# DDC: A Dynamic and Distributed Clustering Algorithm for Networked Virtual Environments based on P2P networks

Moritz Steiner and Ernst W. Biersack
Institut Eurécom
Sophia–Antipolis, France
{steiner,erbi}@eurecom.fr

***Abstract*— We present a distributed algorithm for the clustering of peers in a Networked Virtual Environment (NVE) that are organized using a peer-to-peer (P2P) network based on the Delaunay triangulation. The algorithm is dynamic in the sense that whenever a peer joins or leaves the NVE, the clustering will be adapted if necessary by either splitting a cluster or merging clusters. The main idea of the algorithm is to classify links between adjacent peers into short intra-cluster and long inter-cluster links.**

**The advantages of clustering are multiple: clustering allows to limit queries to the peers of a cluster avoiding to flood the entire network. Since clusters can be seen as a level of abstraction that reduces the amount of information/detail exposed about the NVE, clustering allows for faster navigation in the NVE and reduces the number of messages a node receives when he travels through the NVE.**

***Index Terms*— clustering, Delaunay triangulation, NVE, P2P, social networks.**

## I. Introduction

Networked Virtual Environments (NVEs) are computer generated synthetic worlds that allow simultaneous interactions between multiple participants. Especially with the boom of Massive Multiplayer Online Games (MMOGs), NVEs are becoming more and more popular nowadays. However, to create a large scale NVE, the traditional client-server model does not scale.

Recently peer-to-peer (P2P) architectures [1], [2], [3] have been proposed to solve the scalability issue. Yet, a P2P approach to build a large scale virtual environment raises a number of issues. In shared virtual reality applications, peers are characterized by a position in a virtual space. Unlike to DHT, this position can be chosen and changed freely by the peers. Typically, in P2P-based NVEs each peer only knows its direct neighbors and some other peers in its attention radius. The only way to handle a query in such a scenario is to forward it to all neighbors, that is to flood the entire virtual world as long as no answer to the query is found. Flooding requires a lot of bandwidth and is not very scalable.

Our approach to achieve scalability is based on the properties of a social network of people who are using the system. A social network is a set of people with some interactions among them [4], [5]. In recent years, social networks have been studied with respect to properties such as degree distribution, the small world effect, the search ability, and clustering. If people interact in the virtual world as described by the social networks for the real world, they will tend to form clusters. In this case, one identifies these clusters and limit the scope of queries to the respective cluster, avoiding to flood the entire network.

The second motivation for clustering peers is to abstract a cluster of peers into a single object and to allow faster navigation in the virtual world: not from peer to peer but from cluster to cluster. In this case, another network on top of the overlay of the nodes is needed: A *cluster overlay* network.

Another benefit using clusters is to avoid that a peer $p$ that travels through the NVE must receive a HELLO message from every peer that gets within its proximity, which consumes a lot of bandwidth and significantly slows down the traveling peer $p$. Instead, it would be desirable that only one peer among all the peers of a clusters sends information about the cluster to a traveling peer $p$. Only if $p$ comes so close to the cluster that it could be a part of it, $p$ gets information about the inner structure of the cluster.

This paper proposes a distributed and dynamic algorithm for the clustering of peers (DDC) in a NVE based on a fully distributed P2P network. DDC is not only applicable to NVEs, but to all proximity graphs with a clustered distribution of the nodes where each node only knows its immediate neighbors. Some possible fields of application for DDC are:

- Network positioning systems relying on coordinates [6], [7], [8], [9], [10]. In such systems each host has assigned a position in a $d$-dimensional space. The network distance between two hosts is estimated by computing the distance using their coordinates.
- Latency-driven content delivery networks [11]. DDC can not only ease the detection of a group of nearby hosts, but also the accurate selection of the location for the replicas.
- End system multicast such as Narada [12], which is targeted towards medium sized groups. By clustering peers and establishing multicast communication first between clusters, and then within each cluster, Narada can scale to much bigger groups.

