

VoroNet: A scalable object network based on Voronoi tessellations

Olivier Beaumont, Anne-Marie Kermarrec, Loris Marchal, Etienne Rivière

▶ **To cite this version:**

Olivier Beaumont, Anne-Marie Kermarrec, Loris Marchal, Etienne Rivière. VoroNet: A scalable object network based on Voronoi tessellations. [Research Report] RR-5833, LIP RR-2006-11, INRIA, LIP. 2006, 2+21 p. inria-00071210

HAL Id: inria-00071210
https://inria.hal.science/inria-00071210

Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

VoroNet: A scalable object network based on Voronoi tessellations

Olivier Beaumont — Anne-Marie Kermarrec — Loris Marchal — Etienne Rivière

N° 5833

February 2006

Thème NUM



ISSN 0249-6399 ISRN INRIA/RR--5833--FR+ENG

INRIA
RHÔNE-ALPES

VoroNet: A scalable object network based on Voronoi tessellations

Olivier Beaumont , Anne-Marie Kermarrec , Loris Marchal , Etienne Rivière

Thème NUM — Systèmes numériques
Projets GRAAL, PARIS et SCALAPPLIX

Rapport de recherche n° 5833 — February 2006 — 23 pages

Abstract: In this paper, we propose the design of VoroNet, an object-based peer to peer overlay network relying on Voronoi tessellations, along with its theoretical analysis and experimental evaluation. VoroNet differs from previous overlay networks in that peers are application objects themselves and get identifiers reflecting the semantics of the application instead of relying on hashing functions. Thus it provides a scalable support for efficient search in large collections of data. In VoroNet, objects are organized in an attribute space according to a Voronoi diagram. VoroNet is inspired from the Kleinberg's small-world model where each peer gets connected to close neighbours and maintains an additional pointer to a long-range node. VoroNet improves upon the original proposal as it deals with general object topologies and therefore copes with skewed data distributions. We show that VoroNet can be built and maintained in a fully decentralized way. The theoretical analysis of the system proves that the routing in VoroNet can be achieved in a poly-logarithmic number of hops in the size of the system. The analysis is fully confirmed by our experimental evaluation by simulation.

Key-words: Overlay network, peer-to-peer, scalability, small-world networks, Voronoi tessellations, Computational geometry.

This text is also available as a research report of the Laboratoire de l'Informatique du Parallélisme `http://www.ens-lyon.fr/LIP`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 —Télécopie +33 4 76 61 52 52

VoroNet: un réseau objet-à-objet extensible basé sur un diagramme de Voronoi

Résumé : Dans ce rapport, nous introduisons VoroNet, un réseau d'objets pair-à-pair reposant sur une décomposition de Voronoi de l'espace de nommage. Nous présentons à la fois le protocole, sa justification théorique ainsi que son évaluation expérimentale. VoroNet se distingue des autres réseaux pair-à-pair par le fait que dans VoroNet, les pairs sont les objets de l'application eux-mêmes et ont des identificateurs liés à la sémantique de l'application, au lieu de dépendre d'une fonction de hachage. Cela permet un support extensible pour une recherche efficace dans une grande collection de données. Dans VoroNet, les objets sont organisés dans l'espace de leurs attributs selon un diagramme de Voronoi. VoroNet est inspiré du modèle petit-monde de Kleinberg dans lequel chaque pair est connecté à des voisins proches ainsi qu'à un nœud distant. VoroNet améliore le modèle initial de Kleinberg car il gère des topologies quelconques et ainsi peut faire face à des distributions de données hétérogènes. Nous montrons que VoroNet peut être construit et maintenu de façon totalement décentralisée. L'analyse théorique du système montre que le routage dans VoroNet peut être effectué avec un nombre de sauts poly-logarithmique en la taille du système. Cette analyse est confirmée par l'évaluation expérimentale, menée sous forme de simulations.

Mots-clés : Réseau pair-à-pair, extensibilité, Réseaux petit-monde, diagramme de Voronoi, Géométrie algorithmique.

1 Introduction

Peer to peer has clearly been recognized as a key communication paradigm to build robust and scalable distributed applications. Searching in large networks is one of the core functionalities offered by peer to peer systems. Among the numerous peer to peer networks that have been proposed in the past 5 years, structured peer to peer networks, such as Pastry [9] or Chord [11], have generated a lot of interest. Most of these overlay networks organize physical nodes in a logical network and provide a scalable support for fully decentralized Distributed Hash Tables (DHT). Such networks heavily rely on the use of hashing functions to ensure load balancing. For example, nodes use hashing functions to get an identifier, so that the identifier space is uniformly populated. Likewise, file contents are hashed to produce the key, used to locate objects later on in DHTs. Such networks offer an exact-match interface which make them natural candidates for file systems or archival systems in which requests target well-identified files. While providing efficient key-based lookup, the query mechanism of such systems is often restricted to exact search. More specifically, they are not built to handle range queries on either one or several attributes, mainly due to the hashing mechanism. Either one attribute is associated to a specific node leading to load unbalance for skewed distribution of the attributes, or a node is made responsible for an (attribute, value) pair. This latter choice might require flooding mechanisms or querying the entire set of possible values for that range.

In this paper, we step away from this form of hash-based peer to peer overlay networks and propose the design of VoroNet, an object-based overlay network based on Voronoi tessellations. VoroNet differs from general-purpose peer to peer structured overlays in that it links application objects rather than physical nodes so that objects with similar characteristics are neighbours in the object to object network, providing a natural support for range queries. In addition, VoroNet does not rely on hashing functions to distribute the load or to evenly populate the identifier space. Each object is held by the physical node hosting it, and each physical node is responsible only for the objects it has published in the overlay. VoroNet specifies a d-dimensional attribute space in which object's virtual coordinates (specifying its identifier) are the attribute values associated to the objects themselves. The attribute space is mapped on to a d-dimension Voronoi tessellation. This may lead to skewed data distribution in the space and yet VoroNet ensures an efficient routing. In this paper, we focus on the special case where $d = 2$.

Inspired by the small-world algorithm proposed by Kleinberg [7], it goes beyond the original proposal by generalizing the approach initially proposed in the context of a grid-based system. In the seminal paper of Kleinberg [7], each node of a grid mesh knows its four neighbours in the grid as well as an additional remote node carefully chosen according to its distance in the grid network. This paper provides clear theoretical bounds on routing performance or node degree distribution as long as the aforementioned assumptions hold. This model is too restricted to be directly transposed to any realistic distribution of objects among the attribute space and any kind of graphs or overlay structure. Instead in VoroNet, we propose to use a similar design where each object neighbour set, formed by its Voronoi neighbours, is enhanced with a long range contact chosen according to a precise distribution of long range link lengths. This paper provides similar theoretical bounds on the routing performance for any distribution of data among the attribute space.

Contributions The contribution of this paper is twofold. First we propose the design of a fully decentralized peer to peer object-based overlay network, relying on Voronoi tessellations,

along with a theoretical analysis and evaluation by simulation. VoroNet is innovative as it links objects rather than peers, enabling the naming space to reflect the application itself and therefore simplifying the range-based search. In addition, VoroNet uses the properties of Voronoi tessellations to handle skewed distribution of data. We propose fully distributed join and leave algorithms, requiring only each object to have a (very) limited knowledge of the system. These algorithms are fully distributed, resilient to calculation degeneracy and evenly distribute the load between peers. In addition they are efficient with respect to network traffic as well. Second, we provide a generalization of the Kleinberg algorithms and show that $O(log^2 N)$ routing performance bounds can be achieved in general. Simulation results confirm the theoretical analysis. Note that the specification of associated query mechanisms is out of the scope of this paper.

Roadmap The rest of this paper is organized as follows, in Section 2 we provide some background both on the Kleinberg's model on which our work is largely inspired as well as Voronoi tessellations. Section 3 presents VoroNet in a nutshell and its associated algorithms. The analysis of VoroNet is presented in Section 4. Section 5 provides simulation results of VoroNet. Before concluding and presenting some perspectives of this work, we survey related works.

