributes related to that plane. The query has the same selectivity on both the attributes, thus each attribute has approximately 22.361% ($\sqrt{0.05}$) of selectivity in its domain. The lowest selectivity (highest percentage) ranges in [0.22361; 1], while the highest selectivity ranges in [0.05; 0.22361]. The tests were conducted by executing, for each network size, 100 random generated queries with the three different attributes selectivities reported above. Each query contains all the attributes, and the selectivity on each pair of attributes is determined by randomly choosing the selectivity of one attribute of the couple and by adjusting the second one in accordance with the global selectivity constraint. The queries were solved sequentially, after entering all the nodes in the network and are injected from nodes chosen at random on any level of the network. Hivory's experimental results are compared with the same tests performed on MAAN [4], a well-known P2P system for handling multi-attribute range queries. In Fig. 4 the number of messages was obtained averaging the sum of the Greedy and Compass messages needed to solve a query. The results are presented in a log scale along the y axis in order to allow the perception of the differences in the behaviour for different query selectivity that were otherwise hard to see.

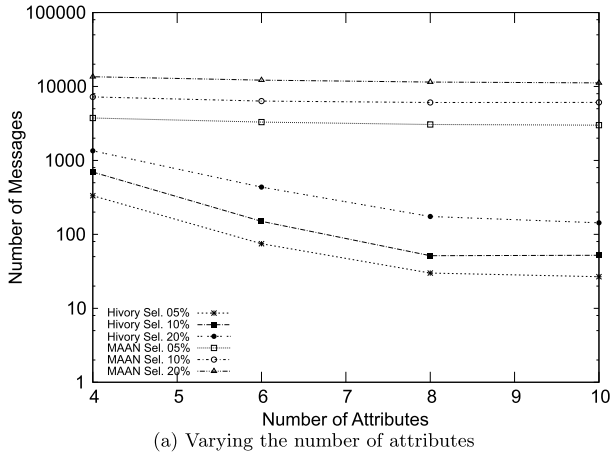
Fig. 4a shows the number of messages required for solving queries with the given selectivity, varying the number of the attributes in a query. MAAN uses a DHT with the highest selectivity (i.e. smallest area) for each query. Thus, its results are almost independent from the number of attributes. On the contrary, as explained in section 4, our system combines the selectivity of all the involved attributes, thus obtaining more discriminating power. As a consequence, when more attributes are involved, less messages are required. The results shows that our system outperforms MAAN from one to two orders of magnitude.

Next, we studied the system behaviour when the network size changes. The results, using the different data distributions with both Hivory and MAAN, are shown in Figs. 4b, 4c and 4d.

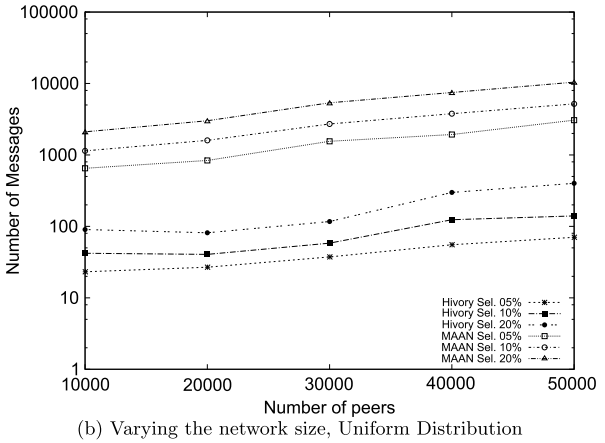
It can be seen that the system performance scales well with the number of peers, maintaining the number of messages under control with all the considered selectivities. The best performances are achieved using power-law distributions, since nodes are more concentrated in some areas where they may collapse in one single cell, due to the absorption radius. Thus, it is easier 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 performance gain with respect to MAAN.