The DDC algorithm presented relies on DTON [13], a fully distributed P2P overlay network based on 3-dimensional

Moritz Steiner is also affiliated with the Department of Mathematics and Computer Science, University of Mannheim, Germany.

Delaunay triangulation (DT) [14], [15]: The peers are seen as points in the DT and the links between them as the edges of the DT.

Given $n$ 3-dimensional points, the 3d Delaunay triangulation connects the points into non–overlapping tetrahedra that fill the convex hull of these points in such a way that the sphere criterion is satisfied, i.e. the circumsphere of the four vertices of any tetrahedron of the DT contains none of the given $n$ points in its interior. The dual of the Delaunay triangulation is the Voronoi Diagram [16], [15], which assigns to each of the $n$ points a region that is nearer to that point than to any other point. The resulting region will form a pattern of packed convex polyhedra covering the whole convex hull of the $n$ points. If all point pairs whose region share a common plan are joined by straight lines, the result is a triangulation of the convex hull of the $n$ points. This triangulation is known as Delaunay triangulation (Figure 1).

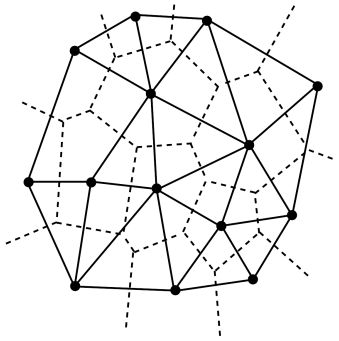

Fig. 1. Delaunay Triangulation and Voronoi diagram (dashed lines) in two-dimensional space.

Given the peers are organized in an overlay that is built according to the rules of the DT, some clustering approaches have previously been presented [17], [18], [19]. The basic idea is to classify the links that connect the peers in short intra–cluster links and long inter–cluster links. However, up to now no dynamic distributed clustering algorithm exists that allows the insertion and the deletion of peers without having to restart the complete cluster computation. The insertion of new peers or the deletion of existing peers can result in the merging of two clusters or the split of one cluster in two parts.

The rest of the paper is organized as follows: Section 2 contains definitions necessary for the description of our algorithm, called DDC. In Section 3 we describe related work in clustering, in Section 4 we describe DDC. In Section 5 we evaluate the algorithm and present some simulation results. Finally, Section 6 concludes the paper.

## II. Definitions

Suppose $P = p_1, \ldots, p_n$ is a point set in 3 dimensions. $p_{ni}$ is the $i$.th component of point $p_n$. The convex hull of 4 affinely independent points from $P$ forms a *Delaunay simplex* (a tetrahedron) if the circumscribed sphere of the simplex contains no point from $P$ in its interior. The union of all Delaunay simplices (tetrahedra) forms the *Delaunay triangulation* $DT(P)$. If the set $P$ is not degenerate – if no five points of $P$ are co-spherical – then $DT(P)$ is unique.

Every point $p \in P$ represents the position of a peer in the NVE. The peers have to have the ability to freely choose and change their coordinates in the virtual world.

Two peers $p_i \in P$ and $p_j \in P$ are linked if they are part of at least one common *Delaunay tetrahedron*. Let $d(p_i, p_j)$ be the Euclidean distance between them:

$$d(p_i, p_j) = \sqrt{\sum_{d=1}^{3} (p_{id} - p_{jd})^2}.$$

Two linked peers are called *Delaunay neighbors*. Connections are therefore based on the spatial relationship among the nodes in the NVE.

The set of points $P$ is partitioned into $m$ subsets $CL_1, \ldots, CL_m$, called clusters.

$$\bigcup_{i=1}^{m} CL_i = P$$

$$CL_i \cap CL_j = \emptyset, \quad 0 < i, j \leq n, i \neq j$$

Each peer $p$ maintains two lists: a list $D(p)$ of its *Delaunay neighbors* and a second list $C(p)$ of its *cluster neighbors*. Additionally, peer $p$ stores the identifier $cid(p)$ of the cluster it belongs to. Note that $C(p) \subseteq D(p)$ and that $C(p) \subseteq CL_{cid(p)}$. $C(p) = D(p)$ if peer $p$ is a *inner–cluster* peer, that means all neighbors of peer $p$ are in the same cluster than peer $p$ itself, which implies that all links incident to peer $p$ are *intra–cluster* links.

