IADIS International Journal on Computer Science and Information Systems
Vol. 4, No. 3, pp. 99-110
ISSN: 1646-3692

VORODSPT: A VORONOI BASED OVERLAY FOR SPATIAL OBJECTS

Dominic Heutelbeck *University of Hagen - D-58084 Hagen - Germany.*
dominic.heutelbeck@fernuni-hagen.de

Christina Sergel *Serkom - D-85229 Markt Indersdorf - Germany.*
cs@wiederda.de

Matthias Hemmje *University of Hagen - D-58084 Hagen - Germany.*
matthias.hemmje@fernuni-hagen.de

ABSTRACT

Distributed space partitioning trees (DPSTs) (Heutelbeck and Hemmje, 2006) describe a distributed spatial index for storage and access of the so-called location knowledge. Location knowledge is acquired in a complex distributed process through positioning, tracking, and program logic to support location-based services which are based upon knowledge about the physical location, shape, and size of real and virtual entities, such as persons, vehicles, cities or other location-based services. Location knowledge is needed at diverse applications in geographical information systems (GIS), virtual collaboration systems and mobility management. The previous implementation of a DSPT, RectNet (Heutelbeck and Hemmje, 2006), was built on an adaptive binary tree shaped network topology. The use of a recursive space partitioning can be useful for load balancing. However, previous implementation requires complex operations to restructure the network. These operations cause significant bursts in network traffic. In this paper we present VoroDSPT, a new DSPT implementation, by using the geometric structure Voronoi Diagram as a topology that provides both greedy-routing for supporting efficient spatial queries and information sharing, providing the base for implementing future efficient heuristics of load balancing.

KEYWORDS

Spatial indexes, Peer-to-Peer, Overlay, Location-based services, Voronoi-Diagram, DSPT.

1. INTRODUCTION

A central problem for the distributed indexing and searching of location knowledge is the non-uniform distribution of keys in the search space. Following the notation of DSPTs (Heutelbeck

and Hemmje, 2006), located objects are represented and stored as (l_o, o)-tuples, where l_o denotes the location of an object o. The location is an arbitrary measurable subset of a two-dimensional search-space, often represented by geometric primitives, such as circles or multi-polygons. A core search operation defined by DSPTs is "lookup all tuples whose locations intersect the set Q". This implies that pair wise relations, such as the intersection between the locations of stored tuples and a query set Q, have to be preserved by the index. Peer-to-peer indexes, such as distributed hash tables (DHTs), e.g., Chord (Stoica et al., 2001), are not feasible to solve this problem, as by hashing the keys the set nature of the keys and the associated relations are lost. In addition the usage of space-filling curves to map the spatial keys is also difficult in case of DSPTs. Here, not only points but also sets are considered as key values. We assume, that the more local neighborhood relation between peers in our approach are also more suitable for the implementation of DSPT systems.

In DHTs, the hashing of the keys is a central concept in order to achieve effective load-balancing between the individual peers. In practice, it is easily observable that given a number of spatial objects, these objects are usually not uniformly distributed across the search space, e.g., the surface of the earth. Hence, a distributed spatial index uses other means for handling the non-uniform distribution of data in the index.

In this paper we present results of our ongoing research in designing DSPTs. The implementation presented in this paper realizes the base for what we call *smooth-load-balancing*, by providing a new way of partitioning the search-space via Voronoi Cells. Future extensions will be able to implement different heuristics for load-balancing. Those heuristics smoothing out the network traffic, based on adaption and the movement of the logical position of the generator points of a Voronoi diagram. In this paper, we will describe the peer-to-peer (P2P) design of our implementation VoroDSPT and we will provide some initial figures, indicating the performance of our system.

The rest of this paper is organized as follows: Section 2 describes different related efforts and the state-of-the-art of DSPT indexing. In Section 3 we present the performance of VoroDSPT. Section 4 describes experimental results. Section 5 states our concluding remarks.

