A fully distributed peer to peer structure based on 3D Delaunay Triangulation

Moritz Steiner and Ernst Biersack

Institut Eurecom, 2229, route des Crêtes, 06904 Sophia-Antipolis, France
{moritz.steiner,ernst.biersack}@eurecom.fr

This paper proposes the 3D Delaunay Triangulation ($\mathcal{DT}$) as a promising solution for constructing scalable p2p networks. The key idea is to maintain for each node a $\mathcal{DT}$ of the neighbour nodes. While demonstrating scalability in a real system is not practical for the current work, we demonstrate the scalability of the 3D $\mathcal{DT}$ using simulation. The results obtained indicate that there are upper bounds on the time needed to join and on the average number of neighbours maintained by a peer. Therefore, the amount of bandwidth and processing requirement for each node is bound, independent of the total number of nodes in the system.

Keywords: p2p, overlay network, graph, triangulation, delaunay

1 Introduction

A peer to peer (p2p) system is one in which autonomous peers have symmetric roles. P2p systems are constructed by connecting various computers (nodes) in a mesh-like fashion, thus forming a virtual network on top of the physical Internet. The term overlay network is thus often used to describe p2p systems. The whole system is only operable, and the peers can only benefit, if the peers that depend on each other in forwarding information and sharing computer resources. This is why issues of scale and redundancy become much more important than in traditional systems.
This paper proposes a fully-distributed p2p architecture, which attempts to solve the scalability problem based on the mathematical construct of the Delaunay Triangulation in 3D. The main contribution of the paper is to propose a resource efficient solution, which requires no server at all, not even for joining.
Section 2 deals with the partitioning of the virtual world; the geometrical construct Delaunay Triangulation is discussed. Section 3 describes the data structure and some of the algorithms we developed, as well as the main problems of the communication protocol.

2 Partitioning the 3D virtual world with the Delaunay Triangulation

2.1 The definition

The Delaunay Triangulation — introduced by Boris Delaunay in 1934 [Del34] — decomposes the convex hull of a point set S into unique cells (triangles in R^2 and tetrahedra in R^3). There are only two conditions building the $\mathcal{DT}$:

Definition 2.1.1 *Let S be a point set of n points in R^2. A $\mathcal{DT}(S)$ in 2D is a triangulation of S where no point d lies inside the circumcircle $\mathcal{C}_{abc}$ of any triangle $\mathcal{T}_{abc}$.*
Let S be a point set of n points in R^3. A $\mathcal{DT}(S)$ in 3D is a triangulation of S where no point e lies inside the circumsphere $\mathcal{C}_{abcd}$ of any tetrahedra $\mathcal{T}_{abcd}$.

The circumsphere is the sphere defined by the four vertices *abcd* of a tetrahedron $\mathcal{T}_{abcd}$. This definition implies that the tetrahedra are not flat, otherwise their circumscribing sphere is not defined.

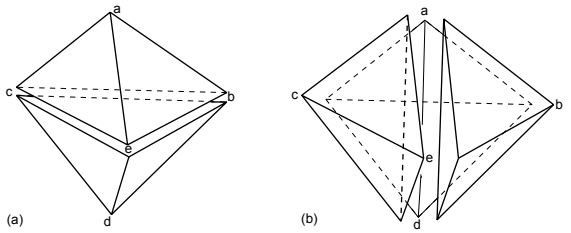

Fig. 1: A hexahedra can be separated into two or three tetrahedra [AK00].

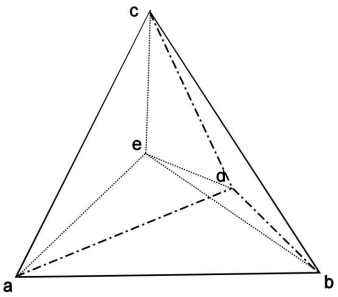

Fig. 2: Four new tetrahedra resulting from the split of $\mathcal{T}_{abcd}$ with e.

