Peer to Peer Multidimensional Overlays: Approximating Complex Structures*

Olivier Beaumont[1], Anne-Marie Kermarrec[2], and Étienne Rivière[2,3]

[1] LaBRI/INRIA Futurs, Bordeaux France
obeaumon@labri.fr
[2] INRIA Rennes Bretagne Atlantique, France
{akermarr,eriviere}@irisa.fr
[3] IRISA/Université de Rennes 1, France

Abstract. Peer to peer overlay networks have proven to be a good support for storing and retrieving data in a fully decentralized way. A sound approach is to structure them in such a way that they reflect the structure of the application. Peers represent objects of the application so that neighbours in the peer to peer network are objects having similar characteristics from the application's point of view. Such structured peer to peer overlay networks provide a natural support for range queries. While some complex structures such as a Voronoï tessellation, where each peer is associated to a cell in the space, are clearly relevant to structure the objects, the associated cost to compute and maintain these structures is usually extremely high for dimensions larger than 2.

We argue that an approximation of a complex structure is enough to provide a native support of range queries. This stems from the fact that neighbours are important while the *exact* space partitioning associated to a given peer is not as crucial. In this paper, we present the design, analysis and evaluation of RayNet, a loosely structured Voronoï-based overlay network. RayNet organizes peers in an approximation of a Voronoï tessellation in a fully decentralized way. It relies on a Monte-Carlo algorithm to estimate the size of a cell and on an epidemic protocol to discover neighbours. In order to ensure efficient (polylogarithmic) routing, RayNet is inspired from the Kleinberg's small world model where each peer gets connected to close neighbours (its approximate Voronoï neighbours in Raynet) and shortcuts, long range neighbours, implemented using an existing Kleinberg-like peer sampling.

1 Introduction

Structure versus search expressiveness. Plethora of peer to peer overlay networks have been proposed in the past years to manage data collection at a large-scale. Peer to peer overlays organize peers in a logical network and are characterized by their underlying structure. As far as data management is concerned, they differentiate each other by the expressiveness and efficiency of the search functionalities they support. The expressiveness of search relates to the way data can be accessed: *(i)* exact search is used to access data objects identified by a unique identifier; *(ii)* attribute-based search enables

* This work was supported in part by French ANR "Masse de données" ALPAGE.

E. Tovar, P. Tsigas, and H. Fouchal (Eds.): OPODIS 2007, LNCS 4878, pp. 315–328, 2007.

to access data using a set of attribute, value pairs; *(iii)* in range queries, the attribute values are specified for a given range At one end of the spectrum lie unstructured overlays in which each peer gets connected to a set of arbitrary neighbours. Such networks rely on constrained flooding techniques to search for data [21]. This provides a way to implement all types of search but such approaches often suffer from lack of efficiency. A query may need to ultimately visit the whole network to ensure exhaustive results. Fully structured overlays lie at the other end of the spectrum. In such networks, peers are organized along a precise structure such as a ring. In DHT-based networks [20], each object gets associated to a given peer. Such networks provide an efficient support for a DHT functionality. However, their expressiveness is naturally limited by the exact-match interface they provide.

We argue that, in order to improve upon the efficiency of expressive queries, the structure of the peer to peer overlay should reflect the application's one. Peers are then application objects and get connected to neighbours (*i.e.* sharing similar characteristics from the application point of view). Such a logical organization provides a natural support for nearest neighbours and range queries. Such peer to peer overlays then support *natively* complex queries. Examples of such approaches are : Sub-2-Sub [26] and Meghdoot [10] for content-based publish and subscribe or Skip-graph based overlays [1,9]. Those structures are however sometimes extremely complex to maintain accurately. For example, maintaining a Voronoï tessellation as in [4] involves a high overhead when the dimension is larger than 2 [6], and is prone to high levels of calculation degeneracy.

Weakening the structure. In this paper we argue that a loose structure is actually enough from the search perspective. What really matters is that each peer gets connected to carefully chosen neighbours, so that the graph can be exhaustively visited. The exact logical structure is not as crucial, provided that its estimation enables correct routing for all requests. In this paper, we propose a general approach based on a Monte-Carlo algorithm to approximate a complex structure, in order to build a loosely structured overlay network. More precisely, we propose an algorithm to approximate the size of Voronoï cells, upon which we build neighbourhood relations.

Contributions. The contributions of this paper are the following. First, we propose a general approach based on a Monte-Carlo method to approximate the size of a Voronoï cell. Then we propose the design and evaluation of RayNet, a weakly structured overlay network, achieving an approximation of a Voronoï tessellation. Following the generic approximation method, each peer in RayNet relies on an epidemic-based protocol to discover its neighbours. Using such a protocol, the quality of the estimation gradually improves to eventually achieve a close approximation of a Voronoï tessellation. This protocol ensures that each peer gets connected to its Voronoï-like neighbours while avoiding the need to accurately compute the exact Voronoï cells, thus keeping the overall overhead low. Each peer in RayNet also maintains a set of long-range links (also called shortcuts) to implement a small-world topology. Efficient (poly-logarithmic) routing in RayNet is achieved by choosing the shortcuts according to a distribution advocated by Kleinberg in [17]. Both links are created by gossip-based protocols. Finally, we evaluate the performance of RayNet through simulations and investigate its performance both in terms of bootstrapping time and routing performance. Note that

implementing the query algorithm is actually out of the scope of this paper, and that we focus on the creation of the overlay itself.