2. RELATED WORK

RectNet (Heutelbeck and Hemmje, 2006) is a DSPT implementation which allows publishing, updating, and searching for located objects in a overlay network. RectNet is based on a rectangular tessellation of the plane similar to CAN (Ratnasamy et al., 2001) and uses the concept of the cluster based routing protocol that relies on the binary space partitioning trees. In this protocol nodes responsible for a subsection of the search space, i.e., clusters, are called *clusterheads* which are responsible for maintaining the structural integrity of the network according to the tessellation. As the number of activities inside individual clusters increases, the communication overhead increases as well and the cluster has to be split in order to avoid that the clusterhead becomes a bottleneck. Another problem is that during the time of reconstruction the routing to the split clusters is interrupted. As RectNet adapts to changing load distributions by restructuring the binary tree structure describing the tessellation, the restructuring protocol is very complex, creating traffic bursts by redistributing a lot of data in a short period of time. The protocol does not offer more fine-grained means of adaption.

Distributed Hierarchic Spatial Index (DHSI) (Berleong, 2007) is a design system for a World Wide Space which provides the conceptual and technological framework for integrating and sharing context models. DHSI has the primary goal of achieving a scalable distributed indexing for mobile objects which have a location and types. DHSI has two overlays: one for classes of types constructed from a binary tree structure and one for spatial requests constructed from a 2-dimensional index structure that represents geometric objects like CAN (Ratnasamy et al., 2001). Therefore, a DHSI-node has to choose between three different kinds of routing links depending on the weight of classes of types and location. This approach tries to balance the load by transferring virtual index-nodes from nodes under heavy loaded to nodes with more free resources by using a so called *splitfactor* for selecting the physical node under the highest load.

GeoPeer (Araújo et al., 2004) is a P2P-system with a greedy algorithm based on a Delaunay triangulation. It provides services to location-aware applications by storing objects in distributed hash tables that are suitable for point data. The greedy algorithm is augmented by a Long Range Contact (LRC) mechanism for reducing path lengths in a two dimensional CAN or Delaunay triangulation. To get a direct LRC, every node which has to forward a message already equipped with a predefined number of hops has to generate these short cuts actively by sending additional messages to call on the source node to create a LRC. Although the use of LRC efficiently reduces average path lengths even in extremely unbalanced node distribution, the maintenance of LRC needs additional message traffic whereas the creation and maintenance of Long Distance Links (LDLs) in VoroDSPT is achieved with local exploitation of storage information.

3. VORODSPT

In VoroDSPT the insertion and retrieval of spatial objects (l_o, o)-tuples defined by DSPTs, is implemented as a service on top of a flat peer-to-peer overlay network, that enables the usage of concepts like neighbourhood, interior regions and boundary of sets and connectedness.. The topology of the overlay is based on a Voronoi Diagram to partition the Euclidean-space into 2-dimensional Voronoi regions. Those regions are used to assign responsibilities to individual peers. Each peer is assigned to a Voronoi region, and stores all (l_o, o)-tuples, where l_o intersects its Voronoi region.

Let S be a set of n points of the Euclidean-plane. A Voronoi Diagram (VD) of s distinct sites in the plane is a partition of the plane into s Voronoi regions. Each region is associated with one of the sites. The individual sites are also called the generator of their respective region. The Voronoi region associated with a site $X \in S$ consists of all the locations in the plane, that are closer to X than to any other point in S. Thus, given an arbitrary site P in the plane, one can determine which of the s generators of S is closest to P by determining which of the s regions contains P. A Voronoi region is a convex polygon (possibly unbounded) determined by bisectors of X and other sites and each region is the intersection of all such bisectors. Okabe et al (Okabe et al., 1992) gave a thorough review of concepts, applications and algorithms related to Voronoi Diagrams. It is well-known that the Voronoi Diagram for s generators in the plane can be constructed in $O(n \log n)$ time (Okabe et al., 1992). In VoroDSPT, Steven's Fortune algorithm (Fortune, 1986) was used for the construction of VD.