2.2 The Delaunay Triangulation construction

The basic component of most algorithms for the construction of the $\mathcal{DT}$ is the *Delaunay diagonal flip*. After a finite number of Delaunay diagonal flips out of any triangulation, the most compact one can be constructed: the $\mathcal{DT}$. For building a p2p infrastructure using $\mathcal{DT}$, a dynamic algorithm is needed. Therefore, in this section, we will focus on an *incremental algorithm*. In the first step the description is only done for the 2D case, then the main problem of switching to 3D is presented.
The addition of a point to the $\mathcal{DT}$ is equal to the problem of computing $\mathcal{DT}_i = \mathcal{DT}(\{p_1,\ldots,p_i\})$ from $\mathcal{DT}_{i-1}$ by inserting p_i. Let $\mathcal{T}_{abc}$ be one of the triangles of $\mathcal{DT}_{i-1}$ whose circumcircle $\mathcal{C}_{abc}$ contains the new point p_i and therefore are *in conflict* with p_i. It is not longer a Delaunay triangle, according to definition 2.1.1. Let $\overline{bc}$ be the edge that lies inside the quadrilateral $abcp_i$. A *Delaunay diagonal flip* has to be done to replace $\mathcal{T}_{abc}$ and $\mathcal{T}_{bcp_i}$ by $\mathcal{T}_{abp_i}$ and $\mathcal{T}_{acp_i}$. The newly created triangles must be tested as well, with the same process until there are no more triangles in conflict with any points [AK00].
In case no triangle is in conflict with p_i, the convex hull of all points from $S = \{p_1,\ldots,p_{i-1}\}$ needs to be enlarged. First p_i is connected to all points from S which it can *see*, that means to which straight lines can be drawn without crossing any edge from $\mathcal{DT}_{i-1}$. All created edges are Delaunay edges to $S \cup p_i$. The opposite edges to p_i must be checked and flipped if necessary.
Liebeherr has used the 2D $\mathcal{DT}$ as structure for a p2p network called Hypercast [LN02]. Hypercast has been tested with up to 10,000 nodes running on up to 100 computers.
In 2D a quadrilateral can only be divided into two triangles, whereas the hexahedra in figure 1 can be separated into two tetrahedra $\mathcal{T}_{abce}$ and $\mathcal{T}_{bcde}$ (Figure 1 a) or into three tetrahedra $\mathcal{T}_{acde}$, $\mathcal{T}_{abde}$ and $\mathcal{T}_{abcd}$ (Figure 1 b). The principal problem in 3D is that there are some sets of points that allow different triangulations, depending on the insertion order of the nodes, therefore the 3D $\mathcal{DT}$ is not unique.
The limited space of this paper does not allow to describe the removal of a point from the $\mathcal{DT}$.

3 A distributed algorithm for the Delaunay Triangulation in 3D

Currently no algorithm exists to compute the $\mathcal{DT}$ in a distributed way. By distributed we mean that each point of the triangulation corresponds to one computer, which is aware of its direct neighbours only. The main problem is the consistency: All nodes are aware of their direct neighbours only and not of all the nodes of the triangulation. Thus, it is very difficult to keep the views of all the nodes consistent.
This section describes the data structure, followed by a short presentation of the main method, which adds a point (join from a peer point of view). Finally, some test and validation methods are presented and the main problems of the communication protocol are discussed.

3.1 The data structure and the basic methods

We do not use the *Quad-Edge* structure developed by Guibas and Stolfi [GS85] or the simpler structure based on a double connected edge list (DCEL) [BKOS00], but one that allows for easier navigation, even though some information is redundant. There are two main objects in this structure: `Node` and `Tetrahedron`. A `Tetrahedron` is the main object for storing the triangulation. It is composed of its four vertices, stored as `Node`s and its circumsphere stored as `Sphere` that consists of a `Point` for its cen-

ter and its radius. Furthermore it stores its four facets as `Planes` and its four neighbour `Tetrahedron`. A `Node` stores its position in a `Point` and maintains a list of its neighbour `Nodes` and a list of the tetrahedra (`Tetrahedron`) where it belongs to. These references allow to navigate in the triangulation. All the other objects have only a supporting function, e.g. the `Vectors` and `Lines`, which are basically needed for the calculation of the center of the circumsphere of a `Tetrahedron` and to calculate the representation of a `Plane` by its normal vector starting from three `Points` or `Nodes`.

3.2 Join the Delaunay Triangulation

The node nj wishing to join needs to know one node belonging to the $\mathcal{DT}$ that executes the join procedure for nj. The nearest node *nn* to the desired location of nj is searched by recursively traveling through the $\mathcal{DT}$. For the following pseudo code we assume that the $\mathcal{DT}$ has at least four nodes and the node *nn* has been found.

```
tetra ← neighbours(nn,2) /* all tetrahedra of nn and their respective neighbour tetrahedra */
liesInside ← false /* true if nj lies at least inside one sphere */
for i ← 0 to tetra.length do
    if nj lies inside sphere of tetra[i] then
        newtetras ← null /* variable for the min. 2 and max. 4 tetrahedra resulting from the split */
        for j ← 0 to 3 do
            newtetras[j] ← new
            Tetrahedron(tetra[i](j%4),tetra[i]((j+1)%4),tetra[i](j+2)%4),nn)
            /* tetra[i](j) returns the j.th point of tetra[i] */
            if not nn lies inside tetra[i] then
                if tetra[i]((j+3)%4) lies inside sphere of tetra[i] then newtetras[j] ← null
            end
        end
        Update the neighbour relationship between the 5 nodes (the four nodes of tetra[i] and nn)
        Update the list of tetrahedra of the 5 nodes
        Update the tetrahedra neighbourship relations between the newtetras
        Update the convex hull (in case one facet of tetra[i] was on it)
        liesInside ← true
    end
end
if not liesInside then enlarge the convex hull of DT with nj
```

If the joining node lies inside one tetrahedron (not only inside its sphere) the split creates four new tetrahedra (Figure 2). If the new node lies outside the tetrahedra (but inside its sphere) only two or three new tetrahedra are created (see section 2.2 and figure 1).

