Geodemlia: A Robust Peer-to-Peer Overlay Supporting Location-Based Search

Christian Gross[1], Dominik Stingl[1], Björn Richerzhagen, Andreas Hemel, Ralf Steinmetz, David Hausheer[2]
Multimedia Communications Lab[1], Peer-to-Peer Systems Engineering[2],
Technische Universität Darmstadt
Email: {chrgross, stingl, richerzhagen, hemel, steinmetz, hausheer}@kom.tu-darmstadt.de

***Abstract*—Existing peer-to-peer overlay approaches for location-based search have proven to be a valid alternative to client-server-based schemes. One of the key issues of the peer-to-peer approach is the high churn rate caused by joining and leaving peers. To address this problem, this paper proposes a new location-aware peer-to-peer overlay termed Geodemlia to achieve a robust and efficient location-based search. To evaluate Geodemlia, a real world workload model for peer-to-peer location-based services is derived from traces of Twitter. Using the workload model, a system parameter analysis of Geodemlia is conducted with the goal of finding a suitable parameter configuration. In addition, the scalability and robustness of Geodemlia is compared to a state-of-the-art tree-based approach by investigating the performance and costs of both overlays under an increasing number of peers, an increasing radius of area searches, an increasing level of churn as well as for different peer placement and search request schemes. The evaluation results reveal that in contrast to the tree-based approach, Geodemlia provides on average a 46% better success ratio as well as a 18% better recall at a moderate higher traffic overhead of 13 bytes/s and an increased average response time of 0.2 s.**

***Index Terms*—Location-based search, area search, peer-to-peer, overlay, geographical search**

I. Introduction

Mobile communication is experiencing a remarkable technological progress. The wide deployment of smartphones, equipped with localization capabilities, video cameras, and wireless broadband Internet connectivity is the key enabling factor of a new class of *location-based services*. Those services enable users to publish location-based data, ranging from small pieces of information including the users current location as well as recommendations on nearby restaurants, places, or shops [8] to large data objects such as images and videos [14], [29]. By defining a particular region of interest, such location-based information can then be found by others.

This unique combination of geographically distributed information and the interest in searching for it has resulted in a variety of peer-to-peer-based (P2P) approaches for location-based search. Those approaches include on the one hand hierarchical tree-based concepts [3], [12], [18], [35]. These concepts, however, suffer from load-balancing and scalability problems as the upper levels of the tree denote a potenial bottleneck in the system. Furthermore, the root peer of the tree might fail due to churn in system such that stability issues arise. On the other hand, there exist approaches using space filling curves on top of a DHT [17]. Those approaches, however, do not preserve the directionality and locality of the multi-dimensional space, which both are important properties enabling efficient search for location-based information [15]. Thereby, locality implies that neighbored location-based information is stored on neighbored peers, whereas directionality means that the mapping of location-based information onto peers in the system preserves the orientation of the multi-dimensional space. Hence, those approaches using space filling curves require a high message overhead while searching for location-based information. In addition, the robustness and stability of the overlay including the persistent storage of data as well as reliable location-based search are a key challenge in any P2P-based approach due to the frequent joining and leaving of peers. In this context robustness means that the performance of the overlay should not drop below a certain threshold with respect to recall and success ratio even under a high churn rate.

To overcome these problems, the following contributions are presented in this paper.

- First, a novel robust peer-to-peer (P2P) overlay called Geodemlia enabling users to search for location-based information around a given geographic location is proposed. For the overlay to be robust, the design of the overlay is inspired by the well known Kademlia overlay [21]. Location-based data in Geodemlia is stored persistently even at high churn rates as it gets periodically replicated onto the k peers that are closest to the point in space that location-based information is associated with. Furthermore, peer locations and location-based information are handled by Geodemlia in a way such that the directionality and locality property are both fulfilled. Geodemlia is designed to be deployed on static peers connected via the Internet forming a P2P overlay, which allows for the persistent storage and efficient area search for location-based data that is generated and uploaded by mobile devices. Hence, it serves as a P2P-based backend for location-based services. Peers are then able to initiate search requests for location-based information around a given location, e.g., searching for italian restaurants within a 2 km radius around a peer's current location.
- Subsequently, a workload model for location-based services is presented, which was derived from traces of

[1,2] Authors supported by the German Research Foundation, Research Group 733, "QuaP2P: Quality Improvement of Peer-to-Peer Systems" and IT R&D program of MKE/KEIT of South Korea (10035587, Development of Social TV Service Enabler based on Next Generation IPTV Infrastructure).

Twitter [13], containing 22 million location-based status updates from 220,000 users. The workload model provides a realistic placement of peers and generation of location-based search requests in simulations.

- Finally, the developed prototype is evaluated in various scenarios using the derived workload model and compared to the hierarchical tree-based approach Globase [18], using the same implementation. The evaluation results reveal that Geodemlia is robust at higher churn rates and that area searches for location-based information are carried out efficiently. Furthermore, it is shown that, throughout the different evaluation scenarios, Geodemlia provides on average a 46% better success ratio as well as as 18% better recall at a moderate higher traffic overhead of 13 bytes/s and an increased average response time of 0.2 s.