The VD containing all the sites of the network, the so-called *global VD*, is never constructed, as it is sufficient to maintain local views of the network at each peer and there is also no global observer who would be able to construct such a diagram. However, we constructed global VDs in experimental settings in order to verify the overall integrity of the network. A peer *p* represents an instance of an active participant in the Voronoi based overlay. Each peer $\boldsymbol{p}$ is assigned a unique node $\boldsymbol{n_p}$ containing a location, a logical, and a physical address. The logical address relates to the node's identifier for the VD-overlay and is computed using a hash function like SHA-1. The location $\boldsymbol{l_p}$, which indicates peer's VD-generator $\boldsymbol{s_p}$, is defined as a coordinate $\boldsymbol{coord(A) = (x_A, y_A)}$. To make it improbable that two nodes get the same coordination, there is an order on the node's location: $\boldsymbol{coord(A)} \leq \boldsymbol{coord(B)}$, if $\boldsymbol{x_A \leq x_B}$ and $\boldsymbol{y_A \leq y_B}$ or $\boldsymbol{y_A = y_B}$. The physical address denotes peer's IP-address with its port number on the underlying network. The maintenance of peer's own *local VD* depends only on its location $\boldsymbol{l_p}$, its local view of spatial environment and on the locations of its known neighbors. The neighborhood and the local view of spatial environment are defined by computing peer's local VD. As the connections within the overlay are implemented using UDP, each peer has to manage the individual connections to its neighbors. This is necessary to be able to detect and to recover from inconsistencies like nodes that disappear without informing their neighbors. This approach is implemented by sending frequently UDP keep-alive messages, which are flexible in regard to their dynamic adjustment of timeout intervals. For reducing the keep-alive overhead a VoroDSPT peer sends only alive request messages to its neighbors and regards their alive request messages as reply messages.

Given that a VoroDSPT peer has six neighbors averaged the costs for the keep-alive process remain constant. The routing is performed by a purely local decision according to the state of its local Voronoi Diagram. A global VD cannot provide more exploitable information for the greedy routing to a peer, as it can be found in the local VD.

This is true as long as the local view is equal to the global possible local view. When a neighbor of a peer leaves the network, or a new neighbor is inserted into the network, there are short periods of time where the local view of the neighborhood and peer's boundary of connectedness may not be true.

During our work we built a test application maintaining a global VD and comparing the local view with the global view. Peers within the network always have the stated identity, whereas during times of change there can be differences. The availability of peers was validated by a test application counting the neighboring nodes from a local view of a single peer *X* and from a global view of peer *X* at a given time. The simulation indicated that shortly after a node failure the local view of neighborhood corresponds to the global view of neighborhood, which is the optimal state. The protocol eliminates the differences, while the peer learns about its changed neighborhood.

A VoroDSPT peer self-organizes into a planar VD. Peers may join and leave dynamically. Each peer constructs and controls its own local VD that determines its state of neighborhood and its Voronoi region, which is set as the peer's governing area for storing spatial objects. The entrance and exit of a peer changes the Voronoi regions. A network based on a VD has the following desirable characteristics:

- expected node degree *O(1)*
- greedy routing with a mean path length of *O(log n)*
- locality, which leads to a dynamic and distributed construction in O(n log n) time

Given an existing Voronoi Diagram, the entrance of a new peer or the exit of an existing one affects only the region adjacent to the location of the peer. Therefore, every peer has to calculate its own local state of view of the global Voronoi Diagram separately and locally. The time and memory needed to compute a local Voronoi Diagram is *O(n log n)* where *n-1* is the number of neighbors the Voronoi region has.

The abstract data structure DSPT (Heutelbeck, 2005) is implemented as a service which efficiently provides operations for publishing a set of spatial objects and searching for spatial data. The service makes certain operations available, such as *publish(object)*, *remove(object)* and spatial queries which allow the use of geometrical data types. Spatial queries consider the spatial relationship between geometrical data, e.g. equality query, which returns all spatial objects whose location is identical to a query location, intersection query, which returns all spatial objects whose location intersects a query location, or inclusion query, that returns all spatial objects where a query location is a subset of the location. Thus, the DSPT service allows access to spatial objects having a static or variable form and position and existing only for a limited time.

3.1 Joining and Leaving

A dynamic network needs to deal with nodes joining the system concurrently and with nodes that fail or leave arbitrarily. For scalability, the join and leave operation should be applied with low time and message complexity. When the network system increases over time, the system should decrease the load on each node and increase the availability of the system (Naor et al, 2004:3).