For a peer $p$, let $Mean(p)$ be the mean length of connections from $p$ to its neighbors $p_i \in D(p)$ in the Delaunay triangulation. That is

$$Mean(p) = \frac{\sum_{i=1}^{\|D(p)\|} d(p, p_i)}{\|D(p)\|}.$$

For a peer $p$, let Dev($p$) be the standard deviation of the length of these links. That is

$$Dev(p) = \sqrt{\frac{\sum_{i=1}^{\|D(p)\|} (d(p, p_i) - Mean(p))^2}{\|D(p)\| - 1}}.$$

## III. Related Work

Cluster analysis has been a research topic for decades. However, most of the algorithms only deal with static data and are centralized. For our problem, we need a distributed clustering algorithm that can handle join and leave of peers. Also, each peer $p$ that executes the algorithm only knows its direct neighbors $D(p)$ and not all $n$ peers in the system.

DTON [13] implements the Delaunay Triangulation in 3d. Our clustering algorithm DDC needs the graph constructed by DTON to cluster the peers.

### A. Threshold Based

Kang and others [18] presented a clustering algorithm relying on the $DT$. The main idea is to remove Delaunay edges whose length is greater than a threshold $t$, and in a second step to remove clusters whose number of objects is less than a given number $cn$. This centralized algorithm could be adopted to a distributed one, but the main disadvantage remains: the thresholds $t$ and $cn$ are global values, if there exist high–density and low–density clusters, they are not recognized properly.

### B. Density Deviation Based

Estivill–Castro et al. [17] first suggested a density based criterion for nodes organized via a $DT$, referred to as *long–short* criterion. The link $p_ip_j$ that connects two points $p_i, p_j$ $0 < i, j \leq n$, $i \neq j$ that are neighbors in the DT is a "short" intra–cluster link – connecting points inside a cluster – if

$$d(p_i, p_j) < Mean(p_i) - w \cdot Dev(p_i) \tag{1}$$

else it is a "long" inter–cluster link – connecting points in different clusters.

The idea behind this criterion is to combine spatial proximity and spatial density. In a Delaunay triangulation, a point $p$ on the border of a cluster has a much larger value $Dev(p)$ since an inner cluster point, since $p$ has both, short distances to neighbors in the same cluster and long distances to neighbors that are not in the same cluster.

## IV. Dynamic Distributed Clustering

In this section we describe our dynamic and distributed clustering algorithm (DDC). Figure 2 shows the result of DDC for a set of 1000 peers.

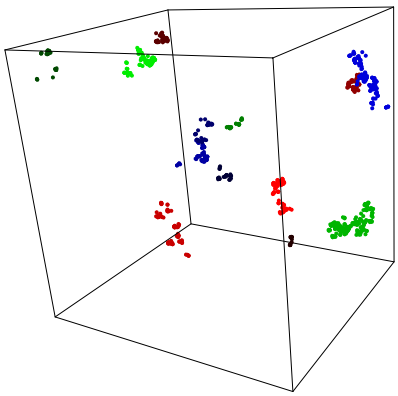

Fig. 2. An example of 1000 peers clustered with DDC.

### A. Distribution

The *long–short* criterion (1) must be adapted to avoid errors in the cluster determination. In the version presented in [17], two Delaunay neighbors $p_i$ and $p_j$ may classify their connecting link $p_ip_j$ differently. Peer $p_i$ may come to the result that it is an intra–cluster link, while peer $p_j$ may classify the same link $p_jp_i$ as inter–cluster link.

DDC uses a weighted average of the local criteria of both peers: The link from $p_i$ to $p_j$ is a intra–cluster link if

$$d(p_i, p_j) < \frac{Mean(p_i) - w \cdot Dev(p_i) + Mean(p_j) - w \cdot Dev(p_j)}{2} \tag{2}$$

else it is a inter–cluster link.

