# A New, Fully Decentralized Grid Generation Method

Panchalee Sukjit* and Daniel Berg*

**Abstract**

In order to manage search-, and routing-functionality in structured P2P-networks, a lot of algorithms like CAN, Chord, Tapestry, Pastry, and more have been developed. In this paper we introduce a distributed algorithm which uses a set of simple rules to build and maintain a complete, parallel-growing and although contradiction-free grid-structure just by the use of local knowledge of each node. This leads to a large-scale growth of a decentralized network based on a global n-dimensional Cartesian coordinate-system built without any global instance.

**Keyword:** P2P, pattern formation, grid, fault-tolerance, scalability

## 1. Introduction

Decentralized approaches for organizing networks have big advantages in managing and maintaining the high dynamics that occur especially in community based networks [1, 3, 4, 8]. Three main-operations have to be implemented for those decentralized networks: The Join-operation, that integrates new peers into an existing network, the Leave-operation,which keeps the network in a consistent state, when a peer leaves the network, and finally the Lookup-operation, that locates desired resources in the network. Unstructured P2P networks use broadcast-mechanisms to perform resource lookups and routing [2]. Adding and removing nodes can be performed with low costs, since there is no special structure that has to be kept consistent. But, on the other hand, due to the absence of an efficient searchable structure, performance in finding resources is poor [10].

Algorithms for structured P2P networks, like Chord, Tapestry, Pastry, or CAN have much more complexity in Join- and Leave- operations, because they maintain certain structures, like rings, trees, or meshes, that make Lookup-operations more efficient and therefore increase their performance [9, 10].

Networks and topologies based on those regular grids have significant advantages in managing computing resources [1, 3]. Efficient routing and fault-tolerant routing algorithms have been developed for these topologies [5]. However, whenever those grid-structures are built, they are built staticly or under centralized control.

In this paper we introduce a distributed locally working algorithm which can set up a complete and regular grid in a decentralized, dynamic environment.

Our approach is to simplify the structure formation and managing process by letting a network grow as a regular grid1).

In section 2.1 we shortly describe the problems and requirements that arise when building a grid without any global view to the grid itself and with limited knowledge about the existing neighbourship. Then, in section B the growth algorithm is introduced. Section 2.2 gives a formal definition of the algorithm. Section 3 discusses the simulation setup, performance evaluation, fault tolerance and possible optimisations. In Section 4, finally, we give a short outlook to further investigations.

## 2. A Decentralized Grid-Building Algorithm.

### 2.1 Requirements

Grids are easy to set up under centralized control and if the number of involved machines is static [6, 7]. Setting up a grid only with local knowledge in opposite is not that simple, if we assume that a node within a grid only knows about its four direct neighbours. Commonly, following requirements are the base for our approach:

• The structure must be built fast and with noncomplex algorithms.

• The structure does not appear in a previously fixed coordinate or cell space system

• It must be easy to repair in case that any changes in the network appear.

• The algorithm is running locally on each peer and only can use the information available on this peer and eventually on its neighborhood peers (since global information is not available).

• The generated overhead shall be minimal and the achieved efficiency maximal.

Uncontrolled joining of new nodes could result in holes within the grid (Fig. 1a). Another problem, which is implied by the existence of those holes is that a node cannot see the situation behind gaps that come up with holes.

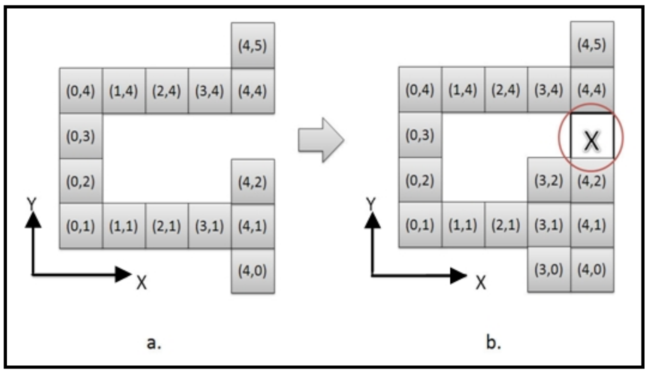

*Figure 1 : Uncontrolled growth process will lead to holes and to overlapping areas*

*Department of Communication Networks, Faculty of Mathematics and Information Fernuniversity in Hagen

This situation is shown in Fig. 1b at the position denoted with *X*: Assume, that Node (4, 2) accepts *X* to be its new neighbour. Since every node just knows about its direct neighbours, the node at position (4, 4) does not know about the existence of *X*. Node (4, 4) still assumes that position (4, 3) is free and therefore is ready to accept a new node at this position. That would lead to the contradiction situation that two nodes could reside at the same position (4, 3).