3.1.1 Node Join

Peers are associated with generators of a Voronoi Diagram. Each peer holds its own location and maintains its local generated Voronoi Diagram. A peer that wishes to join the system, the so-called *joiner*, executes the following protocol:

1. The joiner sends a *WantoJoin*-message to some peer within the network, containing a randomly chosen point in *S*, the so-called *location* l_p *of the joiner*.
2. The peer receiving the *WantoJoin*-message checks, if it is responsible for the joiner's location, i.e., the location is contained in its own Voronoi region. If not, it forwards this message to the neighbor whose location is closest to the joiner's location.
3. If the peer is in fact responsible for the location of the joiner, it calculates an updated Voronoi diagram by adding the location of the joiner as a generator. Then it sends back a *Peer*- message to the joiner, including all the peers which from its point of view are neighbors to the joiner.
4. Upon receiving the *Peer*-message, the joiner replies with a *JoinAck*-message. Then it calculates its local Voronoi Diagram by *addin*g all these neighbors to its VD and validates its neighborhood from its local view. Then, the joiner sends a *Hello*-message, containing the location of the joiner, to all peers the joiner accepted as new neighbors in its local VD.
5. The receiver *R* of a *Hello*-message checks if the sender, i.e., the joiner, is already a neighbor. If not, it is inserted in the local VD and *R* validates, if all original neighbors are still neighbors after the insertion of the joiner. If they are no longer neighbors, R

sends these peers a *NeighborLeave*-message. For the remaining neighbors R tests, if they are neighbors of the joiner. R sends a *NewNeighbor*-message to the joiner, containing these peers.

6. The joiner validates, if the peers in *NewNeigbor*-messages are already neighbors. If not, they are accepted as neighbors. The joiner sends a *Hello*-message to all new neighbors.
7. When the joiner receives the first *NewNeighbor*-message, which does not contain new neighbors, it starts sending Alive-messages.

The number of nodes that need to be updated when a node joins the network is *O(1)* on average and *O(n)* in the worst case. As the mean path length is *O(log n),* finding and updating nodes takes *O(log n)* time on average. We assume that with the use of Long Distance Links this time can be significantly reduced.

The next operation that has to be performed when a node joins the network is to take over the responsibility for the objects with locations intersecting the joiners region in the VD. This is done by the DSPT-service implemented by each peer. If a spatial object is to be published, the real spatial object will be stored in a *local store* at the source peer and the location of that spatial object will be stored in a so-called *remote store* at the publishing peers whose Voronoi regions cover this location. As every published location object must be updated by the source peer, there are *Update*- and *Acknowledge*-messages which are periodically sent between the source peer and the publishing peers. If the entrance or exit of a peer causes changes of Voronoi regions, the update message is forwarded to the new responsible peers and the acknowledge messages contain the new addresses of the publishing peers.

Along the way, publishing objects in the VoroDSPT network implicate the knowledge about address information and the Voronoi regions of the publishing peers. These address information is used to create long distance links (LDLs), and the remote Voronoi region of a publishing peer, which is also sent in an *Acknowledge* message, is used for the performance-enhancing search process of the DSPT queries.

3.1.2 Node Leave

In a safe mode, peers leave the overlay network by informing neighbors about its imminent departure with a *Disconnect*-message. The neighbors remove node n from their neighborhood and recalculate their local VD. For detecting node failures the absence of *Alive*-messages are used. If no *Alive*-messages from a peer *F* are received for a certain amount of time, the neighbors of *F* assume that *F* failed or left the network. In this case they recalculate their local Voronoi Diagram and send a *NeighborLeave*-message containing the new state of the neighborhood to all their Voronoi neighbors. When all peers that were neighbors of *F* updated their neighborhood and notified their neighbors about new Voronoi-neighbors, all affected Voronoi neighbors of *F* become aware of its failure and finally each peer's local state of view of the network topology corresponds to the global Voronoi Diagram.

If a source peer that stores spatial objects fails or leaves the network, it cannot send update-messages to its publishing peers anymore. Then, these service-messages will be missed by the peer responsible for publishing the locations. After a certain amount of time, the publishing peer removes these locations from its remote store. If a publishing peer storing locations fails unexpectedly, locations are not being found for a short time, but source peers will detect the missing acknowledge-messages and republish their spatial objects automatically.