2 Background

In this section, we provide some background on the Kleinberg's model and the Voronoi diagrams, both on which VoroNet is based.

2.1 Kleinberg's model

Kleinberg proposed a small-world graph model [6, 7] that provides both poly-logarithmic paths between any two vertices (*small-world* property) and the *navigability* property: greedy decentralized path discovery algorithms can find poly-logarithmic paths in the graph between any couple of nodes. The model is a $n \times n$ grid where every vertex has edges to its four direct neighbours and k (typically one) long-range neighbours. This long-range neighbour is chosen with a probability proportional to $\frac{1}{d^s}$, where d is the link length, *i.e.* the Euclidean distance between the vertex and its remote neighbour. Figure 1 depicts an example of such a network. Only a subset of the long range links are drawn, for the sake of clarity. In [7], the author shows that $s = 2$ enables both *small-world* and *navigability* properties for the 2-dimension grid. A generalization to d-dimensional spaces [3] has shown that, for any d, choosing $s = d$ allows discovered paths of size in $\theta(\log^2 \frac{n}{k})$ using greedy algorithms.

2.2 Voronoi Diagrams

A Voronoi diagram [2, 4] is a partition of space $\mathcal{V}(P)$ associated to a given finite set of points $P = \{p_1, \ldots, p_n\}$ and a distance measure d. If $d(p_1, p_2)$ denote the Euclidean distance between p_1 and p_2, each point p_i is associated with a *Voronoi region* $R(p_i) = \{p | d(p, p_i) < d(p, p_j), \forall j \neq i\}$. The partition of the space $\{R(p_1), \ldots, R(p_n)\}$ is the Voronoi diagram of P. The boundary between two Voronoi regions is a *Voronoi edge*, and a point where three or more Voronoi regions meet is a *Voronoi vertex*.

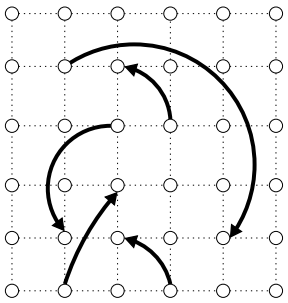


Figure 1: A sample kleinberg network with close links and some long range links.

The dual of the Voronoi diagram $\mathcal{V}(P)$ is the *Delaunay triangulation* $\mathcal{D}(P)$. Let $\mathcal{C}(\Delta_{p_1p_2p_3})$ be the circumscribed circle of the triangle formed by points p_1, p_2 and p_3. The Delaunay triangulation is the set of triangles $\{\Delta_{p_ip_jp_k}\}$ such as $\Delta_{p_ip_jp_k} \in \mathcal{D}(P) \Leftrightarrow \forall p_l \in P - \{p_i, p_j, p_k\}$: $p_l \notin \mathcal{C}(\Delta_{p_ip_jp_k})$.

The relation between $\mathcal{V}(P)$ and $\mathcal{D}(P)$ is straightforward: there exists a Voronoi edge $R(p_i) \leftrightarrow R(p_j) \in \mathcal{V}(P)$ if and only if there is a link between p_i and p_j in $\mathcal{D}(P)$. Figure 2 depicts an example of a Voronoi diagram for a set of points (bold lines) and the associated Delaunay triangulation (dashed lines).

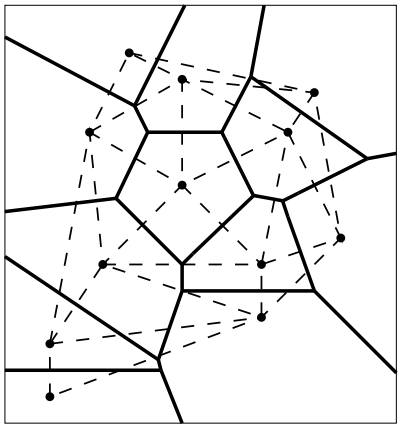


Figure 2: A Voronoi diagram and the associated Delaunay triangulation in the unit cube.

3 VoroNet in a nutshell

We consider a set of objects $\mathcal{O}$. For the sake of clarity, we consider a one-to-one mapping between an object and a physical node. The results can easily be extended to a model in which a node hosts several virtual objects. The overlay design space is a d dimensional space, each dimension representing one attribute. The coordinates of an object in this space are uniquely specified by its values, one for each attribute. In this paper, we focus on $d = 2$ and without loss of generality on the associated unit square[1]. We discuss higher dimensions in

[1] the unit square is the domain $[0 \dots 1] \times [0 \dots 1]$.

the perspectives. More, each node knows a-the value $N_{\max}$, that is the maximal number of objects in the overlay and for which poly-logarithmic routing is guaranteed.

Each object maintains its *view* of the system, *i.e* links to a set of other objects. Each entry of the view is composed of the IP address of the node hosting the object as well as its coordinates in the unit square. The size of this set of neighbours is of order O(1) for any reasonable distribution of objects in the attribute space. Objects in VoroNet are organized according to the Voronoi diagram $\mathcal{V}(\mathcal{O})$. The set of neighbours is composed of: (i) **Voronoi neighbours**, which are the objects whose regions share a Voronoi edge with that object's region; (ii) **Long range neighbour** is an additional remote neighbour providing to the network its small-world characteristics (low diameter and navigability). The way this latter neighbour is chosen in VoroNet is inspired by the method described in [6]: the algorithm for choosing this remote neighbour is depicted further in the paper.

Routing in VoroNet is used both for object insertion and message forwarding. We use a simple greedy and fully deterministic algorithm to route a message from a source to a destination. Note that this algorithm is not sensitive to skewed distribution of objects in the space. We prove that the theoretical number of hops between any two objects is upper bounded by $O(log^2 N_{\max})$, where $N_{\max}$ denotes the maximum number of objects in the overlay.

Joining and leaving the overlay requires to recompute the Voronoi tessellation for a set of objects. This modifies both the Voronoi and long range neighbours. We provide fully distributed algorithms to achieve this: the closest node to the object (being added or deleted) is in charge of recomputing the new partial tessellation. Then, it sends the new diagrams cells to its neighbourhood, so that they can also update their view. The number of associated communications and computations are also of order O(1).

3.1 Object's view management

As previously mentioned, each object in a VoroNet overlay maintains two main sets of neighbours as in Kleinberg's model. In addition, a small set of additional close neighbours is required at each object to ensure routing termination.

The basic structure of VoroNet is a Voronoi diagram. Each object $o \in \mathcal{O}$ has a set of Voronoi neighbours $\{\text{VN}(o)\}$, which are the objects whose Voronoi region share a Voronoi vertex with o's Voronoi region. This set is maintained locally when inserting and deleting objects to the overlay: each $\{\text{VN}(o)\}$ is modified so that the structure formed by $\{\text{VN}(o)\}$ neighbourhood is exactly the Delaunay triangulation of $\mathcal{O}$. Second, to ensure efficient routing, each object o maintains one long range neighbour $\text{LRN}(o)$. The choice of long range range neighbours is drawn from a generalization of Kleinberg's model to Voronoi tessellations in two dimensional spaces. Finally, each object o needs to maintain a set of close neighbours $\{\text{CN}(o)\}$. These neighbours are in a disk of radius $d_{\min}$ centered at o: for each close neighbour $o' \in \{\text{CN}(o)\}$, $d(o, o') \leq d_{\min}$. The radius $d_{\min}$ is very small compared to the attribute space size, but depends on the maximal number of objects in the overlay. We discuss in the perspectives an approach to dynamically adapt this maximal n umber of object. We will see that these neighbours are mandatory to ensure a poly-logarithmic routing cost in spite of irregularities in the object distribution, when many objects are gathered in a small area.