The rules on which the growth process is based must ensure that holes or contradiction situations cannot occur. Furthermore it is necessary to fix holes that arose from peer-failures within the grid.

**2.2 The growth process**

To describe the growing process we assume to have a virtual cartesian 2-D coordinate-system and a root-node in the origin of this coordinate-system, i.e. it has the coordinates (0, 0). To keep things easy we focus on the first quadrant. The discussed algorithm and its rules are symmetric, and therefore work analogous in the other three quadrants. The term 'a node *N* grows in a certain direction' means, that N can declare a new node *N*' to be its neighbour in that direction.

We call nodes that reside within the quadrant ($x > 0$ and $y > 0$) inner nodes. Nodes that reside on one of the axises ($x = 0$ OR $y = 0$) we call skeleton nodes.

An inner node *N* can grow in diagonal direction away from the origin, if and only if N already has a neighbour on its north-side as well as on its east-side (Fig. 3, 4b).

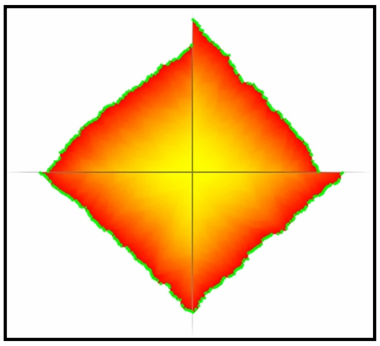

***Figure 2*** *: A simulation with 100, 000 nodes*

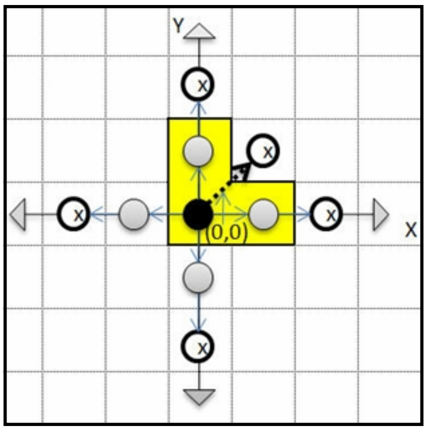

***Figure 3*** *: The growth process at an initial state. The position denoted with x show, to where the grid currently could grow*

This rule is also valid for the skeleton nodes. Additionally, skeleton rules can grow in the direction of the axis on which they reside (Fig. 3, 4).

Using these rules the quadrant will be filled up without producing any holes and without putting more than one node at the same position. The nodes themselves just use local knowledge, i.e. they just know their direct neighbours in north-, east-, south-, and west-direction. After a new node was added the neighbourship information can easily be updated just by informing the grown node's north-neighbour, that it has a new east-neighbour, respectively the grown node's east-neighbour, that it has a new north-neighbour.

The next section gives a mathematical description for this structure and its growth process, that fulfills the requirments given in section 2.1.

**2.3 The algorithm**

Let *Z* denotes the set of integers, and let *Q* be the infinite grid graph with vertex set $V(Q) = Z^2$ where two distinct vertices $(x, y)$ and $(x', y')$ are adjacent in *Q* if and only if

$$x = x' \wedge |y - y'| = 1 \;\text{ or }\; y = y' \wedge |x - x'| = 1$$

We call a sub-graph *G* of *Q* nice, if

$$\begin{aligned}
&(1)\ \textit{if}\ (x,y) \in V(G)\\
&\quad \textit{then}\ (x,0) \in V(G),\ (0,y) \in V(G)\\
&(2)\ \textit{if}\ x < x' < x'' \wedge (x,y) \in V(G) \wedge\\
&\quad (x'',y) \in V(G)\ \textit{then}\ (x',y) \in V(G)\\
&(3)\ \textit{if}\ y < y' < y'' \wedge (x,y) \in V(G) \wedge\\
&\quad (x,y'') \in V(G)\ \textit{then}\ (x,y'') \in V(G)
\end{aligned}$$

It is easy to see that every nice sub-graph of *Q* is connected and induced.