Finally, we tested the performance of Hivory in real scenarios using a dataset with real-life measurements taken on PlanetLab. Services like the content distribution network CoDeeN [40] are available on such platform. One of CoDeeN's sub projects is CoMon [41], whose goal is monitoring the state of PlanetLab's machines. Each of these services is running over a slice, which is a set of resources distributed on a subset of nodes; each machine may handle more active slices at the same time, making the measurements both heterogeneous and realistic for our tests. Every 5 minutes CoMon records the measurements on all the nodes of PlanetLab. From the whole database we selected eight dynamic attributes: CPU load and usage, CPU frequency, available memory, bits transmitted and received on/from the network per second, and both latitude and longitude of the position of the connected workstations.

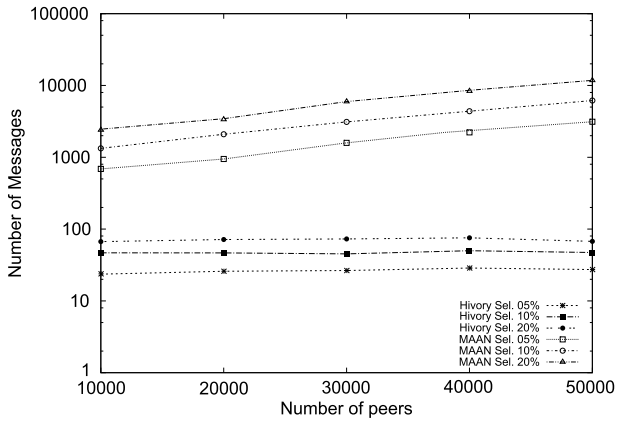
The tests were carried out on a dataset (node centric) of connections made by different users from 20/6/2008 to 23/6/2008. It was obtained by executing a data pre-analysis, which reduced the whole data to 100,000 records for each of the four datasets (a dataset for day). Since not all attributes are present in each record, a subset of 12 attributes was selected. This way, we obtained graphs with 6 levels of two attributes each. The minimum threshold to generate a new level was set equal to 4 in all of these experiments. Moreover, the tests for the Join operation and for query solving have been performed with random permutations of the order of attributes, to assess its impact on the performance.