### B. Dynamic

The main difference of DDC compared to [17] is that in our field of application there is no global view and that the nodes are inserted and removed dynamically, which may result in *splitting* or *merging* of existing clusters.

In the following, we describe how the insertion of a new peer will lead DDC to adapt the clustering.

**Algorithm 1**: Insertion (executed by peer $p$)

```
1  c ← closest(D(p))
2  if intra-cluster(c,p) then
3      C(p) ← (C(c) ∩ D(p)) ∪ c
4      cid(p) ← cid(c)
5      foreach pi ∈ C(p) do
6          send message to pi, to make it do:
7          C(pi) ← C(pi) ∪ p
8      p.redetermination ()
9  else
10     cid(p) ← new unique cluster id
```

The first and second peer in the NVE each form a cluster of their own. Every newly arriving peer $p$ checks if it is near enough to its closest Delaunay neighbor $c$ (line 1) to join its cluster according to the used *long–short* criterion (line 2).

Due to the join of peer $p$, the density near $p$ and therefore near every peer $p_i \in D(p)$, expressed by $Mean(p_i)$ and $Dev(p_i)$, changes. First, peer $p$ updates its cluster neighbor list $C(p)$ by adding all those peers $p_i \in D(p)$ with $cid(p_i) = cid(n)$ (line 3) and notifies them so they can update their respective cluster neighbor list (line 5-7). Second it redetermines the cluster repartition of its neighbors (line 8).

We now focus on the *redetermination* procedure (Alg. 2) which is executed by the newly inserted peer $p$.

**Algorithm 2**: Redetermination (executed by peer $p$)

```
1  cid ← new unique cluster id
2  foreach n ∈ D(p) do
3      if n ∈ C(p) and not intra-cluster(n,p) then
4          C(p) ← C(p) \ n
5          n. split(cid)
6      if n ∉ C(p) and intra-cluster(n,p) then
7          C(p) ← C(p) ∪ n
8          n. merge(cid(n),cid(p))
```

*1) Split:* Peer $n$ and peer $p$ are too far away to be connected via an intra–cluster link (Algorithm 2, line 3). This means that peer $p$ was inserted that so close to peer $c$ that peer $n$ is not any more near enough to $c$ to be in the same cluster as peer $c$. In this case, it seems that the cluster containing the peers $c$, $n$ and $p$ must be split (Figure 3).

We now consider peer $n \in C(n)$ on the reception of the *split* message from peer $p$ (Alg. 3). If this is the first time peer $n$ gets this *split* message (line 1), it updates its cluster identifier to the unique $cid$ generated by peer $p$ (line 2) and checks if a peer $n_i \in C(n)$ passes the *long-short* criterion (line 4). The peers that pass, will all execute *split* and change their cluster identifier to $cid(n)$. The peers that fail the test are removed from $C(n)$ (line 7) and keep their old cluster identifier $cid(c)$. This is mainly the case for the peer that executed the *redetermination* procedure (Alg. 2) and sent the first *split* message.

**Algorithm 3**: Split($cid$) (received by peer $n$)

```
1 if cid(n) ≠ cid then
2     cid(n) ← cid
3     foreach ni ∈ C(n) do
4         if intra-cluster (ni,n) then
5             ni. split (cid)
6         else
7             C(n) ← C(n) \ ni
```

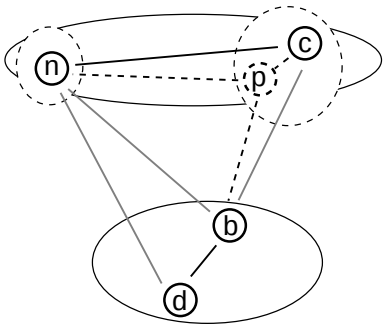


Fig. 3. Split of the cluster $cn$ due to the arrival of peer $p$. The solid lines show the links and the clusters before the split, the dashed lines after wards.