2 Design Rationale

System model. We consider a system composed of n nodes, and a set of objects. We assume that each object is stored on the node that has created it. The overlay is actually linking objects themselves, rather than computing entities. This design choice is similar to the one made for Skip-Graphs based systems [1,9]. Nodes can maintain a set of objects. Although the mapping of objects to physical nodes may be investigated to improve performance, the scope of this paper is to present the object-to-object overlay and its capabilities. Leveraging the presence of multiple objects per node or proposing mapping algorithms of objects to physical nodes that provides better scalability, fault tolerance or performance is therefore left for future work.

We consider a d dimensional attribute space. Each object is exactly identified by a value for each attribute. The attribute values of an object represent the coordinates of the object in the attribute space. This may obviously lead to skewed distribution of objects in the naming space.

We assume that each peer maintains a partial view of the network, called its *view* and consisting of a list of neighbours (IP addresses and coordinates).

Structuring the network using Voronoï diagrams. Figure 1 describes coarsely the targeted structure for a two dimensional data set. A set of objects (black points) is maintained in the distributed application naming space. To achieve a structure that permits nearest neighbour and range queries possibilities, peers having *close* attribute values should be linked in the overlay. Figure 1 shows such links for a sample object o_i. Our general goal for the creation of these links is as follows: for any point p_{target} belonging to the application naming space, for a query that passes through an object o_i, either o_i is the nearest to p_{target} and is the solution, or o_i knows a peer o_j that is nearer to the destination. This property ensures that greedy routing always succeeds, since the distance to the destination point is reduced at each step during the query propagation process.

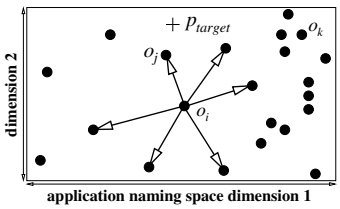


Fig. 1. *Target structure*

A structure that ensures this property is the Delaunay graph, which is the dual of the Voronoï diagram. The Voronoï diagram of a set of generators points $\{p \in \mathbb{R}^d\}$ is a tessellation of $\mathbb{R}^d$ into disjoint cells. Each cell $vc(p_x)$ is composed of all points that are closer to p_x than to any other generator in the set. The links we aim at creating are adjacencies relations between objects cells, and compose the Delaunay graph.

We have already successfully used Voronoï diagrams in the context of routing mechanisms [4] in a structured object-to-object overlay, This overlay provides a native support for range queries and nearest neighbour queries for datasets over two dimensions naming spaces. However, maintaining accurately this structure is extremely costly when the dimension goes over 2 [6]. First, the number of neighbours an object needs to handle is growing exponentially with the dimension.

Second, the maintenance cost to keep *exactly* all these links consistent in spite of nodes and links failure increases accordingly.

However, defining the exact Voronoï cells is more than what is actually needed to ensure that greedy routing succeeds in such a network. What matters is actually the fact that each peer gets connected to its "close" neighbours *along all directions*. Also, imposing a fixed size set of neighbours at each object is desirable for scalability and load balancing purposes.

We base our design on the following observation: *for an object o with neighbourhood consisting of objects whose Voronoï cell shares a boundary with o's cell, the volume of o's cell in the tessellation of all objects is the same as o's cell volume in the Voronoï tessellation of only o and its neighbours*. We are thus interested in discovering neighbours (partial view of the network) $o.view$ for each object o in the system, for which the volume of o's cell in the tessellation of $o \cup o.view$ is minimal. We use a fixed size set of neighbours, and each object exchanges its current view of the network by means of a gossip-based protocol. Figure 2 presents the principle of this evolution: the more peers an object detects, the more opportunities of choosing a peer configuration it encounters to improve its zone approximation. In the following section, we highlight the principles of gossip-based protocols used for overlay construction, presents the biased peer-sampling protocol we use to provide small world characteristics to the overlay (especially for routing efficiency purposes). We then describe the core of our protocol, that is gossip-based construction of *coverage and closeness* at each peer, and the mechanisms that permit this construction, Monte-Carlo Voronoï cell size estimation.

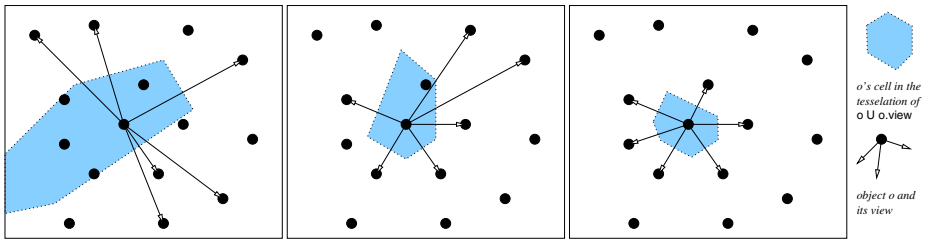


Fig. 2. Desired evolution of an object's neighbourhood: convergence towards the smallest (estimated) Voronoï cell. From random connections (left) to smallest possible zone (right).