3.2 Greedy Routing

In this paper we propose a Voronoi based greedy routing for messages which contain not only point data, but also locations described in polygons and that can be routed to the destination without global knowledge of network topology or prior route discovery.

3.2.1 Voronoi based Greedy Routing

The Voronoi based greedy routing algorithm is used when a peer joins the Voronoi based overlay and forwards the service messages. A peer that gets a message containing a destination node's location needs only a lookup in its local VD to decide whether to process the message or to determine to which neighbor the message is to be sent. Examples are *keep Alive*- or *Disconnect*-messages. Messages which do not have any information about node's location address, like query messages or service messages contain a location l_o and a response address. The receiver of such a message as well has to decide whether to process or forward the message. If the location l_o intersects the spatial environment of the neighbors too, the receiver determines which neighbors are responsible for the location l_o and forwards the message eventually to several neighboring peers.

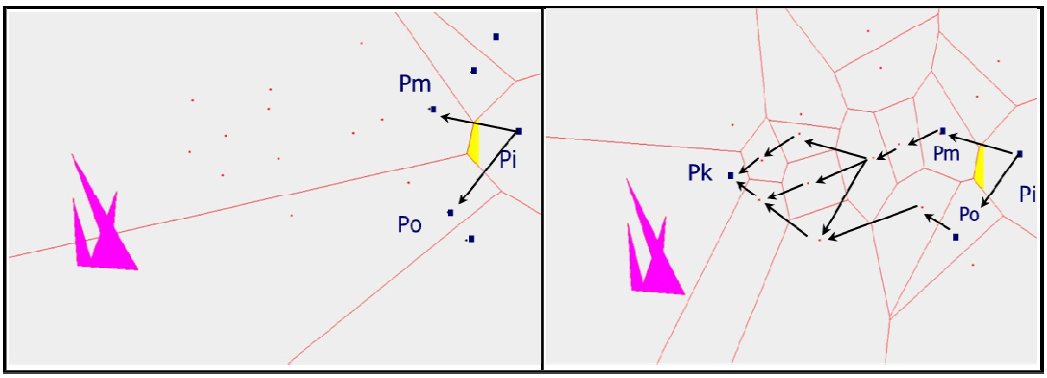

Figure 1. The local view and global view of Voronoi Diagram

Figure 1 shows two images where the whole plane is covered by the Voronoi regions of 19 peers, each managing its own Voronoi region by computing its local Voronoi Diagram. For a single peer the whole space is partitioned in $n+1$ parts where n is the number of its neighbors. The left image of figure 1 shows the local VD of peer P_i. From the local view of peer P_i the pink polygon intersects the assumed Voronoi regions of peer P_o and of peer P_m. Thus, peer P_i sends service-messages to both neighbors P_o and P_m. From the global view of the Voronoi Diagram it is obvious that the Voronoi region of peer P_k completely encompasses the geometric object. The spatial environment which peer P_i knows from its own local Voronoi Diagram does not reflect to the actual global Voronoi diagram. However, it is sufficient for a meaningful and correct routing.

The decision if a local process is necessary and the determination of the next routing hop to the destination contained in the message is done by the greedy routing algorithm. The responsible peer consults its local VD, if its own Voronoi region intersects the destination location of the message. Each peer can make a local decision if it is solely responsible for the location. In this case the greedy algorithm calculates the complete solution. Otherwise the

responsible peer checks the intersections of the location with the neighboring Voronoi regions. After determining the Voronoi-regions by the local Voronoi diagram, the message is sent to the neighbors concerned. In order to avoid message loops, redundant messages are detected by an ID which is created by the peer initiating the communication. This protocol ensures that the message reaches its destination nodes without flooding effect. It is robust against topological network changes, though it is not optimal because of message redundancy.

4. SIMULATION AND EXPERIMENTAL RESULTS

In this section, we perform a basic simulation-based performance analysis of VoroDSPT. Common properties of a P2P network usually include security, anonymity, scalability, load balance and routing efficiency. This paper will focus on the last three aspects. We evaluate the Voronoi based greedy routing by simulating a global overlay network. The performance of any routing protocol strongly depends on the length of the path between two arbitrary nodes in the network. In this context we define the path as a non-empty, acyclic directed graph $P = (V, E)$ in a *2*-dimensional Voronoi-Diagram *VD(V)*, where all nodes x_i are pair wise different and the number of edges is minimal between source- and destination node.