It might happen that peer $c$ and peer $n$ are too far away to be connected by an intra–cluster link but nevertheless are part of the same cluster because there exists a path of intra–cluster links connecting them. In this case the recursive *split* message finds its way back to peer $n$ and all peers of the cluster get the new cluster number $cid(n)$. The cluster keeps its form and is not split but simply renamed.

*2) Merge:* Peer $n$ and peer $p$ are close enough to be connected via an intra–cluster link, but they are not part of the same cluster yet (Algorithm 2, line 6). That means peer $p$ was inserted in between two existing clusters $CL_{cid(n)}$ and $CL_{cid(c)}$ and interconnects them (Figure 4).

We now consider peer $n \in C(p)$ on the reception of the *merge* message from peer $p$ (Algorithm 4). If this is the first time peer $n$ gets this *merge* message (line 1), it changes its cluster identifier $cid(n)$ to $cid(c)$ (line 2) and tells every peer $n_i \in C(n)$ to propagate the new cluster identifier $cid(n) = cid(c)$ to their respective cluster neighbors $C(n_i)$ (line 4+5). The sender of the *merge* message, peer $ni$, is added to $C(n)$ (line 8). This results in a merge of cluster $cid(n)$ and $cid(c)$ to cluster $cid(c)$, all peers of the cluster $cid(n)$ change their cluster identifier to $cid(c)$.

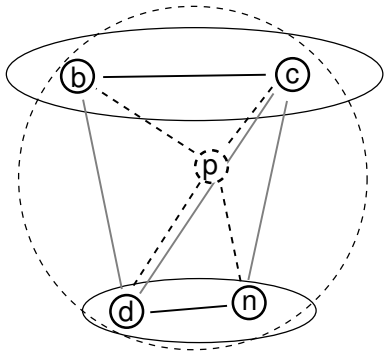


Fig. 4. Merge of the cluster $CL_{cid(c)}$ and $CL_{cid(n)}$ due to the arrival of peer $p$. The solid lines show the links and the clusters before the split, the dashed lines after wards.

**Algorithm 4**: Merge($oldcid$,$newcid$) (received by peer $n$)

```
1 if cid(n) = oldcid then
2     cid(n) ← newcid
3     foreach ni ∈ D(n) do
4         if ni ∈ C(n) then
5             ni. merge (oldcid,newcid)
6         else
7             if intra-cluster (n,ni) then
8                 C(n) ← C(n) ∪ ni
```

The cluster redetermination in case of a peer departure works similar to the case of a peer joining: Either the cluster of the closest peer – who handles the departure in the Delaunay triangulation – must be split or it must be merged with another cluster.

### C. Proof of termination

Each peer $p$ that joins has one closest neighbor $c$, only. Either peer $p$ does not join $c$'s cluster, in this case the algorithm ends, or peer $p$ joins $c$'s cluster and launches the *redetermination* procedure (Algorithm 2), which is only executed once per join or leave event.

At most $|C(p)| - 1$ *split* messages are sent by peer $p$ to all cluster neighbors except to node $c$.

Recursively, a receiver $n$ of a *split* message sends at most $|C(n)| - 1$ new *split* messages. Out of which $t$ messages ($t = |D(p) \cap D(n)|$) are directly ignored. Eventually, every peer of the cluster got one message, the algorithm terminates

here since the terminating condition (Algorithm 3, line 1) is always false. This is a simple "flooding" of the cluster concerned, whereas no peer runs the *split* procedure twice and sends messages a second time.

Exactly the same statements are true for the *merge* procedure. The cluster concerned is simply renamed, while it is assured that no peer executes *merge* twice (Alg. 4, line 1).

## V. Simulation Results and Evaluation

### A. Generation of a Clustered Peer Distribution

To evaluate DDC, first clusters of peers must be generated. For this purpose, we use the so called Lévy Flight [20] that produces a random walk through the plane or the space where the Lévy distribution [21] determines the step size. For the details see [22].

### B. The choice of the clustering threshold $w$

Since no global view of the NVE exists, the optimal value for $w$ cannot be derived from the peer distribution but must be chosen in advance.