A directed graph *T* with vertex set $Z^2$ is defined as follows. The edge set *F* of *T* is $F = F_1 \cup F_2 \cup F_3$ with

$$\begin{aligned}
F_1 &= \{((x,y)\ (x',y')) \in Z^2 \mid\\
&\quad x, x', y, y' \geq 0 \wedge |x'| - |x| = |y'| - |y| = 1\}\\
F_2 &= \{((x,y)\ (x',y')) \in Z^2 \mid\\
&\quad x = x' = 0 \wedge y, y' \geq 0 \wedge |y'| - |y| = 1\}\\
F_3 &= \{((x,y)\ (x',y')) \in Z^2 \mid\\
&\quad y = y' = 0 \wedge x, x' \geq 0 \wedge |x'| - |x| = 1\}
\end{aligned}$$

If $((x, y), (x', y'))$ is an edge in T we say that $(x', y')$ is a descendant of $(x, y)$ and $(x, y)$ is the ancestor of $(x', y')$. The following claims are immediate consequences of the definitions given above:

Claim 1:

The vertex (0, 0) has no ancestor. All other vertices have precisely one ancestor.

Claim 2:

There is a directed path from (0, 0) to any other vertex $(x, y)$ in *T*.

Let *G* be a nice sub-graph of *Q*. The following rules define how $G$ can grow:

R1:

$if\ (x,0) \in V(G) \wedge (x+1,0) \notin V(G)$ then add a new vertex $(x+1, 0)$ and a new edge from $(x, 0)$ to $(x+1, 0)$

R2:

$if\ (0,y) \in V(G) \wedge (0,y+1) \notin V(G)$ then add a new vertex $(0, y+1)$ and a new edge from $(0, y)$ to $(0, y+1)$

R3:

$$if\ (x,y) \in V(G) \wedge (x,y+1) \in V(G) \wedge$$
$$(x+1,y) \in V(G) \wedge (x+1,y+1) \notin V(G)$$

then add new vertex (x+1, y+1) and the edges from $(x, y+1)$ to $(x+1, y+1)$ and from $(x+1, y)$ to $(x+1, y+1)$.

R1 and R2 describe the growth process of skeleton nodes along the axises, while R3 describes, how inner nodes grow.

Claim 3:

Let $G$ be a nice sub-graph of $Q$, and let $G'$ be a graph from $G$ by simultaneously adding vertices following the rules R1, R2, and/or R3 then $G'$ is a nice sub-graph of $G$. In the next section we discuss the algorithm's performance.

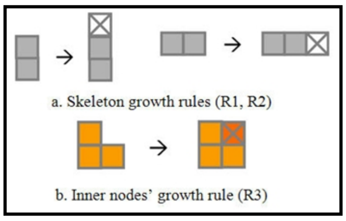
a. Skeleton growth rules (R1, R2)
b. Inner nodes' growth rule (R3)

***Figure 4 :** The visualized growth rules R1, R2 (a), and R3 (b)*

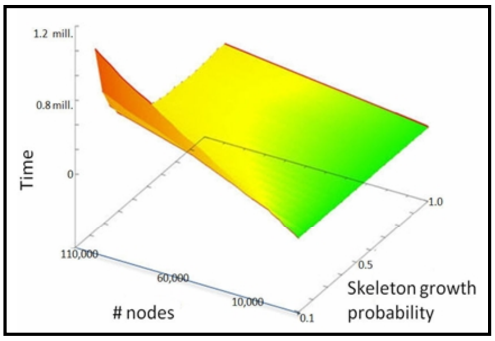

***Figure 5 :** Scalable growth characteristics depending on the skeleton's growth rate*

**3. Performance Evaluation and Fault Tolerance**

**3.1 Simulation Setup**

To observe, how this algorithm performs under real-time conditions, P2PNetSim [11] – a distributed network simulator was utilities. P2PNetSim allows large scaled network simulations,and analysis on cluster computers. Peer-Behaviour of skeleton- and inner nodes can be implemented in Java and then be distributed over the nodes of the simulated network. At simulation start up the peers are interconnected small-world-like in order to simulate the typical physical structure of computers interconnected in the Internet. On top of this structure an overlay-network is built using the grid-algorithm. This allows it to compare metrics of the physical network with those of the overlay-grid. Furthermore, the efficiency of optimisations, like exchanging nodes in the grid, in order to adapt spatial relationships in the overlay to the physical structure (see section "optimisations"), can be observed and analyzed. Fig. 2 shows the output of a simulation, that generated a grid with apprx. 100,000 nodes.