Let $V = x_0, ..., x_n$, $E = x_0x_1,.., x_{n-1}x_n$. The path length L_p is equivalent to the number of edges in a path *p*. The efficiency of a path length is often quantified by the number of nodes traversed during a lookup. A global Voronoi Diagram simulates the arrangement of a fixed set of *N* random generated peers in a fixed overlay network.

4.1 Path Lengths

To understand the greedy routing performance in practice, a Voronoi Diagram was generated with different sets of generators which simulate a global Voronoi overlay. The total number of peers varied from 20 to 5000 nodes and a separate experiment for each value was conducted. Each node in an experiment contacted all the other nodes in a global Voronoi Diagram, the path lengths being measured in hops. As the number of neighbors and their geographical arrangement is not regular, as this is in an *n x n* lattice network (e.g. Kleinberg's Small World network (Kleinberg, 2000)), the path between two arbitrary nodes varies and the path lengths going out and coming back have to be measured separately. The number of paths to the number of hops ratio is shown in figure 2.

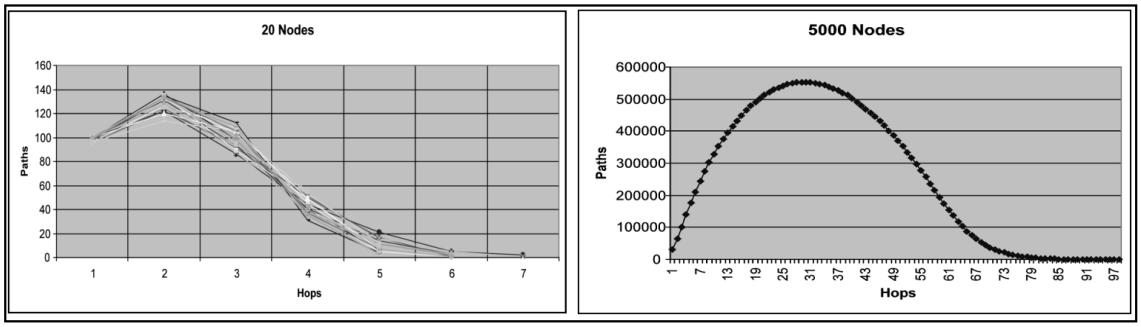

Figure 2. The distribution of hops for 20 (left image) and for 5000 nodes (right image).

The Paths/Hops ratio for 20 and 5000 nodes experiments with 20, 100, 200 and 500 nodes were run 20-fold and a new set was randomly generated every time. The experiments with 1000 and 2000 nodes were run three-fold and the experiment with 5000 nodes was only run once, due to running time.

Figure 2 illustrates that a Voronoi Diagram with 20 nodes has max $n * (n - 1) = 380$ directed paths and that at an average 98 directed paths needed only one hop and 128 paths needed 2 hops etc. To get the mean path lengths, the sum of paths which were weighted with their needed number of hops was divided by the sum of paths. Figure 3 plots the mean path lengths. As expected, the mean path length increases logarithmically with the number of nodes.