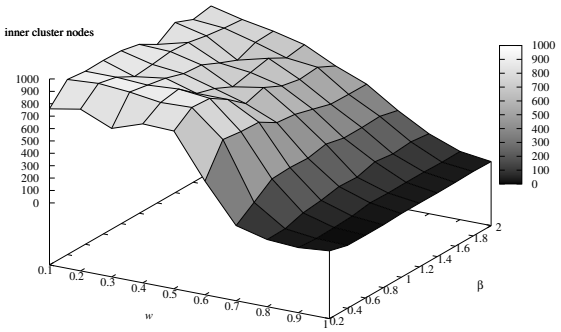


Fig. 5. The number of inner–cluster nodes as function of $w$ and $\beta$ (Average values of 10 runs with 1000 nodes each).

Values between 0 and 1 are reasonable for $w$. To find a value suitable for distributions created with a Lévy Flight, we run simulations for $w = 0.1 \ldots 1.0$ with step size 0.1 and $\beta = 0.2 \ldots 2.0$ with step size 0.05. As it can be noticed in figure 5 the results are independent of $\beta$. Thus, DDC is well adapted to all possible degrees of clustered node distribution. The results depend on $w$, in the figure a "knee" at $w = 0.6$ is observable. The number of inner–cluster nodes (all nodes $n$ for which $C(n) = D(n)$) is stable for $0 < w < 0.6$ and begins to decrease at $w \approx 0.6$. For $w < 0.6$ there exists only one cluster in most cases, therefore the number of inner–cluster nodes is equal to the total number of nodes. The same observations as for the number of inner–cluster nodes can be made for the number of clusters and the number of cluster neighbors per node. Therefore, we come to the conclusion that the choice $w = 0.6$ is suitable for determining clusters in distributions generated with the Lévy Flight.

### C. Difficult cases

In Section V-B we can show that DDC, using the improved criterion (2), deals well with all sorts of node distributions generated with the Lévy Flight. However, there will be always cases where automatic clustering may not perform completely satisfactory. In the following, we present one, for another case we refer to [22].

*Bridges between clusters:* If two clusters are connected via a bridge (Figure 6) the two clusters can not be distinguished using local knowledge. With the help of the clustering criterion it is possible to assume that long links are links between peers in different clusters, because they are too far away to be in the same cluster. Whereas we assume that short links are always intra–cluster links, which is not always true. Sometimes these very short links can form *bridges* between clusters. These bridges are not recognized properly, since only local (one hop) knowledge is available: clusters connected by bridges are seen as one cluster by the *merge* procedure. For the application in a NVE that does not matter, because we assume that peers acting in the NVE do not deliberately form bridges, which do not give any benefit to them.

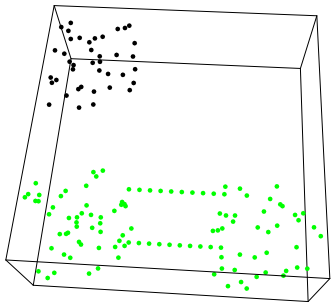

Fig. 6. Two clusters connected by two bridges are not properly distinguished.

## VI. Conclusion and Future Work

This paper presents DDC, a dynamic and distributed algorithm to cluster nodes in a P2P–based NVE. Clusters of different shapes and density are detected. The dynamic approach copes with peers insertion and deletion without having to restart the cluster detection from scratch. DDC tests locally if the density changed so much that a cluster has to be split or that two cluster have to be merged.

We are currently investigating how the clustering threshold $w$ can be chosen automatically by DDC. As we have seen in figure 5, the right choice of $w$ is crucial. Since the range of possible values for $w$ is very small, we envision the following iterative solution: DDC is run for $w = 0.1, \ldots, 1.0$ with step size $\Delta = 0.2$. For each value of $w$ considered, DDC computes the average cluster size $cs(w)$ and takes as suitable value the value $w$ for which $cs(w - \Delta) - cs(w)$ is maximal. Goosip-based algorithms for computing totals and averages in a distributed way already exist, see e.g. [23], and DDC can use them to determine the best $w$ in a fully distributed manner.