These three types of neighbours are illustrated in Figure 3

Links between Voronoi neighbours and close neighbours are of symmetric nature. Long range neighbours are chosen at a given node and are, by definition, asymmetric. This can be a problem if t, the chosen long range neighbour $\text{LRN}(o)$ leaves the overlay: it will not be

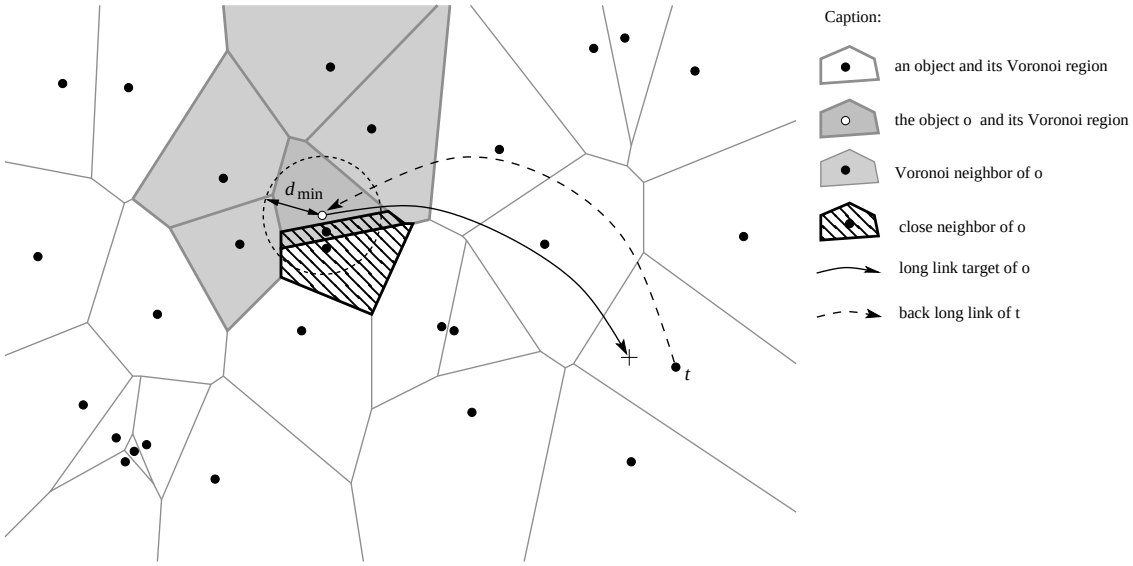


Figure 3: Example of neighbourhood in VoroNet

able to contact o so it can notify a new long range neighbour. To overcome this difficulty, we add o as a special neighbour of t, and will mention it as the "Back Long Range" neighbour, denoted by $\text{BLR}_N(o)$. This extra neighbour is nevertheless not used for routing. Obviously, the size of data structures stored at each object o depends on the distribution of objects in the unit square. However, under reasonable assumptions on the distribution of objects, we have been able to prove that the size of the data structures stored at each node is of order $O(1)$. The details of data structures size analysis are in Section 4.1.

3.2 Routing in VoroNet: small paths and navigability

We now state the key property of VoroNet: using a very simple greedy routing algorithm, we achieve efficient routing (in terms of number of hops). The greedy routing algorithm works as follows: when object o that receives a message m aimed at a target point P, m is forwarded to the neighbour n of o that minimizes the Euclidean distance between n and P, among the neighbours $\{\text{CN}(o)\}$, $\{\text{VN}(o)\}$ and $\text{LR}_N(o)$. If P corresponds to an object, then we find this object. If P does not correspond to an object, we find the Voronoi region where P lies, and the object responsible for this Voronoi region. The routing algorithm is formally described in Section 4. In both cases, the number of hops required to route the message is poly-logarithmic in $|\mathcal{O}|$, the number of objects in the overlay.

3.3 Object insertion and removal

We now introduce the mechanism for inserting a new object in the overlay. Details, proofs of correctness and complexity results are provided in section 4.2. Suppose a new object wants to join the overlay. Its attributes determine the point o where it will be located in the attribute space. We assume that it knows an object x. Starting from this object x, the following operations are performed:

1. The simple greedy routing algorithm is applied to find the object o' satisfying $o \in \mathcal{R}(o')$.

2. o' is responsible for computing the new region associated to o and its neighbourhood. Thus, it provides o with both $\mathcal{R}(o)$ and $\{\text{VN}(o)\}$, and update its own neighbourhood. o' also determines if the responsibility of a back long range neighbour and destination has to be given to o. Note that the target of an object is a point in the 2D space. At a any moment, the long range link points to the object responsible for the region containing this point. We will endeavor to keep this property, so that even if some object disappears from the overlay, the object in charge of the target of the long range link is always the closest from the target point.

3. o computes its close neighbours $\{\text{CN}(o)\}$ (we prove in Appendix **??** that this can be done by considering only its Voronoi neighbours $\{\text{VN}(o)\}$ in the updated Voronoi diagram and their close neighbours).

4. Last, o has to choose a target, *i.e.* a point in the unit square and to find as $\text{LRN}(o)$, the object that is at that time the nearest from that target point.

Let us now consider the operations involved when an object o leaves the overlay. First, since the Voronoi region of o is removed, o is responsible for computing new neighbouring regions and for informing its neighbourhood. Second, if the object o was in charge of a long range link with target point P, belonging to object x, it determines which object o' among its Voronoi neighbours is now in charge of the point P, and delegates the responsibility of the link to o. Note that object x can be reached thanks to the back-long-range link.

4 VoroNet protocol analysis

In this Section, we describe precisely the algorithm used for routing messages and maintaining the overlay structure, and provide the proofs of their correctness. We also focus on the description of the data structure at each node to prove that its cost is in $O(1)$.

4.1 Memory cost at a physical node

First, we investigate the space complexity of VoroNet data structures at each node, under the basic assumption of one object per node.

A neigbhour structure size A neighbour in $\{\text{CN}(x)\}$ and $\text{LRN}(x)$ is an IP adress and identifier (port number). Neigbhours in $\text{BLRN}(x)$ are equivalent, but store also the original point chosen by the neighbour as long range link destination. Voronoi neighbours are a special case where one needs to store their entire region description (set of Voronoi vertices and associated related objects). Theese Voronoi neighbours are hold in a tree like structure, each Voronoi region being asociated to a Voronoi vertex and an object's coordinates and adress. To permit insertion at the responsibility of one object (see Section 3.3), we store both an object's direct neighbours and their neighbours's neighbours. Nonetheless, details of related mechanisms for maintaining and using these data structures are out of the scope of this paper.

Neighbours set size Obviously, the size of data structures stored at each object o depends on the distribution of objects in the unit square. For instance, if all objects belong to $\mathcal{B}(o, d_{\min})$, where $\mathcal{B}(o, d_{\min})$ denotes the disk centered at o whose radius is $d_{\min}$, then all

the objects will belong to $\{\text{CN}(o)\}$. Similarly, if all objects lie on a circle centered at o, then all the objects will belong to $\{\text{VN}(o)\}$, due to the Voronoi diagram definition. Nevertheless, given reasonable assumptions on the distribution of the objects in the unit square, one can prove that the expected number of neighbours, and therefore the expected size of the data structure stored at o are in $O(1)$.

Voronoi neighbours. Even if the number of objects in $\{\text{VN}(o)\}$ may be large in the very special case of co-circular points, the graph induced by the Voronoi diagram is planar and therefore, the expected number of objects in $\{\text{VN}(o)\}$ is bounded by 6 [4]. Thus, for a reasonable distribution of objects, the number of neighbours in $\{\text{VN}(o)\}$ is of order $O(1)$.

Close neighbours Let us assume that we know *a priori* the upper bound $N_{\max}$ on the number of objects that can belong to the overlay at any time, and let us define $d_{\min}$ by $d_{\min} = \frac{1}{\pi N_{\max}}$. If the distribution of objects in the unit square is almost uniform, then the expected number of neighbours (among the N existing objects) in $\mathcal{B}(o, d_{\min})$ is of order $\pi d_{\min} 2N \leq \pi d_{\min} 2N_{\max} = 1$. Therefore, the expected number of objects in $\{\text{CN}(x)\}$ is of order $O(1)$ for any "reasonable" distribution of objects.