3.3 Tests and validation

During the execution of the code, whenever a change occurs, all nodes and tetrahedra affected are checked for compliance with definition 2.1.1. Moreover all neighbour relations between the tetrahedra are checked: Two tetrahedra having a common plane must cross-reference each another. Several different scenarios have been tested to validate the algorithm: Randomly distributed nodes; Nodes randomly distributed on the surface of a sphere; Points distributed in smaller and bigger clusters, computed with the Lévy Flight method [Lév37].

As there is no general view of all nodes and all tetrahedra, all nodes write the tetrahedra they know in a common file to gain a global view. This file is then compared with the output of the CGAL $\mathcal{DT}$ algorithm [CGA].

3.4 The communication protocol

The protocol itself is not discussed in this paper, but two major problems are described: the consistence of the triangulation during and after the concurrent execution of more than one task and the crash of a peer and the involved reparation of the triangulation.

It is crucial for the coherence of the virtual world, to locally accept only one alteration a time. For example

two peers may join or leave the network at the same time only if they do not alter the same nodes or tetrahedra. Otherwise the two (ore more) peers would create or destroy tetrahedra or peer relationships without knowing about the actions of the other one, which would automatically lead to an inconsistent triangulation. Therefore, all peers involved in a join or leave procedure are locked. Peers can still perform actions, e.g. send messages, but they cannot alter the triangulation. This approach is comparable to the lock mechanisms implemented in database systems to allow transactions conform with the ACID paradigm. A peer that wants to join tries to lock all nodes involved. If it doesn't succeed, it unlocks them and retries after a certain time, to avoid deadlocks.
In the case of a crash of one of the peers, the other peers must detect the crash. Hence heartbeat messages are sent to the neighbours. If one peer has not received the heartbeat message from one of its neighbours, it takes the leadership and locks all the neighbours of the crashed peer and executes the *leaving procedure* for the crashed peer.

3.5 Complexity

The task of building the $\mathcal{DT}$ in a static way (all the nodes are known at the beginning of the construction) has a lower bound of $\Omega(n \log n)$ in 2D and $\Omega(n^2)$ in 3D, where n is the number of nodes [YB98].
In our case not the overall time complexity is important, but the complexity for joining one node to the $\mathcal{DT}$. As the simulation results have shown, the number of neighbours of a node is bound, as well as the number of tetrahedra destroyed during the join. These two numbers do not grow with the total number of nodes in the $\mathcal{DT}$. If the nodes are randomly distributed, the average number of neighbour nodes is 15, the maximum is 35. The average number of tetrahedra destroyed during the join of a node is 15, the maximum is 28. The time complexity of the join procedure does not depend on the total number of nodes in the $\mathcal{DT}$, but it only depends on the number of neighbour nodes. This can be seen very clearly, because the main loop of the pseudo code in section 3.2 is executed for all neighbour tetrahedra of degree two, this number directly depends on the number of neighbour nodes. Since the number of neighbour nodes is nearly constant, and does especially not grow with the number of nodes in the $\mathcal{DT}$, the complexity of the join procedure is constant as well.

4 Conclusion

This paper presents a solution to the scalability problem in p2p networks, by showing a way to partition the 3D virtual space with the help of the Delaunay Triangulation. More work is needed to improve the performance of the presented algorithm and to allow efficient movements of the nodes without using the join and leave methods.

References

[AK00] F. Aurenhammer and R. Klein. *Handbook of Computational Geometry*, chapter 18, pages 201–290. Elsevier Science Publishers, 2000.

[BKOS00] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[CGA] CGAL. The cgal website. http://www.cgal.org.

[Del34] B. Delaunay. Sur la sphère vide. A la mémoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 7:793–800, 1934.

[GS85] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[LN02] J. Liebeherr and M. Nahas. Application-layer multicasting with Delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications*, 20(8):1472–1488, October 2002.

[Lév37] P. Lévy. *Théorie de l'Addition des Variables Aléatories*. Gauthier-Villars, Paris, 1937.

[YB98] M. Yvinec and J.-D. Boissonnat. *Algorithmic Geometry*. Cambridge University Press, 1998.