The rest of this paper is structured as follows: Section II presents the system model comprising the assumptions as well as the functional requirements of the developed prototype. Section III presents the design of Geodemlia followed by Section IV dealing with the evaluation. Finally, related work is discussed in Section V and a conclusion as well as an outlook on future work are given in Section VI.

II. System Model

The design of Geodemlia is based on the following assumptions: (i) Peers are located on a sphere, which represents the shape of the earth, with each peer residing at exactly one point on the sphere. The motivation for using a sphere model is that most localization techniques return spherical coordinates, which can be directly handled by Geodemlia. (ii) Each peer p is able to determine its own location $l_p = (\phi, \psi)$ with a reasonable accuracy using well known localization techniques such as GPS, IP locator services [9], or WiFi router footprints [6]. Thereby, $\phi \in [-180°, 180°]$ denotes the longitude and $\psi \in [-90°, 90°]$ the latitude of a peer's position on the sphere. (iii) For calculating the distance $d(l_1, l_2)$ between two two arbitrary points $l_1 = (\phi_1, \psi_1)$ and $l_2 = (\phi_2, \psi_2)$ on the sphere, the Haversine formula [28] shown in Equation 1 is used.

$$d(l_1, l_2) = d(\phi_1, \psi_1, \phi_2, \psi_2) = \\ 2r \cdot arcsin\left(\sqrt{a(\phi_1, \phi_2) + b(\phi_1, \phi_2)a(\psi_1, \psi_2)}\right) \quad (1)$$

with

$$a(\phi_1, \phi_2) = sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) \quad (2)$$

and

$$b(\phi_1, \phi_2) = cos(\phi_1)\, cos(\phi_2) \quad (3)$$

(iv) All peers in the overlay are assumed to be connected over TCP/IP such that two arbitrary chosen peers in the overlay are able to exchange information with each other's as long as they know each others IP addresses and port. (v) Each stored data object o is associated with a fixed geographical location l_o as well as a set of search tags s describing that information. (vi) For each location-based search being initiated, a circular shape of the search area is assumed, although it can have any other parameterizable shape, e.g., a rectangle. (vii) Finally, each peer p and data object o is assumed to have a random identifier $i \in [0, 2^{160-1}]$.

The Geodemlia overlay provides the following interface:

- Given a FIND_NODES(l_s, k, b) request containing a point $l_s = (\phi_s, \psi_s)$ on the sphere and an integer value k, any peer p receiving this request should answer with the k closest peers with respect to the query location l_s it knows about. In order to avoid already found peers being returned multiple times by different peers, a bloom filter b of size 160 bits containing the already found peers is attached by the querying peer p to the request. Using the bloom filter, a receiving peer q can determine which peers the querying peer p already has received and can add additional peers to its response.
- Given a STORE(o, l_o) request containing an object o with location l_o, the overlay should store the object persistently, meaning that neither high churn rates nor the sudden failure of peers should lead to a loss of data. To avoid storing and replicating outdated information, a peer storing an data object can specify a maximum lifetime for each object to be stored after which it will be discarded. In this paper, however, objects are considered to be stored with an infinite lifetime.
- Given an AREA_SEARCH(l_s, r, s, b) request with the parameters $l_s = (\phi_s, \psi_s)$ denoting the longitude and latitude of the point of interest, a radius r around that point, and a search term s, the system should return all stored objects at point $l_o = (\phi_o, \psi_o)$ that fulfill the condition $d(l_s, l_o) < r$ and that match the search term s. The search term s may represent abstract categories such as restaurants, shops or keywords that describe the objects that the user is currently interested in. In addition, the peer attaches a bloom filter b that is computed from the IDs of already found peers and data objects. Based on the bloom filter, peers that receive an AREA_SEARCH(l_s, r, s, b) request can avoid including already received information into their response.

III. System Design

In the following, the design of the Geodemlia prototype is presented, including the description of the routing table structure as well as of the methods for join, leave, area search, and store. In addition, the details of the mechanism for maintaining the routing table and stored data objects are given.

A. Routing Table and Overlay Structure

In Geodemlia each peer divides the geographical space into n predefined directions $j \in [0, n-1]$ based on the bearing angle $\theta \in [-\pi, \pi]$ in radians clockwise from north as shown in Figure 1. For determining in which direction $j \in J$ a given peer $q \in P$ with position $l_q = (\phi_q, \psi_q)$ is located, the peer $p \in P$ calculates the bearing angle θ using Equation 4.

$$\theta = atan2(c, d) \quad (4)$$

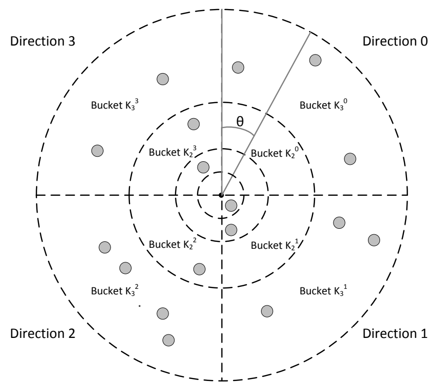


Figure 1. Routing table structure of a Geodemlia peer.