Future work also includes dealing in an efficient way with peer movements in the virtual worlds.

## REFERENCES


[1] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world," in *International Conference on Parallel and Distributed Techniques and Applications*, 2003.

[2] M. Steiner and E. W. Biersack, "A fully distributed peer to peer structure based on 3d delaunay triangulation," in *Algotel - Septièmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, May 2005, pp. 93–96.

[3] S.-Y. Hu and G.-M. Liao, "Scalable peer-to-peer networked virtual environment," in *SIGCOMM Workshops*, August 2004.

[4] M. E. J. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," in *PNAS*, vol. 99, 2002, pp. 2566–2572.

[5] K. A. Lehmann and M. Kaufmann, *Random Graphs, Small-Worlds and Scale-Free Networks*, ser. LNCS. Springer, 2005, vol. 3485, ch. 6, pp. 57–76.

[6] T. E. Ng and H. Zhang, "A network positioning system for the internet," in *USENIX 2004 Annual Technical Conference*, 2004, pp. 141–154.

[7] M. Costa, M. Castro, A. Rowstron, and P. Key, "Pic: Practical internet coordinates for distance estimation," in *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 178–187.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2004, pp. 15–26.

[9] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proceedings of the Internet Measurement Conference (IMC)*. Miami Beach, Florida, USA: ACM, October 2003.

[10] M. Pias, J. Crowcroft, S. Wilbur, and T. H. S. Bhatti, "Lighthouses for scalable distributed location," in *Proceedings of $2^{nd}$ International Workshop on Peer-to-Peer Systems (IPTPS)*, ser. LNCS. Berkeley, CA, USA: Springer, February 2003, pp. 278–291.

[11] M. Szymaniak, G. Pierre, and M. van Steen, "Latency-driven replica placement," in *SAINT '05: Proceedings of the The 2005 Symposium on Applications and the Internet (SAINT'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 399–405.

[12] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system Multicast," *IEEE Journal on Selected Areas in Communication, Special Issue on Networking Support for Multicast*, vol. 20, October 2002.

[13] M. Steiner, "Structure and algorithms for the collaboration between peers and their application in solipsis," Master's thesis, University of Mannheim and Institut Eurécom, 2005.

[14] B. Delaunay, "Sur la sphère vide. A la mémoire de Georges Voronoi." *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, vol. 7, pp. 793–800, 1934.

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

[16] G. Voronoï, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire: Sur quelques propriétées des formes quadritiques positives parfaites." *Journal für die Reine and Angewandte Mathematik*, vol. 133, pp. 97–178, 1907.

[17] V. Estivill-Castro, I. Lee, and A. T. Murray, "Criteria on proximity graphs for boundary extraction and spatial clustering," in *PAKDD '01: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. London, UK: Springer-Verlag, 2001, pp. 348–357.

[18] I.-S. Kang, T. wan Kim, and K.-J. Li, "A spatial data mining method by delaunay triangulation," in *GIS '97: Proceedings of the 5th ACM international workshop on Advances in geographic information systems*. New York, NY, USA: ACM Press, 1997, pp. 35–39.

[19] C. Eldershaw and M. Hegland, "Cluster analysis using triangulation," in *Computational Techniques and Applications*, 1997.

[20] P. Lévy, *Théorie de l'Addition des Variables Aléatoires*. Gauthier-Villiers, Paris, 1937.

[21] M. Gutowski, "Lévy Flights as an Underlying Mechanism for Global Optimization Algorithms," *ArXiv Mathematical Physics e-prints*, June 2001.

[22] M. Steiner and E. W. Biersack, "DDC: A Dynamic and Distributed Clustering Algorithm for Networked Virtual Environments based on P2P networks," Institut Eurécom, Tech. Rep., 2006. [Online]. Available: http://www.eurecom.fr/∼steiner/clustering.pdf

[23] A. Montresor, M. Jelasity, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems*, vol. 23, no. 3, pp. 219–252, Aug. 2005.