Long-range and back-long-range neighbours In the basic setting, each object o chooses exactly one long range neighbour. On the other hand, $\text{BLRN}(o)$ may be large: o is the long range contact of all objects y such that $\text{LRT}(y) \in \mathcal{R}(o)$. If o if far from other objects, it may have a larger number of contacts in $\text{BLRN}(o)$. Nevertheless, high-distance long range contacts have a small probability compared to more local one, thus one can expect that for a reasonable distribution of objects in the unit square, $|\text{BLRN}(x)|$ will be in $O(1)$.

Under reasonable assumptions on the distribution of objects, one can expect that the size of the data structure stored at each node is in $O(1)$. This property will be used when analyzing the set of local functions used to maintain the overlay.

4.2 Analysis of overlay management costs

A set of basic local functions are needed to define the main operations of the protocol. Local here means that their usage incure no communication more than regular and explictiy defined message sending. This part focus on their definition, the analysis of their time complexity and on the network complexity of the protocol as a whole.

4.2.1 Adding a Voronoi region

Let p be a point (to be added as an object). We assume here that we have already found the Voronoi region where p is located: we call o the object such that $p \in R(o)$. The function AddVoronoiRegion(p) is executed by o. AddVoronoiRegion prepares data structures for the new object and for its Voronoi neighbours (new Voronoi regions and subsets of $\text{BLRN}(o)$). One should notice that o is not in charge of creating $\text{LRN}(p)$. o performs three operations:

- It adds p to its Voronoi region and updates modified boundaries of both its Voronoi region and its Voronoi neibhbours's Voronoi region. To add a point to the current local Voronoi diagram, given the Voronoi region that contains this point in the non-updated diagram, we use the algorithm proposed by Sugihara and Iri [13]. It uses local

exploration methods based on combinatorial decisions, and has the strong advantage that it permits reconstruction of local Voronoi regions that are topologically consistent, even if calculation degeneracy takes place. Further details of their algorithm is out of the scope of this paper.

- Then o determines $\{\text{CN}(p)\}$: each new neighbour y of p in the updated Voronoi diagram sends to o the set of its neighbours z (either in $\{\text{CN}(y)\}$ or in $\{\text{VN}(y)\}$) whose distance to p satisfies $d(p,z) \leq d_{\min}$. o then declares p as close neighbour to all nodes y such that $y \in \{\text{CN}(p)\}$. Lemma 1 proves that with this method, p gets all its close neighbours: no object located at a distance lower than $d_{\min}$ is forgotten.

- In order to determine the objects in $\text{BLRN}(p)$, each neighbour y (in the updated Voronoi diagram) sends (and removes them from its own list) the set of its neighbours z (in $\text{BLRN}(y)$) such that $d(p, \text{LRT}(z)) < d(y, \text{LRT}(z))$.

  o then declares to all objects y such as $\text{LRT}(y) \in R(x)$ their new long range neighbour p.

Lemma 1. *Suppose that a new object p has joined the overlay, and that it has just determine its Voronoi neighbours, that is its neighbours in the Voronoi diagram. Then all the close neighbours of p are either some of its Voronoi neighbours, or some of the close neighbours of its Voronoi neighbours.*

Proof. Remember that $R(o)$ denotes the Voronoi region of object o in the current Voronoi diagram, containing the object x.

We call A the set of Voronoi neighbours of x, plus x. We want to prove that $\forall y$, such that $d(x,y) \leq d_{\min}$, there exists an $z \in A$ with $d(z,y) \leq d_{\min}$

We consider two cases:

- Case 1: y is a a Voronoi neighbour of x. Then $z = x$ satisfy the proposition.

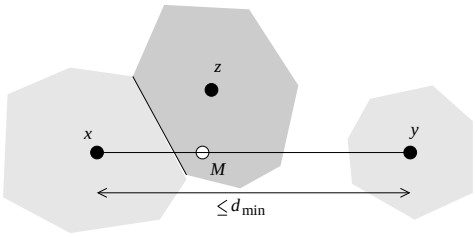


Figure 4: Computing the close neighbours

- Case 2: y is not a Voronoi neighbour of x. We consider the segment $[x,y]$, as illustrated in Figure 4, The parts of this segment belonging to the Voronoi regions of x and y are not contiguous. We consider the object z responsible for the part of the segment contiguous to the part in $\mathit{Region}(x)$, and a point M (which is not an object) in this part of the segment. By triangular inequality, we have: $d(z,y) \leq d(z,M) + d(M,y)$, but as M is in $\mathit{Region}(z)$, we get: $d(z,M) \leq d(x,M)$ together with the previous inequality: $d(z,y) \leq d(x,M) + d(M,y) \leq d_{\min}$. So y is a close neighbour of z, which is a Voronoi neighbour of x. □

Then, o sends to p and to all its Voronoi neighbourhood their updated region and potentially new BLRn()s and {cn()}. Thus, p is inserted in the overlay, and can proceed with the next step, wich is depicted further in the paper. The number of operations and the number of exchanged messages involved by AddVoronoiRegion are both in $O(|\{\text{vn}(o)\}|)$, where $|\{\text{vn}(o)\}|$ denotes the number of neighbours of o (see [12]). Determining $\text{LRn}(x)$ cannot be done by exchanging messages only between o and its neighbours, and will therefore be considered in Section 4.3.2.

4.2.2 Remove a Voronoi region

Let x be an object to be removed from the overlay. The function RemoveVoronoiRegion(x) is executed by object x. It performs the following operations:

- It updates the Voronoi diagram, especially the new boundaries between other objects. Clearly, due to the definition of the Voronoi diagram, only the boundaries between neighbours of x in the Voronoi diagram will be affected by the removal of x. A possible solution consists therefore to compute at x the new Voronoi diagram between its neighbours and to send the boundaries to all nodes $y \in \{\text{vn}(x)\}$.
- x sends a message to all nodes in $\{\text{cn}(x)\}$ to inform them of its close departure.
- x sends the information about each object $y \in \text{BLRn}(x)$ to the object $z \in \{\text{vn}(x)\}$ such that $d(z, \text{LRt}(y))$ is minimal. It also sends a message to y to inform it that z is now its long link neighbour.

The number of operations involved by RemoveVoronoiRegion is of order $O(\text{Nnb}(x))$ nd the number of exchanged messages is of order $O(\text{Nnb}(x))$, where $\text{Nnb}(x)$ denotes the number of neighbours of x before its removal.

4.2.3 Routing mechanism

Distance to a Voronoi region For the analysis of the protocol, one needs to define an additional function, executed at object o for a given point p, wich is:

$$\text{point}r = \text{DistanceToRegion}(p)$$

.

It determines the point $x \in \mathcal{R}(o)$ such that $d(p, r)$ is minimal. If $p \in \mathcal{R}(o)$, then o's coordinates are returned. Obviously, DistanceToRegion requires $\text{O}(|\{\text{vn}(o)\}|)$ computation cost.

Greedy routing Let x be a point in $[0, 1] \times [0, 1]$ and y be an object in the Voronoi diagram. GreedyNeighbour(x) is executed at y and returns z, where z is the object in $\{\text{vn}(y)\} \cup \{\text{cn}(y)\} \cup \text{LRn}(x)$ such that $d(x, z)$ is minimal.

It follows that the number of operations involved by GreedyNeighbour is of order $O(\text{Nnb}(y))$, where $\text{Nnb}(y)$ denotes the overall number of neighbours of y, and does not involve any exchange of messages.

4.3 Overlay maintenance

In this section, we focus on the proofs of the algorithms that deal with maintenance and creation of the VoroNet overlay, namely insertion of new objects and query handling.

4.3.1 Adding an object to the overlay

In this section, we define the set of operations used to add a new object, which involve more than one object and the set of its neighbour. The number of involved objects for each operation will be analyzed in Section 4.4.