3 Approximation Through Gossip

In this paper, we use gossip-based protocols to create and maintain the peer to peer overlay network. Although the focus of this paper is to approximate the neighbourhood at each peer, ultimately routing efficiently (in poly-logarithmic time) through the structure is an important concern. A small-world topology is created to achieve this. In this section we provide some background on small-world networks and gossip-based protocols. We then describe an existing gossip-based protocol that approximates a small-world topology. Finally, we present how we extend the generic gossip-based protocol framework to build the neighbourhood of each peer.

3.1 Small-World Networks

Small-world network models were introduced to investigate the inherent routing capabilities of human relations networks. In such network models, each peer is connected to its closest neighbours in a topology as well as additional long-range contacts, also called shortcuts. Watts and Strogatz [27] introduce such a small-world topology where shortcuts are picked uniformly at random. In 2000, Kleinberg [17] demonstrated that poly-logarithmic routing could be achieved using a greedy algorithm if such shortcuts were chosen according to a specific distribution (d-harmonic). In his work, Kleinberg consider a $n \times n$ grid where every vertex has edges to its four direct neighbours and k (typically one) long-range neighbour(s). This long-range neighbour is chosen with a probability proportional to $\frac{1}{l^d}$, where d is the dimension and l is the Euclidean distance between the vertex and its remote neighbour. These results can be extended to more general topologies and higher dimensions [3,4]

3.2 Gossip-Based Overlay Construction

Gossip-based protocols, first introduced to reliably disseminate events in large systems, have now been recognized as a scalable and reliable basic building block to instantiate and maintain peer to peer overlay networks. Their scalability stems from their simplicity, their ability to capture system dynamics and the emergent properties they lead to. They have been successfully applied to a large number of settings from reliable broadcast [5] to overlay maintenance [8,12,23,25], and from aggregation [15] to system size estimation [22] and are now turned into a generic and sound substrate for building and maintaining large-scale overlay networks [24].

A gossip-based protocol relies on a periodic exchange of information between peers. Such a period is called a *cycle*. Each peer keeps a (usually fixed-size) set of peers, called its *view*. Periodically, each peer picks a target from its view of the system, exchanges some information with it and processes the received information. If the information exchanged relates to neighbourhood, such a protocol creates an overlay network. We focus on such protocols in this paper. A gossip-based protocol is characterized by the following three parameters:

- **Peer selection policy:** each peer p_i chooses periodically a gossip target from its view $p_i.view$;
- **State exchanged:** the state exchanged between peers is membership information and consists of a list of peers (subset of their views);
- **State processing:** upon receipt of the list, the receiving peer merges the list of peers received with its own view to compose a new list of neighbours.

It turns out that these parameters can be tuned so that the resulting graph exhibit properties which are extremely close to those of a random graph [8,12,25], providing a *Peer Sampling Service*: each peer's view contains a set of randomly drawn other peers from the network and this view changes at each cycle. More generally, it has been shown that arbitrary structures can be maintained this way, including fully structured peer to peer overlay networks [11,23,26].

For instance, it has been shown in [7] that peer sampling protocol can be biased in order to approximate the distribution advocated by Kleinberg to improve routing in small-world networks. This can be achieved by simply adapting the *state processing* phase, to keep in the view, a set of peers that exhibits a Kleinberg-like long link length distribution. We use this protocol, called *small-world peer sampling* in the remaining of this document, as the substrate of our protocol, to achieve efficient routing.

3.3 Approximating the Close Neighbourhood: Coverage and Closeness

It has been shown in [11,25] that the same generic gossip protocol can be used to enable each peer to create links to its closest neighbours according to a given proximity metric. The peer selected to gossip with is then chosen as the closest from the view, and the state processing keeps the closest peers from the union of the local and received views. Such a clustering protocol is usually run concomitantly with a peer sampling service in order to ensure connectivity and to leave peers with the ability to cluster nodes[1].

In this paper, we propose to use a generalization of such a protocol to approximate the neighbourhood of a given peer. However, minimizing distances to each peer independently is not sufficient to ensure that the routing will succeed in all directions. Thus, instead of optimizing each item of the view independently, our approach is to decide on a new view as a whole. That means that, at each gossip cycle, set of peers are examined as *configurations* (potential new views) and not independently. To the best of our knowledge, this is the first time such an approach of generalization of gossip-based overlay construction protocols is proposed.

We denote as the *utility* of a new configuration the metric that permits us to decide whether a configuration is better than the current view or not. This utility is the estimation of the Voronoï cell size, as decided by our Monte-Carlo estimation algorithm (see Section 4.1). This metric ensures that (1) closeness is achieved, which means that eventually a peer will get to know peers that are as close as possible to itself and (2) coverage is ensured, *i.e.* eventually each portion of the space surrounding a peer is covered by a neighbour, if such a peer exists in the system.