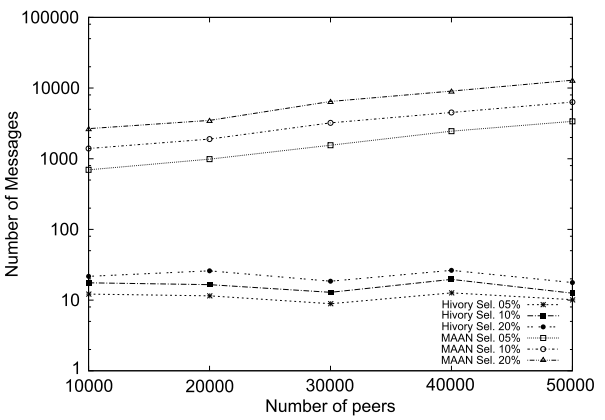
(a) Varying the number of attributes



(b) Varying the network size, Uniform Distribution

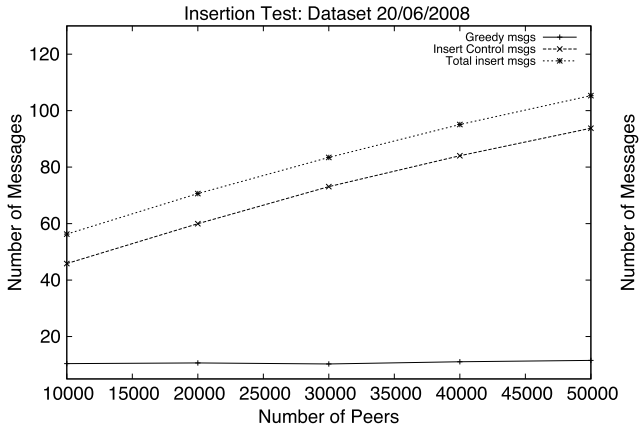


(c) Varying the network size, Power Law Distribution with $\alpha = 2$

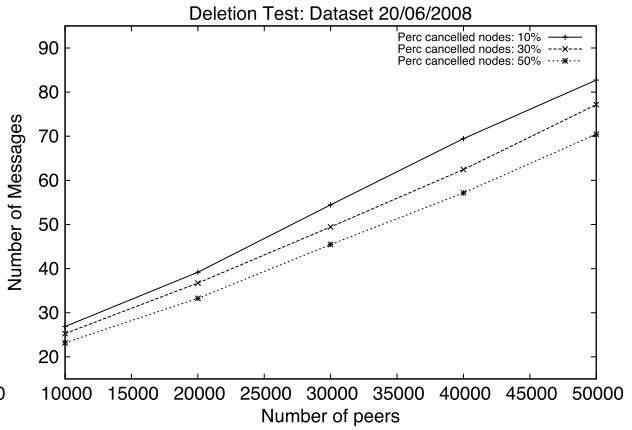


(d) Varying the network size power-low distribution with $\alpha = 3$

Fig. 4. Range queries: comparison between Hivory and MAAN.



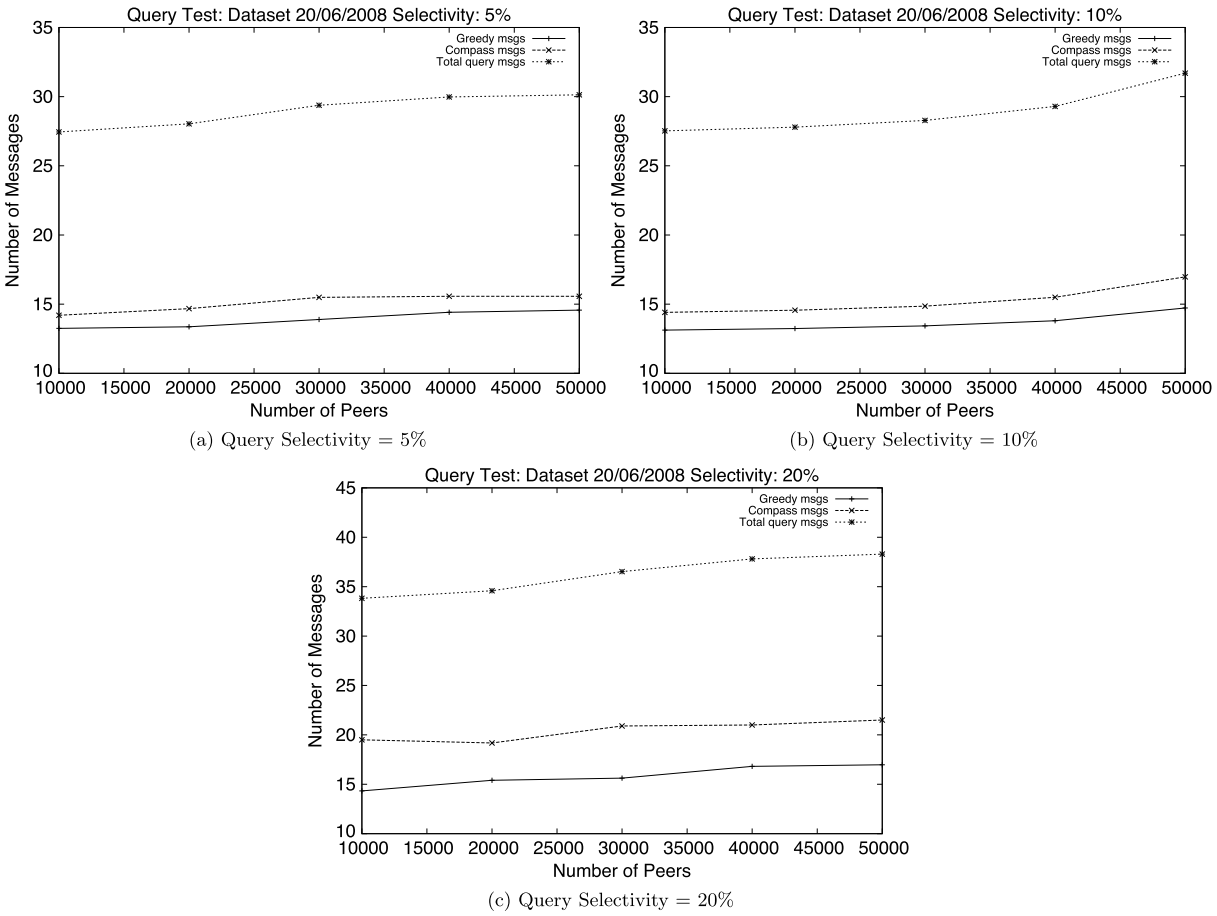
(a) Join operation costs varying the network size