In order to add an object, we perform the following operations: we assume that each object x "knows" at least one "starting point" object s in the overlay. Greedy routing is performed from s to find an object y such that $d(z = \textsc{DistanceToRegion}(x), y) \leq \frac{1}{3}d(x, y)$.

This (weak) condition ensures a low (poly-logarithmic) number of steps of Greedy Routing (as we will prove it in Section 4.4), but does not ensure that $y = \mathcal{R}(x)$. In order to add x to the overlay, we therefore first add a fictive object located at z, that will be removed later. Adding z from y is possible since $z \in R(y)$, and as we will prove it in Section 4.4, adding x from z is also possible, since the condition $d(z = \textsc{DistanceToRegion}(x), y) \leq \frac{1}{3}d(x, y)$ ensures that $x \in R(z)$. The function UpdateDataStructure is then used for updating the data structure computed by AddVoronoiRegion. This leads to the algorithm depicted in Algorithm 1. In following algorithms, we assume that the message Spawn(*F, parameters list, destination*) invokes the remote execution of function F with the *parameter list* at node *destination*. We also denote by *CurrentObject* the local object where a procedure is executed.

```
AddObject(x)
  z = DistanceToRegion(x)
  if d(z, x) > 1/3 d(x, CurrentObject) and d(x, CurrentObject) > d_min
  then
    Spawn(AddObject, x, Greedyneighbour(x))
  else
    AddVoronoiRegion(z)
    Spawn(CreateObject, z, Data(z), x)
  return
```

```
CreateObject(z, Data(z))
  UpdateDataStructure(z)
  AddVoronoiRegion(x)
  UpdateDataStructure(x)
  RemoveVoronoiRegion(z)
  LRt = Choose-LRT()
  Spawn(SearchLongLink, x, LRt, Greedyneighbour(LRt))
  return
```

Algorithm 1: Algorithms used for the addition of a point in the overlay

4.3.2 Choosing a long link target

The data structure for new object x is not yet complete, since x needs to choose an object as LRN(x). To do this, x first determines its long-link target point (LRT), using algorithm CHOOSE-LRT that will be discussed later and is depicted in Algorithm 3. We need to find the object z such that LRT $\in R(z)$. To find z, we proceed as for adding an object at position LRT from x. Once LRT has been added, it is possible to determine the object y such that LRT $\in R(y)$. Note that since adding an object requires adding an extra object (to be removed), finding LRN(x) requires to add two objects (to be removed!), as depicted in Algorithm 2.

```
SEARCHLONGLINK(x, LRT)
  z = DISTANCETOREGION(LRT)
  if d(z, LRT) > (1/3)d(LRT, CurrentObject) and
  d(LRT, CurrentObject) > d_min then
    Spawn(SEARCHLONGLINK, x, LRT,
          GREEDYNEIGHBOUR(LRT))
  else
    ADDVORONOIREGION(z)
    Spawn(ESTABLISHLONGLINK, z, DATA(z), x)
  return
```

```
ESTABLISHLONGLINK(z, DATA(z))
  UPDATEDATASTRUCTURE(z)
  ADDVORONOIREGION(LRT)
  UPDATEDATASTRUCTURE(LRT)
  Find y ∈ {VN(LRT)} such that d(y, LRT) is minimal
  REMOVEVORONOIREGION(z)
  REMOVEVORONOIREGION(LRT)
  LRN(x) = y
  return
```

Algorithm 2: Algorithms used for finding the long link for x

The function CHOOSE-LRT is defined by analogy to Kleinberg's work [6]. The probability for object x to choose LRT(x) in $\mathcal{B}(y, dr)$ is chosen to be proportional to $\frac{\pi dr^2}{d(x,y)^2}$. We assume that LRT($x$) may be out of $[0,1] \times [0,1]$, but in any case, LRN(x) will be chosen as the closest object (in $[0,1] \times [0,1]$) to LRT(x). The algorithm CHOOSE-LRT is depicted in Algorithm 3.

```
CHOOSE-LRT()
  Choose a with uniform probability in [ln(d_min), ln(√2)]
  Choose θ with uniform probability in [0, 2π]
  Set δ = (e^a cos(θ), e^a sin(θ))
  Set LRT = x + δ
  return
```

Algorithm 3: Algorithms used for finding LRT(x)

At last, the function RemoveObject(x) simply consists in calling RemoveVoronoiRegion at object x. The function RemoveVoronoiRegion for deleting an object from the overlay does not involve any communication with non neighbours nodes and can be executed with order O(1) communications.

4.3.3 Handling queries

The algorithm for handling queries is depicted in Algorithm 4. It is almost similar to SearchLongLink. We need to find the object y such that $Query \in R(y)$. To find y, we proceed as for adding an object at position $Query$ from x. Once $Query$ has been added, it is possible to determine the object y such that $Query \in R(y)$. Note that since adding an object requires adding an extra object (to be removed), finding LRn(x) requires to add 2 objects (to be removed!), as depicted in Algorithm 4.

```
HandlingQuery(x, Query)
  z = DistanceToRegion(Query)
  if d(z, Query) > (1/3) d(Query, CurrentObject) and d(Query, CurrentObject) > d_min then
    Spawn(HandlingQuery, x, Query, GreedyNeighbour(Query))
  else
    AddVoronoiRegion(z)
    AddVoronoiRegion(Query)
    RemoveVoronoiRegion(z)
    Find y ∈ {vn(Query)} such that d(y, Query) is minimal
    Spawn(AnswerQuery, x, Query, y)
    RemoveVoronoiRegion(Query)
  return
```

Algorithm 4: Algorithms used for handling the Request $Query$ from x

4.4 Proofs

Useful Lemmas

Lemma 2. *Using function* Choose-LRT*, the probability that* LRt(x) *belongs to a small surface* $\mathrm{d}S$ *at distance* d *from* x *is given by* $\frac{\mathrm{d}S}{Kd^2}$*, where* $K = \frac{1}{2\pi \ln(\pi N_{\max}\sqrt{2})}$.

Proof. Using Choose-LRT function, a is chosen with uniform probability in $[\ln(d_{\min}), \ln(\sqrt{2})]$. The long-link target of x is then chosen at a distance $d = e^a$ of x. Let us compute the probability $p_{r\leq d\leq r+\mathrm{d}r}$ for d to be in the interval $[r, r+\mathrm{d}r]$. This is the probability for a to be chosen in the interval $[\ln(r), \ln(r+\mathrm{d}r)]$, which gives:

$$p_{r\leq d\leq r+\mathrm{d}r} = \frac{\ln(r+\mathrm{d}r)-\ln(r)}{\ln(\sqrt{2})-\ln(d_{\min})} = \frac{\ln(1+\frac{\mathrm{d}r}{r})}{\ln(\frac{\sqrt{2}}{d_{\min}})} = \frac{1}{\ln(\frac{\sqrt{2}}{d_{\min}})}\frac{\mathrm{d}r}{r} \tag{1}$$

since a is chosen with uniform probability in $[\ln(d_{\min}), \ln(\sqrt{2})]$.

Let us consider a small surface element located at a distance between r and $r+\mathrm{d}r$ and angle between θ and $\theta+\mathrm{d}\theta$ from x, whose surface is $\mathrm{d}S = r\mathrm{d}r\mathrm{d}\theta$. The probability that LRt(x)

belongs to this element is given by

$$\frac{\mathrm{d}\theta}{2\pi}\frac{1}{\ln(\frac{\sqrt{2}}{d_{\min}})}\frac{\mathrm{d}r}{r} = \frac{1}{2\pi\ln(\frac{\sqrt{2}}{d_{\min}})}\frac{r\mathrm{d}\theta\mathrm{d}r}{r^2} = \frac{\mathrm{d}S}{Kr^2},$$

where

$$K = 2\pi\ln(\pi N_{\max}\sqrt{2}).$$

□

Lemma 3. *The probability for* $\mathrm{LRT}(x)$ *to be chosen in a disk of center* y *and radius* fr*, where* $r = d(x,y)$ *is lower bounded by* $\frac{\pi f^2}{K(1+f)^2}$.