4 Protocol Details

In this section we provide the details of building and maintaining RayNet. RayNet is based on a gossip-based approach: at each cycle, an object o chooses a gossip partner o_d from its current view (or a subset of its view) of the system to gossip with. After the state is exchanged, o then evaluates if there exists a new view (configuration of objects) that ensures better coverage and closeness. The candidate configurations have thus to be considered as a whole, and peer objects cannot be selected independently.

4.1 View Evolution Using Voronoï Cell Size Estimation

Size of the view. To ensure coverage and closeness, an object uses the estimated volume of its Voronoï cell based on its set of neighbours. Effectively, greedy routing succeeds if

[1] Obviously non uniform topologies would be prone to create disconnected clusters otherwise.

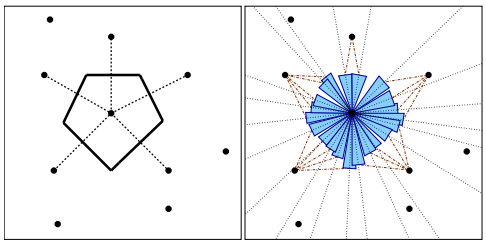


Fig. 3. Illustration of the Monte Carlo method (o is the central point)

o knows neighbours in each possible direction (to get closer to any other target object) and close neighbours (when the target object is close to o). If the volume of the Voronoï cell at o is bounded, then o knows Delaunay neighbours in all directions and if the volume of this cell is the smallest possible one, then these peers are among o's closest neighbours. In general, $2d + 1$ neighbours are enough to get a bounded Voronoï cell. In order to keep extra close neighbours, we set the size c of objects views to $c = 3d + 1$. Moreover, we assume that peers exchange their entire view during a gossip operation.

Monte-Carlo cell volume estimation. Once views have been exchanged, object o needs to estimate the volume of its Voronoï cell, for every possible configuration (on a naive basis; we show in the following Sections that examining *all possible* configurations is not mandatory). The volume of the cell is computed for each configuration. That is, given a set of objects $o.view \cup o_d.view = \{o_1, \ldots, o_n\}$, for each possible configuration $\{o_{i_1}, \ldots, o_{i_c}\}$ of size c, we estimate the volume of the Voronoï cell of o in the tessellation of points $o \cup \{o_{i_1}, \ldots, o_{i_c}\}$. Then, if a new configuration is found, for which the volume of the cell of object o is reduced, this configuration is used as o's new view.

There is no need to effectively compute the cell itself, which would be computationally expensive and prone to high levels of calculation degeneracy. Instead, we propose a new Monte-Carlo method for estimating this volume. Figure 3 presents an illustration of this approach in a two dimensional space. Note that this approach scales to higher dimensions.

A set of R *rays* is created, whose starting point is o and directions are drawn uniformly at random on the unit hyper-sphere. To this end, we use the method described in [18] that provides uniform probability distribution of points on the hyper-sphere. Algorithm 1.left describes the method. Rays (dashed lines starting from o on Figure 3) will act as *probes*, for which we discover the closest intersection point p_{int} lying on the ray r with a (virtual) Voronoï cell of another object in the configuration, this object being the object o_2 for which $\lambda = ||p_{int}, o|| = ||p_{int}, o_2||$ is minimal. For this, the function compDistOnRay() in Algorithm 1.left computes λ for each point. Distances $\lambda = ||p_{int}, o_2||$ are represented by discontinuous lines from o_2 to the intersection p_{int} on Figure 3. Lines (a) to (b) of Algorithm 1.left present the selection of the closest peers for each ray. We keep all λ values for each ray (set Λ), and use them to compute the estimation of the cell volume as follows (line (c) of Algorithm 1.left). Each ray r is associated to a *ball* of radius λ_r whose volume is given by $(BallVol \times (\lambda_r)^d)/R$, where

$BallVol$ is the volume of the unit ball in dimension d. The volume of the estimated cell is the average value, for all rays, of volumes of such balls (the contribution for each ray is represented as grey cones on Figure 3). Such an estimator of the volume of the Voronoï cell is clearly unbiased, so that the estimated volume converges to the volume of the Voronoï cell when $R \rightarrow +\infty$. Nevertheless, the convergence strongly depends on the shape of the Voronoï cell, thus imposing the use of a large enough R (10^3 in the current implementation).

```
calcVolume()
parameters      : config (SET[objects])
begin
    SET[double] Λ ← ∅
    o.rays ← createRays(R)
(a) for double[] r ∈ o.rays do
        double λ ← ∞
        for object o_j ∈ config do
            double l ← compDistOnRay(r,o_j)
            if l < dist then
                λ ← l
(b)     Λ ← Λ ∪ λ
    /* BallVol contains the unit Ball volume in dimension
    d */
(c) return BallVol × Σ_{λ∈Λ}(λ^d) / R
end
```

```
update_naive()
parameters      : o_d.view (SET[objects])
Local variables:
  S : SET[objects]
  vol : double
begin
    o.current_vol ← calcVolume(o.view)
    foreach S ∈ P_c(view ∪ o_d.view) do
        vol ← calcVolume(S)
        if vol < o.current_vol then
            o.view ← S
            o.current_vol ← vol
end
```

Algorithm 1. Monte-Carlo algorithm for estimating the volume of the cell for object o (left) and naive update algorithm for o receiving $o_d.view$ (right).