(b) Leave operation costs varying the network size

Fig. 5. Join/Leave operation costs on the PlanetLab dataset.

Fig. 5a shows the average number of messages required to execute a Join operation in an overlay network made up of 50000 nodes, which corresponds to the maximum number of machines in the dataset. Such number of messages was obtained averaging the sum of the Greedy and Control messages needed to manage a peer insertion. It can be seen that the cost related to greedy routing remains under 20 messages even when a network is composed by 50.000 nodes. Instead, the cost due to control messages range from 43 to 105 messages. Like for previous tests, we performed 100 different insertions for each network size and took the final mean value. We observed that the Join operation presents good scalability.



(a) Query Selectivity = 5%

(b) Query Selectivity = 10%

(c) Query Selectivity = 20%

Fig. 6. Query resolution costs on the PlanetLab dataset, varying network size.

Fig. 5b shows the average number of Control messages required to execute a Leave operation of a node from an overlay network made up of 50,000 nodes. It is computed by changing the percentage of nodes removed from the network in each experiment. The graph shows that removing a higher percentage of nodes leads to a lower cost on the single removal operation. This is because the first removal operations are more expensive than the last ones, since the network already contains a smaller number of peers after some of them are erased. The results show that this operation presents good scalability.

Figs. 6a, 6b, 6c show the number of messages required for solving queries with the selectivities used in the previous tests. Such number of messages was obtained averaging the sum of the Greedy and Compass messages needed to solve a query. Again, to reduce the fluctuations due to the randomness of the query and the number of nodes belonging to a AoI, which does not necessarily correspond to its size, 500 queries were randomly generated for each value of selectivity. It can be seen that the query resolution has good scalability. With larger AoIs, the number of exchanged messages gets higher. Such number reaches a maximum value of about 40 when the AoI measures 20% of the area of the level were the query was solved.

8. Conclusions

This paper has presented Hivory, a P2P support for multidimensional range queries based on a hierarchy of Voronoi Tessellations. The search of the matches that solve a query is performed by a top down visit of the hierarchy, by restricting the search space at every level. The theoretical analysis of the complexity of the main operations show that the system requires a small number of messages to solve multi-attribute range queries, with respect to similar state-of-the-art systems. Hivory combines the good properties of classical Voronoi-based networks with the support for a high number of attributes. Even if the system has been evaluated through a simulator, the data for building the hierarchy of Voronoi Tessellations are taken from a set of real-traces of Planet Lab. The analytical observations and the experimental results show a great scalability respect to the number of nodes and the number of attributes. With respect to the latter, the ability of the system to combine the selectivity of all the attributes allows Hivory to exchange a smaller number of messages when dealing with systems characterized by a large number of attributes. As a future work, we plan to deploy Hivory in a real scenario and to

experiment it on further data sets. Another research direction we want to endeavour regards the selection of the super-peers that act as gateway, and in that sense we plan to study high-performance heuristics to decide the best super-peers without hurting the system performance.