Proof. The distance between x and a point in the disk of center y and radius fr is lower bounded by $(1+f)r$, so that, using previous lemma, probability that $\mathrm{LRT}(x)$ belongs to this disk whose surface is $\pi f 2r 2$ is lower bounded by

$$\frac{\pi f^2 r^2}{K(1+f)^2 r^2} = \frac{\pi f^2}{K(1+f)^2}.$$

The strong point in this result is that this bound does not depend on r. □

Note that all three algorithms AddObject, SearchLongLink and HandlingQuery that involve greedy routing (i.e. that involve more than one object and its neighbours) are based on the same iteration process. The general framework of these function is depicted in Algorithm 5, where Route stands for any function that has to route a message before performing some action (AddObject, SearchLongLink or HandlingQuery), and *Target* denote the target (either the object to be added, the long link to establish or the query).

```
Route(x, Target)
  z = DistanceToRegion(Target)
  if d(z, Target) > (1/3)d(Target, CurrentObject) and d(Target, CurrentObject) > d_min then
    Spawn(Route, x, Target, GreedyNeighbour(Target))
  else
    AddVoronoiRegion(z)
    AddVoronoiRegion(Target)
    Perform some local computations depending on the operation at z
    RemoveVoronoiRegion(z)
    (depending on the operation, RemoveVoronoiRegion(Target))
  return
```

Algorithm 5: Framework for routing for non-local Algorithms from x

First, we need to prove that the algorithm is correct, i.e. that once

$$d(\textsc{DistanceToRegion}(\mathit{Target}), \mathit{Target}) \leq \frac{1}{3}d(\mathit{Target}, \mathit{CurrentObject})$$

or $d(\mathit{Target}, \mathit{CurrentObject}) > d_{\min}$, then z and *Target* can be added via AddVoronoiRegion(z) and AddVoronoiRegion(*Target*). Lemma 4 proves the correctness, then the number of steps (i.e. of calls of Greedyneighbour) will be analyzed. Lemma 5 proves that the number of steps is poly-logarithmic in $N_{\max}$.

Lemma 4. *Let us assume that*

$$d(\text{DistanceToRegion}(\mathit{Target}), \mathit{Target}) \leq \frac{1}{3} d(\mathit{Target}, \mathit{CurrentObject})$$

or

$$d(\mathit{Target}, \mathit{CurrentObject}) \leq d_{\min}.$$

Then z and Target can be successively added using functions AddVoronoiRegion(z) *and* AddVoronoiRegion(Target).

Proof. The case $d(\mathit{Target}, \mathit{CurrentObject}) \leq d_{\min}$ is simple since in this case, both z and *Target* belong to the close neighbourhood of *CurrentObject*, so that insertions can be performed directly.

If $d(z = \text{DistanceToRegion}(\mathit{Target}), \mathit{Target}) \leq \frac{1}{3} d(\mathit{Target}, \mathit{CurrentObject})$, by definition, $z = \text{DistanceToRegion}(\mathit{Target})$ is a point of the Voronoi region $R(\mathit{CurrentObject})$, so that AddVoronoiRegion(z) can be executed at *CurrentObject*.

Once z has been added to the overlay, it remains to prove that $\mathit{Target} \in R(z)$. We know that

$$\forall \text{ object } s, \quad d(\mathit{CurrentObject}, z) \leq d(s, z) \text{ since } z \in R(\mathit{CurrentObject})$$

$$\text{and } d(z, \mathit{Target}) \leq \frac{1}{3} d(\mathit{Target}, \mathit{CurrentObject}).$$

It follows that

$$\forall \text{ object } s, \quad d(z, \mathit{Target}) + d(\mathit{Target}, s) \geq d(z, s) \geq d(\mathit{CurrentObject}, z) \geq 3d(z, \mathit{Target})$$

and therefore

$$d(s, \mathit{Target}) \geq 2d(z, \mathit{Target}),$$

which proves that $\mathit{Target} \in R(z)$ and therefore, *Target* can be added from z to the Voronoi diagram via AddVoronoiRegion(*Target*). □

Lemma 5. *The number of calls to* GreedyNeighbour *in Algorithm 5 is of order* $O(\ln^2 N_{\max})$.

Proof. The proof is directly adapted from the proof proposed by Kleinberg[7] in the case of 2D grids. The main difficulty when analyzing the number of steps needed by Algorithm 5 is that *Target* is not a priori an existing object, so that we cannot converge toward *Target*. Nevertheless, we can prove that the number of steps needed to meet the condition

$$d(z = \text{DistanceToRegion}(\mathit{Target}), \mathit{Target}) \leq \frac{1}{3} d(\mathit{Target}, \mathit{CurrentObject})$$

is of order $O(\ln 2N_{\max})$, and previous lemma proves that once this condition is satisfied, it is actually possible to add the object *Target*.

Let us consider a step of the algorithm, executed at object *CurrentObject*. We will denote by $\mathit{Obj}(\mathit{Target})$ the object such that $\mathit{Target} \in R(\mathit{Obj}(\mathit{Target}))$ (of course, this object is not known at this step) and let us denote $d = d(\mathit{CurrentObject}, \mathit{Target})$

The probability that the long-link target LRt(*CurrentObject*) belongs to the disk of center *Target* and radius $\frac{d}{6}$ is lower bounded (Lemma 3) by

$$\frac{1}{98 \ln(\sqrt{2}\pi N_{\max})}.$$

Let X denote the total number of calls to GREEDYNEIGHBOUR before reaching an object s such that LRT(s) belongs to the disk of center *Target* and radius $\frac{d}{6}$. We have

$$E(X) = \sum_{i=1}^{+\infty} \Pr[X \geq i] \leq \sum_{i=1}^{+\infty} \left(1 - \frac{1}{98 \ln(\sqrt{2}\pi N_{\max})}\right)^{i-1} = 98 \ln(\sqrt{2}\pi N_{\max}).$$

Let us assume now that we have reached an object *CurrentObject* such that LRT(*CurrentObject*) belongs to the disk of center *Target* and radius $\frac{d}{6}$. We will prove that

- either GREEDYNEIGHBOUR(*CurrentObject*) satisfies

$$d(\text{GREEDYNEIGHBOUR}(\mathit{CurrentObject}), \mathit{Target}) \leq \frac{5}{6} d(\mathit{CurrentObject}, \mathit{Target})$$

- or the condition

$$d(z = \text{DISTANCETOREGION}(\mathit{Target}), \mathit{Target}) \leq \frac{1}{3} d(\mathit{Target}, \mathit{CurrentObject})$$

  is fulfilled.

We consider two cases:

- **First case:** $d(\mathit{Target}, \mathit{Obj}(\mathit{Target})) < \frac{d}{2}$.
  Let us denote by $L = \mathit{Obj}(\text{LRT}(\mathit{CurrentObject}))$. In that case,

$$\begin{array}{rcl} d(\mathit{Target}, L) & \leq & d(L, \text{LRT}(\mathit{CurrentObject})) + d(\text{LRT}(\mathit{CurrentObject}), \mathit{Target}) \\ & \leq & d(\text{LRT}(\mathit{CurrentObject}), \mathit{Obj}(\mathit{Target})) + d(\text{LRT}(\mathit{CurrentObject}), \mathit{Target}) \\ & & \quad (\text{since } \text{LRT}(\mathit{CurrentObject}) \in R(L)) \\ & \leq & 2d(\text{LRT}(\mathit{CurrentObject}), \mathit{Target}) + d(\mathit{Target}, \mathit{Obj}(\mathit{Target})) \\ & \leq & \frac{d}{3} + \frac{d}{2} \\ & \leq & \frac{5d}{6} \end{array}$$

  Therefore, in this case,

$$d(\text{GREEDYNEIGHBOUR}(\mathit{CurrentObject}), \mathit{Target}) \leq \frac{5}{6} d(\mathit{CurrentObject}, \mathit{Target})$$