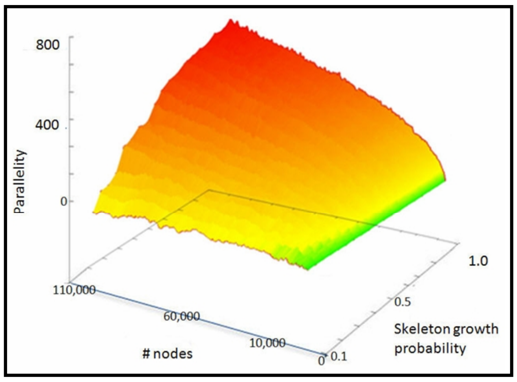

***Figure 6 :** Dependency of growth-parallelity from the skeleton's growth-rate*

**3.2 Performance Evaluation**

Based on the described algorithm the network grows at diagonal locations at the grid's border. Border sides which are parallel to one of the axises will not grow until new nodes from a axis will cause new situations to which the growth-rule R3 can be applied.

The more the network structure converges to a rectangle the less it will be able to grow.

On the other hand, if the skeleton nodes grow too fast in comparison to the inner nodes, the network tends to get degenerated at its axises, which grow far out of the grid's dense area.

Fig. 5 shows how the inner node/skeleton node growth ratio influences the grid's overall ability to grow. The slower the skeleton grows, the more time the whole network needs to grow. Fig. 6 shows the number of free nodes (parallelity) depending on the skeleton-growth-probability. Free nodes are those that are able to accept new nodes. The parallelity increases with increasing skeleton-growth-probability.

**3.3 Fault Tolerance**

Assuming that any node in the grid could fail at any time, there must be mechanisms to fix holes reliably. One possibility is, that whenever a node detects a missing neighbour it may initiate a process which moves a node from the grid's border to the defective position. This moving process, of course, must still follow the rules R1-R3. This might lead into situations where holes can't be refilled until other nodes fix their neighbourships. In this case the moving process could be delegated to another node that resides at the border of the hole, and that is able to apply one of the rules R1,R2 or R3 to accept a new neighbour.

Another approach to handle leaving nodes is implied by claim 1. Every node (except the root node) has precisely one ancestor. In the first quadrant, for any node D at position $(x, y)\ x, y > 0$, its ancestor A resides at position $(x - 1, y - 1)$.

For skeleton nodes on the $x$ - axis ($x > 0$, $y = 0$) the ancestors can be found at positions ($x$ - 1, 0), and ancestors of skeleton nodes on the $y$-axis at (0, $y > 0$) sit at positions (0, $y$ - 1).

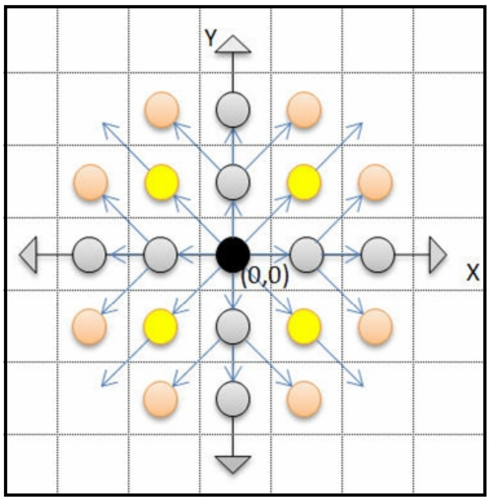

***Figure** 7 : Relationships (diagonal lines, and vertical & horizontal lines on the axises) form a tree*

These relationships span up a tree within the grid, where the root-node at position (0, 0) is mapped to the tree's root, while all other peers model the tree's inner nodes (Fig. 7). The peers at the top of the tree (the tree's leafs) are those that are able to grow, i.e. to accept joining nodes to extend the grid. If any node failure occurs, the tree explicitly and uniquely denotes the next ancestor as the node, that is responsible for handling this failure. This node might incur the missing node's routing capabilities and therefore adheres the grid virtually. Another possibility would be, that the responsible node reorganizes the grid in that way, that it asks a node at the grid's border for changing its position in order to fix the hole. The exchange of nodes could be performed quickly, since this just requires a node's coordinates to be changed.

In the next chapter, we suggest some optimisations, that could decrease the complexity for finding nodes at the grid's border.

### 3.4 Optimizations

Following optimisations are suggested to improve performance and fault-tolerance:

**• Finding alternatives to skeleton nodes**

The growth-behaviour of the skeleton nodes decides, how the grid grows. Special tuning is required to control the skeleton growth, and therefore the growth of the whole grid. Alternatives should be investigated to make the growth-process more efficient and independent from the skeleton-concept.

**• Higher dimensions**

Instead of limiting the structure to two dimensions, the grid could grow in three or more dimensions.

**• Grid hierarchies**

Setting up hierarchies of grids with lower granularity, to improve lookup- and join-operation performance.