References

[1] L. Atzori, A. Iera, G. Morabito, The Internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[2] F. Paganelli, D. Parlanti, A dht-based discovery service for the Internet of things, J. Comput. Netw. Commun. 2012 (2012).
[3] B. Flavio, A.M. Rodolfo, N. Preethi, J. Zhu, Fog computing: a platform for Internet of things and analytics, in: Big Data and Internet of Things: A Roadmap for Smart Environments, 2014, pp. 169–186.
[4] M. Cai, M. Frank, J. Chen, P. Szekely, MAAN: a multi-attribute addressable network for grid information services, in: GRID'03: Proc. of the 4th Int. Workshop on Grid Computing, IEEE Computer Society, Washington, DC, USA, 2003, p. 184.
[5] Y. Shu, B.C. Ooi, K.-L. Tan, A. Zhou, Supporting multi-dimensional range queries in peer-to-peer systems, in: IEEE International Conference on Peer-to-Peer Computing, 2005, pp. 173–180.
[6] A.R. Bharambe, M. Agrawal, 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.
[7] T. Pitoura, N. Ntarmos, P. Triantafillou, Saturn: range queries, load balancing and fault tolerance in dht data systems, IEEE Trans. Knowl. Data Eng. 24 (7) (2012) 1313–1327.
[8] J. Albrecht, D. Oppenheimer, A. Vahdat, D.A. Patterson, Design and implementation trade-offs for wide-area resource discovery, ACM Trans. Internet Technol. 8 (4) (2008) 1–44.
[9] H. Shen, C.-Z. Xu, Leveraging a compound graph-based dht for multi-attribute range queries with performance analysis, IEEE Trans. Comput. 61 (4) (2012) 433–447.
[10] S. Lodi, G. Moro, C. Sartori, Distributed data clustering in multi-dimensional peer-to-peer networks, in: Proceedings of the Twenty-First Australasian Conference on Database Technologies, vol. 104, Australian Computer Society, Inc., 2010, pp. 171–178.
[11] R. Giordanelli, C. Mastroianni, M. Meo, A self-organizing p2p system with multi-dimensional structure, in: Proceedings of the 8th ACM International Conference on Autonomic Computing, ACM, 2011, pp. 51–60.
[12] E. Carlini, L. Ricci, M. Coppola, Flexible load distribution for hybrid distributed virtual environments, Future Gener. Comput. Syst. 29 (6) (2013) 1561–1572.
[13] V. Bioglio, R. Gaeta, M. Grangetto, M. Sereno, Rateless codes and random walks for p2p resource discovery in grids, IEEE Trans. Parallel Distrib. Syst. 25 (2014) 1014–1023.
[14] S. Sioutas, T. Triantafillou, G. Papaloukopoulos, E. Sakkopoulos, K. Tsichlas, Y. Manolopoulos, Art: sub-logarithmic decentralized range query processing with probabilistic guarantees, IEEE Trans. Parallel Distrib. Syst. 31 (1) (May 2013) 71–109.
[15] H.V. Jagadish, B.C. Ooi, Q.H. Vu, BATON: a balanced tree structure for peer-to-peer networks, in: Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30–September 2, 2005, 2005, pp. 661–672.
[16] M.A. Arefin, M.Y.S. Uddin, I. Gupta, K. Nahrstedt, Q-tree: a multi-attribute based range query solution for tele-immersive framework, in: 29th IEEE International Conference on Distributed Computing Systems, ICDCS 2009, 22–26 June 2009, Montreal, Québec, Canada, 2009, pp. 299–307.
[17] S. Ramabhadran, S. Ratnasamy, J.M. Hellerstein, S. Shenker, Prefix hash tree: an indexing data structure over distributed hash tables, in: Proc. of the 23rd ACM Symposium on Principles of Distributed Computing, 2004.
[18] O. Beaumont, A. Marie Kermarrec, L. Marchal, E. Riviere, E. Lyon, Voronet: a scalable object network based on Voronoi tessellations, in: Proceedings of the 21st International Parallel and Distributed Processing Symposium, IPDPS 2007, Society Press, 2007.
[19] F. Banaei-Kashani, 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, ACM, New York, NY, USA, 2004, pp. 304–313.
[20] M. Albano, M. Baldanzi, R. Baraglia, L. Ricci, Voraque: range queries on Voronoi overlays, in: Proceedings of 13th IEEE Symposium on Computers and Communications, July 2008.