- **Second case:** $d(\mathit{Target}, \mathit{Obj}(\mathit{Target})) \geq \frac{d}{2}$.
  In that case,

$$d(L, \mathit{Target}) \geq d(\mathit{Target}, \mathit{Obj}(\mathit{Target})) \geq \frac{d}{2} \text{ since } \mathit{Target} \in R(\mathit{Obj}(\mathit{Target})).$$

  Moreover,

$$d(z = \text{DISTANCETOREGION}(\mathit{Target}), \mathit{Target}) \leq d(\text{LRT}(\mathit{CurrentObject}), \mathit{Target}) \leq \frac{d}{6}.$$

  Therefore,

$$d(z = \text{DISTANCETOREGION}(\mathit{Target}), \mathit{Target}) \leq \frac{1}{3} d(\mathit{Target}, \mathit{CurrentObject}).$$

Therefore, from *CurrentObject*, after an expected number of $98\ln(\sqrt{2}\pi N_{\max})$ calls to GREEDYNEIGHBOUR, either Algorithm 5 stops because

$$d(z = \text{DISTANCETOREGION}(Target), Target) \leq \frac{1}{3}d(Target, CurrentObject) \text{ is satisfied}$$

or the distance between *CurrentObject* and *Target* has been divided by $\frac{6}{5}$. Let us call a super-step such a sequence of calls to GREEDYNEIGHBOUR.

Since after each super-step, either the algorithm stops or the distance between *CurrentObject* and *Target* has been divided by $\frac{6}{5}$, the number of super-steps is bounded by

$$\frac{\ln(\sqrt{2}\pi N_{max})}{\ln(\frac{6}{5})},$$

since the initial distance is smaller than $\sqrt{2}$ and the algorithm stops as soon as the distance is smaller than $d_{\min}$.

Therefore, by linearity of expectations, the expected number of steps N of the algorithm is given by

$$E(N) \leq \frac{\ln(\sqrt{2}\pi N_{max})}{\ln(\frac{6}{5})} * 98\ln(\sqrt{2}\pi N_{\max}) \leq \alpha \ln^2(N_{max})$$

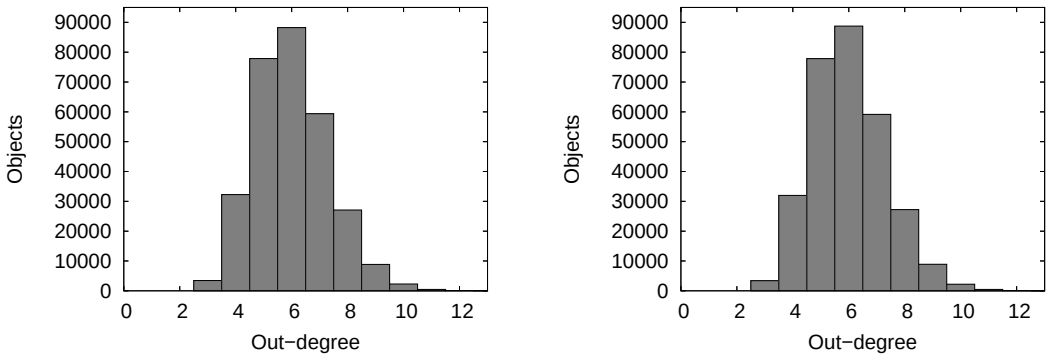
for a suitable choice of α, which achieves the proof. □

5 Experimental results

In this section, we present the results of experimental studies of VoroNet, obtained through extensive simulations. All simulations were run for a 300.000 objects overlay. To determine the impact of object distributions in VoroNet, we used four distributions of object values in the unit square: (i) even distribution and (ii) power-law distribution where the frequency of the ith most popular value is proportional to $\frac{1}{i^\alpha}$. We used three different values for α, 1, 2 and 5, for low, mid and high-skewness distributions respectively. VoroNet was evaluated with two metrics, (i) the distribution of neighbourhood size and (ii) the routing performance in terms of logical hops.

Distribution of neighbourhood sizes In an attempt to confirm the O(1) size of the set $\{\text{VN}(o)\}$, we evaluate VoroNet under different distributions. Figure 5 shows that the number of Voronoi neighbours is as expected centered around 6 regardless of the distribution. Results for low and mid-sparse distributions are equivalent.

Poly-logarithmic routing costs, long range neighbours The objective of VoroNet is to ensure poly-logarithmic routing costs for any distribution of nodes. First, we measure the influence of the distribution on routing cost. For the sake of simplicity, we only consider one long range neighbour per node. Figure 6 depicts the evolution of route lengths, for all four distributions. These routes lengths are mean values for 100.000 random couples of different objects in the overlay, computed after every 10.000 adds of objects to the overlay. One can notice that the routing performance has a poly-logarithmic shape, and is not altered by the distribution of the data set and its sparsity. Second, we prove that the length of VoroNet greedy routes are of order $O(\log^x(|\mathcal{O}|))$ by plotting $log(H)$ (where H is the mean hop count)



(a) Uniform

(b) Sparse, $\alpha = 5$

Figure 5: Distribution of $|\{\text{VN}(o)\}|$ for uniform and highly sparse distributions.

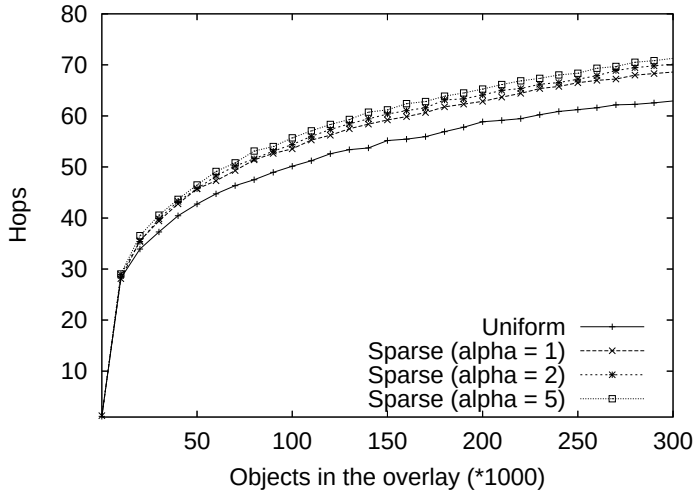


Figure 6: Mean route lengths as a function of the overlay size for the four distributions.

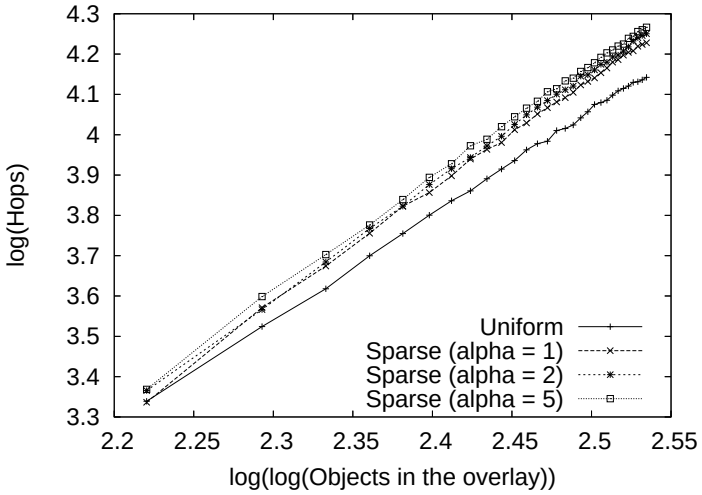
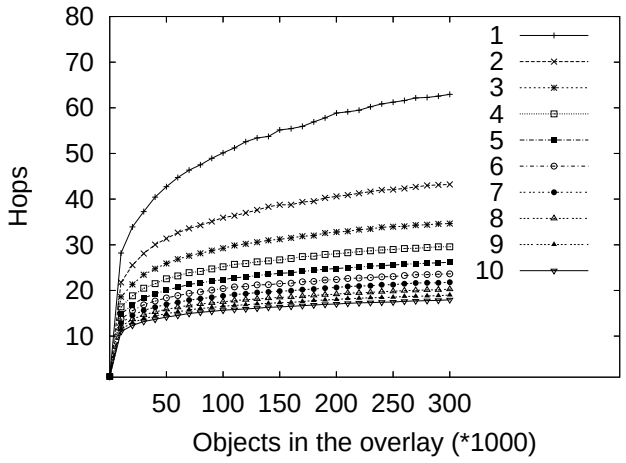


Figure 7: log(H) as a function of log(log($|\mathcal{O}|$)), for different distributions.