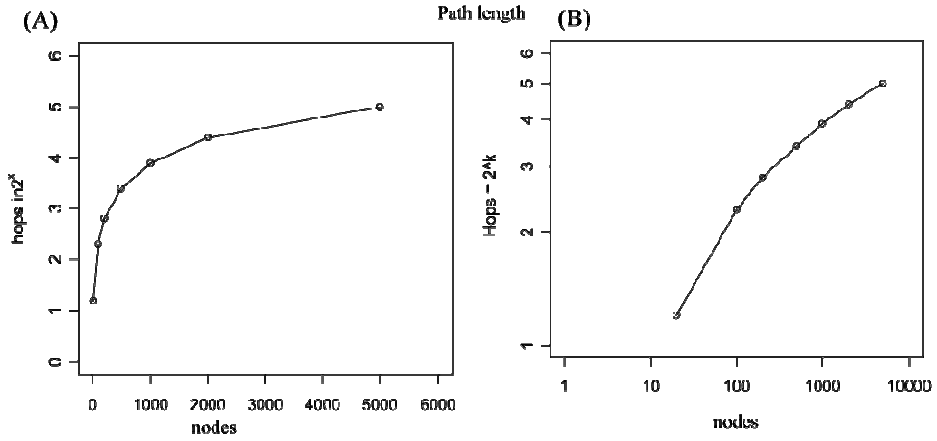

Figure 3. The mean path length of a Voronoi-based network

Figure 3 (A) plots the mean path length for a network with 5000 nodes, the x-axis denotes the nodes, the y-axis marks the hops needed with 2^5. Figure 3 (B) shows the mean path lengths in a logarithmic diagram. So with high probability the lengths of the path to resolve a query is *O(log N)*, where *N* is the total number of nodes in the network.

4.2 Worst Case Path Lengths

In this section it will be shown that there exists a worst case path lengths with *O(n)*. The greedy routing in a Voronoi Diagram is based on the property to send and forward messages to neighbors. When the node degree, the number of neighbors with which a node must maintain continuous contact, is high, then it is highly probable that at least one path to a destination will be found whose path lengths is shorter or equal to others. A peer that has only one neighbor cannot forward a message. It can only send it to its neighbor. A peer with two neighbors can forward a received message only in one direction thus the lengths of the path increases. So when all *n* generators of a path *p* lie on the same straight line, then the path length of this collinear path *p* is *n-1*. That is the worst case.

For a collinear distribution of generator-points in a 2-dimensional Voronoi Diagram we can give a formula from the greedy routing measurements which describe the worst case path lengths: let *N* be the total number of nodes, *k* the number of hops, and *p* the number of paths: *p = 2*(N - k). E.g.*, if there is a collinear distribution of *N = 20* nodes, then there exists a total number of paths *P=380*. We have exactly p=38 paths with a path length of k=1, and exact p=2 paths with path length of k=19. Figure 4 (A) shows the path distribution with 20 randomly generated nodes in a 2-dimensional Voronoi Diagram. The mean path length is 2 and the maximal measured path length was 7. In figure 5 (B) the worst case path distribution is represented with 20 collinear nodes. In this case, the mean path length is 7 and the maximal path length amounts to 19.

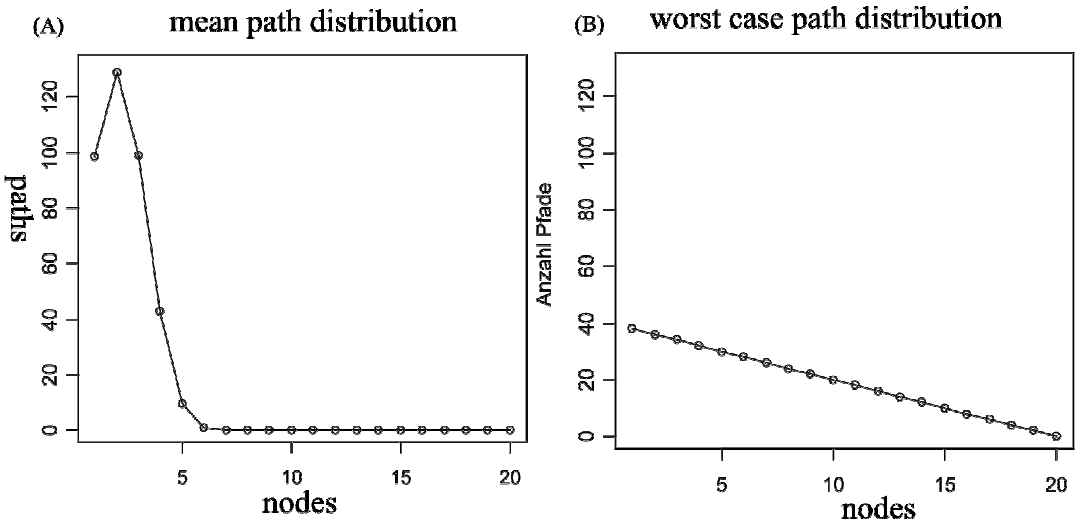

Figure 4. Path distribution with 20 nodes

We expect that the use of long distance links, which are generated by storing spatial objects, and the resulting improved spatial search process will further improve the performance of the overlay.