4.2 Discovery of a New Configuration: Naive Approach

We describe in this section and in Algorithm 1.right the naive approach to select a new view for an object o upon reception of the view $o_d.view$. In order to determine the best view among the set of candidates, we need to estimate the volume of the Voronoï cell of o for the subgraph $\mathcal{S} \bigcup o$ for each possible set $\mathcal{S}$ of c peer objects in the augmented view. That is, each possible subset of size c among $o.view \cup o_d.view$ shall be evaluated for replacement of $o.view$.

Evaluating all $C_{2c}^{c} = \mathrm{O}(c!)$ possible configurations would provide exhaustive and accurate results, though at an unaffordable price. Therefore, we propose in the next Section a more realistic algorithm significantly reducing the overall complexity to a cost that is linear in the space dimension d.

4.3 Discovery of a New Configuration: Efficient, Linear Time Approach

Algorithm 2 presented in this section requires rays for a given object to be chosen once and for all upon creation of the object, in order to save information between configurations' associated cell volumes. Each peer o maintains a bipartite graph $best$ containing on one side peers objects of $o.view$, and on the other side the rays $o.rays$. We denote by $best_O(r)$ the Voronoï neighbour o_p of o according to ray r: it is the node o_p such that a ray issued from o and whose direction is r first reaches the Voronoï cell of o_p (this entry is never empty). Similarly, we denote by $\{best_R(o_p)\}$ the set of rays for which o_p is the current Voronoï neighbour of o (this set may be empty).

The objective is as follows: to compute o's new view, for each object o_p in $o_d.view \cap \overline{o.view}$ (*i.e.* all peers for which $\{best_R(o_p)\}$ does not contain any information), we determine the set of rays for which o_d is the Voronoï neighbour of o in the augmented view Voronoï diagram. This operation is described by lines (a) to (b) of Algorithm 2. Peers found to be a Voronoï neighbour of o for a given ray are stored in the set $improve$, which has the same semantic as $best_O$, except that entries for some rays can be empty.

On line (c), either $improve$ or $best_O$ has information, for each ray, about which peer in the augmented view is a Voronoï neighbour of o. The next step is to compute to which extent each peer is needed in the new configuration. More precisely, given a peer o_x, we compute the volume of the cell of o with all peers *but* o_x (lines (c)-(d)). If the volume of the cell increases dramatically, that means that peer o_x was mandatory to ensure closeness and proximity. On the other hand, if the volume remains the same, then peer o_x has no contribution to coverage nor closeness.

Note that, unlike the naive method (Algorithms 1), it is not necessary to iterate through all peers of the tested configuration to find the peer with the smallest λ value. This information is usually contained in either $best_O$, if such a peer lies in $o.view$, or in $improve$, if such a peer is a candidate peer from the distant view. The only case when one needs to iterate through all peers is when the best known peer for a given ray is o_x, the currently ignored peer.

Volumes associated to each peer (*i.e.* the volume without that peer in the configuration) are stored in the map $volumes$. This map is then sorted by decreasing volume values : starting from entries of peers that contributes highly to coverage and closeness, to entries of peers that have no or few contribution to coverage and closeness. The new

update()
parameters : $o_p.view$ (SET[objects]) /* distant view */
Local variables:
improve (map ray $\rightarrow$ object) init $\emptyset$ /*improve has the same semantic as $best_O$*/
volumes (list of pairs (object,volume)) init $\emptyset$
begin
(a) **foreach** *ray* $r \in o.rays$ **do**
double $best_\lambda = \perp$
object $imp = \perp$
foreach *object* $o_j \in (o_d.view \cap \overline{o.view})$ **do**
$\lambda \leftarrow$ distOnRay(r, o_j)
if $\lambda < \begin{cases} best_O(r) & \text{if } best_\lambda = \perp \\ best_\lambda & \text{if } best_\lambda \neq \perp \end{cases}$ **then**
$imp \leftarrow o_j$
$best_\lambda = \lambda$
if $best_\lambda \neq \perp$ **then**
(b) improve[r] $= imp$
(c) **foreach** *object* $o_x \in o.view \cup (o_d.view \cap \overline{o_i.view})$ **do**
(d) volumes $\leftarrow$ volumes $\cup$ pair(o_x,calcVolumeImproved($best \cup improve$, $(o_d.view \cap \overline{o_i.view}) \setminus o_x$))
sort volumes by decreasing volume
$o.view \leftarrow \{volumes_1.o, \ldots, volumes_c.o\}$
update $best_O$ and $best_R$
end

Algorithm 2. Update of object's view $o.view$: efficient approach. Sets $best_O$ and $best_R$ are constructed and coherent i.r.t. the current $o.view$ when starting the algorithm.

configuration is built from the c peers that presents the maximum contribution, *i.e.* peers of the first c entries of $volume$.