[21] L. Rodrigues, F. Araujo, Geopeer: a location-aware p2p system, in: 3rd IEEE International Conference on Network Computing and Applications, NCA'04, 2004.
[22] E. Kranakis, H. Singh, J. Urrutia, Compass routing on geometric networks, in: Proceedings of 11th Can. Conf. on Computational Geometry, CCCG, August 1999.
[23] M. Mordacchini, L. Ricci, L. Ferrucci, R.B.M. Albano, Hivory: range queries on hierarchical Voronoi overlays, in: Proceedings of 10th IEEE P2P Computing Conference, Jul. 2010.
[24] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for Internet applications, IEEE/ACM Trans. Netw. 11 (1) (2003) 17–32.
[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, A scalable content-addressable network, in: Proc. SIGCOMM'01, ACM Press, New York, NY, USA, 2001, pp. 161–172.
[26] A.I.T. Rowstron, 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, Springer-Verlag, London, UK, 2001, pp. 329–350.
[27] A. Andrzejak, Z. Xu, Scalable, efficient range queries for grid information services, in: R.L. Graham, N. Shahmehri (Eds.), Peer-to-Peer Computing, IEEE Computer Society, 2002, pp. 33–40.
[28] R. Baraglia, P. Dazzi, B. Guidi, L. Ricci, Godel: Delaunay overlays in p2p networks via gossip, in: IEEE P2P, 2012, pp. 1–12.
[29] J. Liebeherr, M. Nahas, W. Si, Application-layer multicasting with Delaunay triangulation overlays, IEEE J. Sel. Areas Commun. 20 (8) (2002) 1472–1488.
[30] F. Aurenhammer, Voronoi diagrams-a survey of a fundamental geometric data structure, ACM Comput. Surv. 23 (September 1991).
[31] D. Lee, S. Lam, Efficient and accurate protocols for distributed Delaunay triangulation under churn, in: IEEE International Conference on Network Protocols, ICNP 2008, IEEE, 2008, pp. 124–136.
[32] M. Ohnishi, R. Nishide, S. Ueshima, Incremental construction of Delaunay overlaid network for virtual collaborative space, in: Third International Conference on Creating, Connecting and Collaborating Through Computing, C5 2005, IEEE, 2005, pp. 75–82.
[33] H. Kato, T. Eguchi, M. Ohnishi, S. Ueshima, Autonomous generation of spherical p2p Delaunay network for global Internet applications, in: The Fourth International Conference on Creating, Connecting and Collaborating Through Computing, C5'06, IEEE, 2006, pp. 184–191.
[34] R. Baraglia, P. Dazzi, M. Mordacchini, L. Ricci, A peer-to-peer recommender system for self-emerging user communities based on gossip overlays, J. Comput. Syst. Sci. 79 (2) (2013) 291–308.
[35] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: a programming model for large-scale applications on the Internet of things, in: Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, MCC'13, ACM, New York, NY, USA, 2013, pp. 15–20.
[36] H. Samet, Foundations of Multidimensional and Metric Data Structures, Morgan Kaufmann, 2006.
[37] L. Ferrucci, Un sistema gerarchico basato su voronoi per la risoluzione di query multiattributo, Master Thesis, University of Pisa etd-11102009-151747, December 2009.

[38] M. Jelasity, A. Montresor, G.P. Jesi, S. Voulgaris, The Peersim simulator, http://peersim.sf.net.
[39] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, Planetlab: an overlay testbed for broad-coverage services, SIGCOMM Comput. Commun. Rev. 33 (3) (Jul. 2003) 3–12 [Online], available http://doi.acm.org/10.1145/956993.956995.
[40] L. Wang, K. Park, R. Pang, V.S. Pai, L.L. Peterson, Reliability and security in the CoDeeN content distribution network, in: USENIX Annual Technical Conference, General Track, June 2004, pp. 171–184.
[41] K. Park, V.S. Pai, Comon: a mostly-scalable monitoring system for planetlab, SIGOPS Oper. Syst. Rev. 40 (1) (Jan. 2006) 65–74 [Online], available: http://doi.acm.org/10.1145/1113361.1113374.