**• Flatpacks**

Flatpack of planes with increasing and then decreasing granularity. The granularity depends on communication bandwidth available on each node.

**• Node exchange**

pairwise node exchange, in order to reflect physical neighbourships within the logical grid. Beside the physical neighbourship, any other criteria could be used for this kind of reorganisation, for example, to set up a content- or resource-based neighbourship.

**• Jumper nodes**

Permanently available nodes with high bandwidth may act as jumper stations. Jumpers are connected using UTH network structures [4]. They can be used for speeding up far distance connections, when they are found on a processed routing path [12].

**• Pheromone paths to jumpers**

Ant-like pheromones are used to mark paths to jumper nodes. When detected by a message on its routing path, the pheromone-level influences the routing path and leads the message to a quick jumper node.

**• Self organizing path systems**

Finding different paths for difference directions. Avoids message-collisions. Number of detected collisions can be used to decide when, and where to set up an alternative route.

## 4. Conclusion and Outlook

In this article we described a distributed algorithm that sets up an anarchic growing, complete and regular grid. It can be used as an overlay-network. Though the algorithm just works with local knowledge, the upcoming grid is hole- and contradiction-free (i.e. no overlapping areas occur). This logical grid can be used for routing, and resource lookups. It is flexible in that way, that nodes could change their positions, for example to reflect any thematic similarities in nodes' relationships to each other.

The scalability of the grid depends on the ratio of the number of grow-able nodes at the border of the grid and the number of inner nodes. This ratio, again, depends on the growing behaviour of the skeleton-nodes. Further investigations must be done, to find the ideal growing behaviour of those skeleton nodes, or even to get rid of the skeleton nodes at all by replacing them by a better concept.

Another task of further investigations, is to create grids with higher dimensions. This would raise the scalability on the one hand, and would reduce routing- and lookup-complexity on the other hand.

## 5. References

[1] D. Tavangarian and G. Hipper, "A New Architecture for Efficient Parallel Computing in Workstation Clusters: Concepts and Experiences", *Proc. The High-Performance 1998, Tentner, Boston, Massachusetts,* Boston, Massachusetts, pp.271-276, April. 1998.

[2] J. Widmer, C. Fragouli and J.-Y.Le Boudec, "Low-complexity energy-efficient broadcasting in wireless ad hoc networks using network coding", *Proc. First Workshop, Netw. Coding, Theory, and Appl. (NetCod 2005),* Riva del Garda Italy, April. 2005.

[3] S. Ratnasamy, P. Francis, M. Handley, R.M. Karp and

S. Shenker, "A Scalable Content Addressable Network", *Proc. ACM SIGCOMM*, 2001.

[4] D. Berg, H. Coltzau, P. Sukjit, H. Unger and J. Nicolaysen, "Passive RFID tag Processing using a P2P architecture". *Proc. Malaysian Software Engineering Conference,* 2007.

[5] A. Mello, L. Copello Ost, O. Gehm Moraes, N. Laert and V. Calazans, "Evaluation of Routing Algorithms on Mesh Based NoCs", *Proc. Technical Report Series* No.040, May. 2004.

[6] T.M. Riaz, R.H. Nielsen, J.M. Pedersen and O.B. Madsen, "On Line Segment Length and Mapping 4-Regular-Grid Structures in Network Infrastructures", *Proc. of 5th International Symposium, CSNDSP 19-21,* July 2006, Patras Greece, pp. 435-439, 2006.

[7] Fritzke: "Growing Grid - a self-organizing network with constant neighbourhood range and adaptation strength", *Proc. In Springer Netherlands on Neural* Processing Letters, Vol. 2, April. 2006.

[8] G. Sakarian, H. Unger and U. Lechner, "About the value of Virtual Communities in P2P networks." *Proc. ISSADS 2004, Guadalajara, Mexico, Lecture Notes in Computer Science (LNCS) 3061,* Guadalajara Mexico, 2004.

[9] I. Stoica, R. Morris, R. David, Karger, M. Frans Kaashoek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications". *In Sigcomm'01, Proc. of the 2001 conference on Applications technologies architectures and protocols for computer communications,* ACM Press, pp. 149-160, 2001.

[10] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim, "A Survey and Coparism of Peer-to-Peer Overlay Network Shemes", *IEEE Communications Survey and Tutorial,* March. 2004.

[11] H. Coltzau, "Specification and Implementation of a Simulation Environment for Large P2P-Systems", *Diploma, University Of Rostock,* 2006.

[12] D. Easley and J. Kleinberg, "The Small-World Phenomenon", *In Networks: Spring 2007,* 2007.