The cost of the approach is as follows: there are up to $(r \times c)$ calls to method distOnRay(), if all c candidates were unknown to the current peer, and up to $(2 \times c)$ calls to calcVolume(). Each call to distOnRay() has cost 1: it is a fixed size set of scalar products. Each call to calcVolume() takes $r \times (1 + \frac{2\times c-1}{2\times c})$ operations, where the term $\frac{2\times c-1}{2\times c}$ stands for the few cases where the "best" peer is the currently ignored peer o_x (on average, $\frac{1}{2\times c}$ occurrences per call). The overall cost is thus $\simeq 5(r \times c)$ operations, where r is a constant and c only depends on the dimension of the naming space d, *i.e.* $c =$O(d). The overall cost of the improved update algorithm is thus O(d) operations.

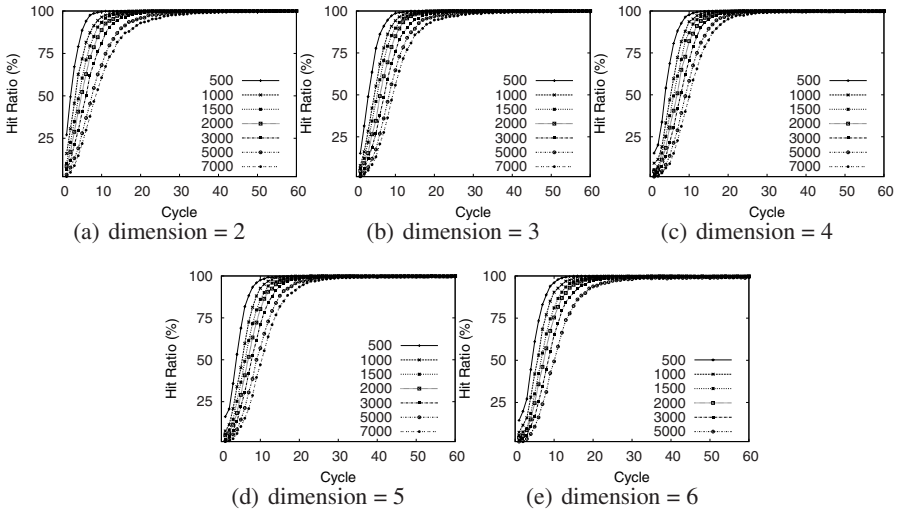
5 Experimental Evaluation

In this section, we evaluate RayNet along two metrics: (1) the time needed by a chaotic system to converge towards an overlay where all routes succeed and (2) when such an overlay is created, how many steps are required by greedy routing from any object to the nearest object of a target point, as a function of system size. Expected results are respectively: (1) a fast convergence and self-organization towards full success for routing requests and (2) a poly-logarithmic evolution of the route size according to the size of the system, thanks to the small-world peer sampling layer.

We developed a simulator using Java, and ran simulations for populations of objects ranging from 500 to 7.000 objects. The dimension of the object naming space d ranges from 2 to 6. All objects points are drawn uniformly at random in this space. For all experiments, $r = 10^3$ rays were used to estimate cell volumes, and $3 \times d+1$ neighbours are kept at each object. At each cycle, two exchanges take place, one for the small-world peer sampling layer (8 peers out of 20 maintained peers are sent), the other for the coverage and closeness layer (exchange of views). Also, for the first two cycles, each peer selects randomly 10 peers from the small-world peer sampling layer and assess them for potential inclusion in a new configuration to bootstrap the coverage and closeness level.

Bootstrapping the overlay. First, we evaluate the time RayNet takes to converge towards an overlay state where every routing request succeed. The overlay is initialized to a random graph for the small-world peer sampling layer, and no peer for the coverage and closeness layer. This makes sense as bootstrapping from a chaotic state is the worst case for gossip-based overlay construction mechanisms. More, following the proposal of [16] (with successful instantiations such as [13,23,26]), this represents the case where a distributed application needs the rapid instantiation of a routing substrate on top of a peer sampling layer. This experiment shows that our proposal fits perfectly in this scope, while being obviously applicable to long-term runs.

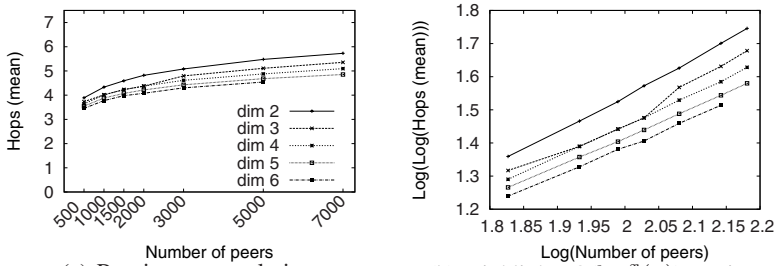
Figure 5 presents the results for all dimensions, and for different object population sizes. *Hit ratio* denotes the proportion of routes that succeed onto exactly the object that is nearest to the query destination. At each cycle, 20.000 random (object, destination point) pairs are tested. As expected, the hit ratio increases with the number of exchanges. In addition, perfect routing is achieved within at most 30 to 35 cycles, regardless of the dimension. Note that the cycle period is to be defined by the application,



(a) dimension = 2 (b) dimension = 3 (c) dimension = 4

(d) dimension = 5 (e) dimension = 6

Fig. 4. Evolution of routes *hit ratio* for dimensions 2 to 6



(a) Routing cost evolution. (b) Highlights O($\log^x(n)$) routing.