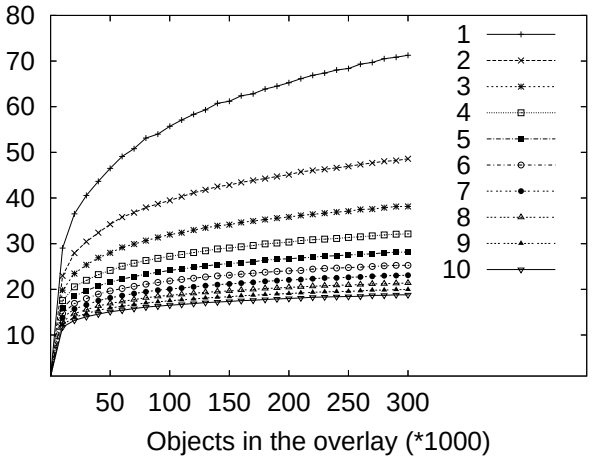
against $log(log(|\mathcal{O}|)$. Moreover, it enables to determine the value of x, the slope of this line. Figure 7 shows that (i) routing costs are poly-logarithmic in the number of objects and (ii) x is close to 2, thus confirming theoretical results. Finally, we analyze the impact of using more than one long range neighbour per node. All links are drawn using the same algorithm. Figure 5 shows that increasing the number of long range neighbours consistently improves the routing performance. However, we observe that the impact is the most significant up to 6 long range neighbours.

6 Related Work

In this section, we survey some routing protocols and overlays based on Voronoi tessellations or their dual, Delaunay triangulations, and extensions of the Kleinberg's model, that are related to our work. First, Steiner and Biersack [10] propose to use Delaunay triangulations in two or three dimensions to build peer to peer networks in the context of virtual community networks. Their approach proposes ideas to deal with the three dimensional case, but does not deal with long range contacts nor calculation degeneracy. Indeed, proposed algorithm is based on edge flipping, which is known to behave badly in presence of calculation degeneracy [12]. Delaunay-based networks are also used in ad-hoc networks, see [8] where the construction of the overlay in constrained by ranges of transmission allowed to nodes (sensors). On the other hand, some work has been done to adapt the Kleinberg model to more general topologies or higher dimensions. It has been demonstrated in [5] that it is possible to extend Kleinberg's model to more general graphs. Also, Barrière *et al.* work [3] on higher dimensions small-world networks is of particular interest in our context. These theoretical frameworks present proofs of feasibility, but no detailed protocols providing desired properties. Another method for adding long range links to a general graph is the HopLevel protocol [1], where long range contacts are created in a lazy way. The main drawback of this method is its lack of theoretical analysis, an aspect VoroNet is particularly aiming at. To the best of our knowledge, VoroNet



(a) Uniform



(b) Sparse, $\alpha = 5$

Figure 8: Study of the influence of the number of long range links on routing.

is the first work to combine these aspects alltogether, using Voronoi diagrams and extension of the Kleinberg's model for poly-logarithmic routing, along with proofs and evaluations.

7 Conclusion and perspectives

In this paper, we have proposed VoroNet, a object-based scalable overlay network rellying on Voronoi tessellations. VoroNet links application objects rather than physical nodes in the attribute space, thus allowing a support for further research on efficient query mechanisms, such as range queries. It is based on Voronoi diagrams, and inheritates their structural properties, such as O(1) direct neighbour set. Moreover, VoroNet is the basis for the proposal of a generalization of Kleinberg's work to generic data distributions, providing poly-logarithmic greedy routing and fair distributions of neighbourhood sizes, even with highly sparse distributions.

This research opens the way to many perspectives. First, the nature of the network enables to design and evaluate rich query mechanisms. The simplest one is a range query on one of the attribute: this query may be represented as a segment in the unit square. Then all objects lying on this segment can be reach easily by forwarding the query along this line, potentially splitting the line in different subset for improved latencies. Moreover, Delaunay triangulation is known to be a t-spanner [8, 4], that is for any subset of the graph, it is possible to efficiently build a spanning tree. Since every square-like subset of the network is itself a Delaunay triangulation, we could use this property to build efficient range query mechanisms. Even richer query mechanisms, such as radius queries, where all objects in a given disk are queried, can also be considered.

Second, we want to investigate some issues on platform dynamism. In the current version of VoroNet, the maximal number of objects $N_{\max}$ must be known in advance, and routing complexity bounds are poly-logarithmic in $N_{\max}$. It would be interesting to deal dynamically with a upper estimation of the number of objects in the overlay. A first solution would consist in having a background process estimating the overall number of objects, increasing the value of $N_{\max}$ by a certain factor if a threshold is reached. All objects would then have to choose another long range contact, according to the new corresponding value of $d_{\min}$. This solution may induce too much communications in the bootstrap process of the network, thus a more dynamic solution should consist in updating only the objects whose neighbourhood is too dense, i.e. objects whose number of objects in their close neighbourhood becomes larger than a given threshold. This solution enables to refine the long range neighbours only where it becomes necessary.

Third, the generalization of VoroNet to upper dimension for the attribute space is also under work. Generalization of Voronoi diagrams in upper dimension exist, but the average number of neighbours is not bounded in this case, so that it is impossible to guarantee that all data structures are of order $O(1)$. Nevertheless, such a generalization may be considered if the objects only need to know the objects in their neighbourhood, and not the description of their boundaries (whose size is of order $O(n^d)$, where n denotes the number of objects in the neighbourhood and d denotes the space dimension).

References

[1] F. Araújo and L. Rodrigues. Long range contacts in overlay networks. In *Euro-par 2005*, pages 1153–1162, Lisbon, Portugal, August 2005. Springer-Verlag, LNCS 3648.

[2] F. Aurenhammer and R. Klein. Voronoi diagrams. In J. Sack and G. Urrutia, editors, *Handbook of Computational Geometry*, pages 201–290. Elsevier Science Publishing, 2000.

[3] L. Barrière, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient routing in networks with long range contacts. In *DISC '01: Proceedings of the 15th International Conference on Distributed Computing*, pages 270–284, London, UK, 2001. Springer-Verlag.

[4] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.

[5] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Could any graph be turned into a small world? *Theoretical Computer Science*, 2005.

[6] J. Kleinberg. Navigation in a small world. *Nature*, 406, 2000.

[7] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symposium on Theory of Computing*, 2000.

[8] X.-Y. Li, Y. Wang, and O. Frieder. Localized routing for wireless ad hoc networks. In *Proc. of IEEE ICC*, 2003.

[9] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, 2001.

[10] M. Steiner and E.-W. Biersack. A fully distributed peer to peer structure based on 3D Delaunay triangulation. In *Algotel 2005, 7emes Rencontres Francophones sur les aspects Algorithmiques des Télécommunications, May 11-13, 2005 - Presqu'île de Giens, France*, May 2005.

[11] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM'01*, 2001.

[12] K. Sugihara and M. Iri. Construction of the Voronoi Diagram for "One Million" Generators in Single-Precision Arithmetic. *Proceedings of the IEEE*, 80:1471–1484, 1992.

[13] Kokichi Sugihara. Robust geometric computation based on topological consistency. In *ICCS '01: Proceedings of the International Conference on Computational Sciences-Part I*, pages 12–26, London, UK, 2001. Springer-Verlag.



Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399