5. CONCLUSION

In this paper we address the problem of efficient dynamic and distributed storage and search for spatial objects in a P2P overlay that is based on a Voronoi Diagram. The VoroDSPT protocol solves this problem with the implemented data structure DSPT, which offers spatial queries: given a location, it determines all those spatial objects whose locations intersect, include or are equal to the given query location. In a nearly uniformly distributed N-node overlay, each node maintains a local Voronoi Diagram with an average neighboring number of *6*. For managing network connectivity, each node maintains routing information for only about *O(1)* neighbors. For resolving lookups the greedy routing guarantees delivering messages to other nodes with a mean path length of *O(log n)*. Updates to the routing information for nodes leaving and joining require *O(log n)* messages.

VoroDSPT features a simple and practical method for storing located objects for sharing spatial information and the exploitation of locally available information for robust collaboration of peers.

Handling spatial data in a distributed two-dimensional overlay, load balancing is the main problem that has to be solved and will be addressed in future work. In this paper we presented VoroDSPT as an approach, which provides spatial services to location-aware applications, featuring a localized efficient and scalable routing scheme.

As discussed above, the load-balancing in earlier DSPT implementations, i.e., RectNet, is not very fine grained and results in network traffic bursts. While the implementation of VoroDSPT described in this paper is static, and does not adapt to dynamic load distributions, a main motivation for following the Voronoi Diagram approach is in fact the possibility for finer grained load balancing.

One strategy for redistribution of load is the recursive space partitioning of a responsible region a peer has to self-manage. A good example is RectNet, whose search space is partitioned, based on a set of parallel axis of rectangles and where an overloaded peer splits its responsible region into further rectangles which correspond to load. To perform a dynamic load balancing in VoroDSPT, we propose to enable the individual peers to heuristically move their generators. Our current implementation is now able to perform such movement operations in addition to the join and leave operations. In addition, information about the workload of individual peers is piggybacked on the alive messages exchanged between neighboring nodes and experiments are ongoing evaluation different movement strategies. The results will be published in future work.

REFERENCES

Book

Okabe A., Boots B. and Sugihara K., 1992. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* Wiley ISBN 0471934305, Chichester, England.

Conference paper or contributed volume

Araújo F. and Rodrigues L., 2004. GeoPeer: A Location-Aware Peer-to-Peer System. *Proceedings of the Third IEEE International Symposium on Network Computing and Applications 0-7695-2242-4*, Universit at Lissabon Portugal.

Berleong R., 2007. Verteilte räumliche und typbasierte Indexstrukturen über Kontextanbietern. Studienarbeit Nr. 2080, University Stuttgart.

Casavant T.L. and Kuhl J.G., 1988. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering, vol. 14, no. 2, pp. 141-15.*

Fortune S., 1986. A Sweepline algorithm for Voronoi diagram. *Proc.2nd Annual Symp.Computational Geometry*, YorktownHeights, NY, Technical Report pp.313-322.

Heutelbeck D., Hemmje M., 2006: A Peer-to-Peer Data Structure for Dynamic Location Data. In: *Proc. of IEEE PerCom*, Pisa, Italy.

Heutelbeck D., Hemmje M., 2005: A Peer-to-Peer Rendezvous Infrastructure for Location-Based Multiplayer Games. In: *Proc. of Workshop on Computer Games and CSCW at the ECSCW'05* in Paris.

Kleinberg J., 2000, The Small-World Phenomenon: An Algorithmic Perspective, in *Proc. 32nd ACM Symp.Theory of Computing*, 163-170.

Naor and Wieder 2004. Scalable and dynamic quorum systems. Research supported in part by the *RAND/APX grant from the EU Program IST.*

Stoica I. et al, 2001. Chord: A scalable peer-to-peer lookup service for Internet applications. Technical Report TR-819, MIT.

Ratnasamy S. et al, 2001. A Scalable Content-Addressable Network. In *Proc.of the ACM SIGCOMM*, San Diego.