Fig. 5. Routing efficiency (data for (a) and (b) is the same)

and depends on the trade-off between quality of service and cost on computing entities. It is possible however to bootstrap faster by using shorter periods at the beginning and to decrease it when steady state has been reached. In a dynamic scenario, objects would join gradually, and each object can use several short-term gossip exchanges to insert themselves faster in the overlay. The hit ratio converges slightly slower if there are more nodes. Note that this does impact neither the time a node would need to join an already constructed overlay, nor the complexity of local self-organization of the structure. Figure 5 shows that approximating the structure does not impact routing correctness.

Routing efficiency. The second evaluation metric is the routing efficiency: how many routing steps are needed on average to route between a source object an a destination point. This metric is directly impacted by the performance of the small-world peer sampling substrate as well as the quality of the close neighbourhood. It has a great impact on the efficiency of search mechanisms that can be proposed over the RayNet overlay. Figures 5.(a) and 5.(b) present the evolution of the routing costs as a function

of the number of objects, for several dimensions. Particularly, Figure 5.(b) plots the $\log\log$(mean hops) as a function of $\log$(objects). The line shape of Figure 5.(b) proves that route sizes are poly-logarithmic in the number of objects, as expected by the small world characteristic of RayNet. We consider this property as being the key to scalability of future search mechanisms. The reason why higher dimensions present smaller routing paths is due to the fact that the size of the view at each objects increases linearly with the dimension d: for final steps (where small world links are not used), more possibilities are available for deciding on the next step of the route, which obviously slightly decreases the number of steps that use links from the coverage and closeness layer. This shows that approximating the structure does not impact routing performance.

6 Related Works

Other protocols have been proposed to deal with multidimensional data querying and complex query support in large scale distributed systems. Structured overlays with exact-search interface have been used to implement range queries [2] even if such overlays are not *natively* addressing such capabilities. These approaches present relatively high costs of maintenance of the structures: either a second indexing mechanism based on objects rather than nodes is built, whose cost is added to the cost of the structured overlay itself, or a single index is used but with the need for an implicit load balancing algorithm to replace the inherent load balancing provided by hash mechanisms. RayNet steps away from these approaches by being designed with the native support for complex queries in mind from the beginning.

The authors previously used *exact* Voronoï diagram (in dimension 2 only) for the design of VoroNet [4]. This structured overlay organizes objects in an overlay that, like RayNet, reflects exactly the application semantic space, by using the *exact* Delaunay graph (and not an approximation) as the basic routing substrate along with explicit small-world construction. Using such exact structures, while providing efficient search and routing, suffers from two drawbacks: (1) maintaining the Delaunay complex for higher dimensions would be too costly and (2) maintenance in two dimensions in face of churn is a difficult (yet not unsolvable) problem. RayNet addresses these two problems by (1) using an estimation of Voronoï cells as the basis for the construction of a subset of the Delaunay complex and (2) using Gossip-Based, self-organizing protocols that embed both protocol construction and re-organization in the same protocol, relieving the need for explicit fault tolerance mechanisms.

Skip-Webs [1] are multidimensional data structures that enable querying of data on a large scale, with multidimensional attributes. Nonetheless, maintaining such a structure in presence of churn may have a tremendous cost. Note that using Gossip-based techniques to construct this "Skip-List-like" structure could benefit from Gossip-based overlay construction protocols, such as the ones used for uni-dimensional data in GosSkip [9].

7 Conclusion

In this paper, we presented a new approach to create overlays that reflect a distributed application shared objects naming space. Organizing application objects in a distributed

data structure based on the Delaunay graph of object points is sound but costly. We show that accuracy is not crucial and that reasonable approximation does not impact routing in such a structure. This paper presents the design and evaluation of RayNet, a peer to peer overlay that links objects in a multi-dimensional naming space, where each object's view is drawn according to an estimation of its Voronoï cell size using a Monte-Carlo algorithm. Gossip-based protocols are extensively used to provide self-organization properties and routing efficiency. Simulation results convey the soundness and efficiency of the approach.

Next steps in this research are the following. First, we would like to investigate complex queries mechanisms for which RayNet was designed to be the support. At the moment, range queries are implemented by using constraint flooding; refined mechanisms can be proposed by carrying some state on the query dissemination messages. We would like to investigate the scalability to higher dimensions of the mechanisms provided by [19]. Second, although gossip-based protocols are inherently resilient to nodes failures, little research has been done on securing such protocols. Following the early proposal of [14], we would like to investigate mechanisms to make our protocol resilient to adversary behaviours and detect malicious peers.

Acknowledgments. We would like to thank François Bonnet, who helped us to integrate the gossip-based small-world peer sampling in RayNet [7] and Philippe Duchon, whose comments and expertise helped us on early stages of this work.

References

1. Arge, L., Eppstein, D., Goodrich, M.T.: Skip-webs: efficient distributed data structures for multi-dimensional data sets. In: PODC 2005, pp. 69–76 (2005)
2. Aspnes, J., Kirsch, J., Krishnamurthy, A.: Load balancing and locality in range-queriable data structures. In: PODC 2004, pp. 115–124 (2004)
3. Barrière, L., Fraigniaud, P., Kranakis, E., Krizanc, D.: Efficient routing in networks with long range contacts. In: Welch, J.L. (ed.) DISC 2001. LNCS, vol. 2180, pp. 270–284. Springer, Heidelberg (2001)
4. Beaumont, O., Kermarrec, A.-M., Marchal, L., Rivière, É.: VoroNet: A scalable object network based on voronoi tessellations. In: IPDPS 2007 (March 2007)
5. Birman, K.P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y.: Bimodal multicast. ACM Transactions on Computer Systems 17(2), 41–88 (1999)
6. Boissonnat, J.-D., Yvinec, M.: Algorithmic Geometry. Cambridge University Press, Cambridge (1998)
7. Bonnet, F., Kermarrec, A.-M., Raynal, M.: Small-world networks: From theoretical bounds to practical systems. In: Tovar, E., Tsigas, P., Fouchal, H. (eds.) OPODIS 2007. LNCS, vol. 4878, pp. 315–328. Springer, Heidelberg (2007)
8. Eugster, P.T., Guerraoui, R., Handurukande, S.B., Kouznetsov, P., Kermarrec, A.-M.: Lightweight probabilistic broadcast. ACM Transactions on Computer Systems 21(4), 341–374 (2003)
9. Guerraoui, R., Handurukande, S.B., Huguenin, K., Kermarrec, A.-M., Fessant, F.L., Rivière, É.: Gosskip, an efficient, fault-tolerant and self organizing overlay using gossip-based construction and skip-lists principles. In: IEEE P2P, Cambridge, pp. 12–22. IEEE Computer Society Press, Los Alamitos (2006)

10. Gupta, A., Sahin, O.D., Agrawal, D., Abbadi, A.E.: Meghdoot: content-based publish/subscribe over p2p networks. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 254–273. Springer, Heidelberg (2004)
11. Jelasity, M., Babaoglu, O.: T-man: Gossip-based overlay topology management. Engineering Self-Organising Systems 1(15) (2005)
12. Jelasity, M., Guerraoui, R., Kermarrec, A.-M., van Steen, M.: The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 79–98. Springer, Heidelberg (2004)
13. Jelasity, M., Kermarrec, A.-M.: Ordered slicing of very large-scale overlay networks. In: IEEE P2P, Cambridge, pp. 117–124 (September 2006)
14. Jelasity, M., Montresor, A., Babaoglu, O.: Towards secure epidemics: Detection and removal of malicious peers in epidemic-style protocols. Technical Report UBLCS-2003-14, University of Bologna, Department of Computer Science, Bologna, Italy (November 2003)
15. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Transactions on Computer Systems 23(3), 219–252 (2005)
16. Jelasity, M., Montresor, A., Babaoglu, O.: The bootstrapping service. In: ICDCSW 2006: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems, Lisboa, Portugal, p. 11 (July 2006)
17. Kleinberg, J.: The small-world phenomenon: An algorithmic perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing, Portland, OR, USA, pp. 163–170 (May 2000)
18. Knuth, D.E.: Seminumerical Algorithms. In: The Art of Computer Programming, vol. 2, Addison-Wesley, Reading, Massachusetts (1981)
19. Liebeherr, J., Nahas, M.: Application-layer multicast with delaunay triangulations. IEEE Journal on Selected Areas in Communications, Special Issue on Network Support for Multicast Communication 40(8), 1472–1488 (2002)
20. Lua, E.K., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A survey and comparison of peer-to-peer overlay network schemes. In: IEEE Communications survey and tutorial (March 2004)
21. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS 2002: the 16th international conference on Supercomputing, New York, pp. 84–95 (2002)
22. Merrer, E.L., Kermarrec, A.-M., Massoulié, L.: Peer to peer size estimation in large and dynamic networks: A comparative study. In: 15th IEEE HPDC, Paris, pp. 7–17 (June 2006)
23. Montresor, A., Jelasity, M., Babaoglu, O.: Chord on demand. In: IEEE P2P, Washington, pp. 87–94 (2005)
24. Rivière, É., Baldoni, R., Li, H., Pereira, J.: Compositional gossip: a conceptual architecture for designing gossip-based applications. ACM SIGOPS Operating Systems Review, special issue on Gossip-based Networking (October 2007)
25. Voulgaris, S.: Epidemic-Based Self-Organization in Peer-to-Peer Systems. PhD thesis, Vrije Universiteit, Amsterdam (November 2006)
26. Voulgaris, S., Rivière, É., Kermarrec, A.-M., van Steen, M.: Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks. In: IPTPS, Santa Barbara (February 2006)
27. Watts, D.J., Strogatz, S.H.: Collective dynamics of small world networks. Nature 393, 440–442 (1998)