$$c = sin(\phi_q - \phi_p) \cdot cos(\psi_q) \tag{5}$$

$$d = cos(\psi_p) \cdot sin(\psi_q) - sin(\psi_p) \cdot cos(\psi_q) \cdot cos(\phi_q - \phi_p) \tag{6}$$

Based on the bearing angle θ, the direction j is calculated using Equation 7.

$$j = \left\lfloor \frac{[(\theta + 2\pi) \mod 2\pi] \cdot n}{2\pi} \right\rfloor \tag{7}$$

E.g., given a bearing $\theta = \frac{3}{2}\pi$ and a splitting factor $n = 4$, Equation 7 would return $j = 3$, which is the upper left quadrant.

The set of directions normally consist of the four cardinal points, but segmentations into any other number of directions are also possible. For each direction j, the space is further divided into distance buckets K_i^j while for each bucket at peer p in direction j the following condition holds:

$$\forall q \in K_i^j \subseteq P : d(l_p, l_q) \in [2^i, 2^{i+1}[\tag{8}$$

Each bucket K_i^j in Geodemlia stores a fixed number of k peers. For each peer being stored in the routing table, the following information is kept:

- the location $l_p = (\phi_p, \psi_p)$ of the peer,
- its peer ID $i \in [0, 2^{160-1}]$, and
- the IP address and port under which the peer is reachable via the underlay.

In total each peer maintains $|J| \cdot \log_2(d_{\text{max}}) \cdot k$ overlay contacts with d_{max} being half of the circumference of the earth.

In organizing the routing table $N(p)$ of a peer p as presented above, the *long-range connectivity* property [1] shown in Equation 9 of the overlay structure is fulfilled.

$$P\left[q \in N(p)\right] \propto \frac{1}{d(l_p, l_q)^2} \tag{9}$$

This means that chances for a peer q being part of a peer p's routing table $N(p)$ are inverse proportional to the distance $d(l_p, l_q)$ between them. In addition, the routing table structure ensures that a peer has detailed knowledge about peers close by and less knowledge about peers being further away. The same property is fulfilled by a variety of structured overlays and ensures that routing converges within a logarithmic amount of routing steps [1].

B. Find k-Closest Nodes

Finding the set of k closest peers with respect to a given location l_s is the most important and basic operation in Geodemlia. A peer issuing a FIND_NODES(l_s, k, b) request with a given location l_s and a number k of peers that should be found closest to the location l_s, first searches its routing table for the k closest peers it knows about and puts them in a list C of contacts to be queried. Afterwards, the peer computes the bloom filter b from the IDs of peers in the list C. Subsequently, it picks α peers from the list and sends them a FIND_NODES(l_s, k, b) request. Thereby, α denotes a system-wide parameter defining the number of parallel lookups. Nodes receiving that request, query their routing table for the k peers closest to the location l_s they know about and that have not been included in the bloom filter b yet. Afterwards, the peer returns a list of peers to the querying peer, which will merge the newly discovered peers into its list C of peers to be contacted, thereby ignoring already contacted peers. After that, the querying peer recomputes the bloom filter b, again chooses α yet not contacted peers from its list C and sends out another FIND_NODES(l_s, k, b) request. This procedure is repeated until the querying peer does not discover any further peers closer to the target location l_s.

C. Store

As already mentioned in Section II, each data object is assumed to be associated with a fixed geographical location l_o. In order to store a certain data object in the Geodemlia overlay, the k closest peers with respect to l_o have to be found. Therefore, the STORE(o, l_o) method internally utilizes the FIND_NODES(l_o, k, b) functionality of the overlay to find the k closest peers. After finding the set of k closest peers, the data object o is stored on them.

D. Area Search

One of the major differences between Geodemlia and Kademlia is that it provides mean for searching for location-based information given a search location l_s, a radius r around that location, and a search term s. As location-based data objects o get stored on the peers closest to their location l_o, the area search procedure AREA_SEARCH(l_s, r, s, b) has to find the following two sets of peers: (i) All peers p whose location l_p falls into the given query area, (ii) the set of closest peers that are located outside the query area but that are closest to it. The idea behind these two sets of peers is that the search area can be arbitrarily small such that no peer falls into the search area. To cope with this problem and to increase the success of area search request, the k closest peers surrounding the search area are also included in the search process. In addition, the search scheme ensures that even in scenarios with a sparse peer distribution where no peer is located inside the search area that data objects can be found.

A peer p issuing an AREA_SEARCH(l_s, r, s, b) request, checks its routing table for the set of the k closest peers to the query location l_s and puts them into a list C of peers to be contacted. Furthermore, peer p initializes a list $\overline{C}$ of peers that have not been contacted. Using the IDs of peers in list C, the bloom filter b is computed and attached to the search request. For each peer $q \in C$, the peer sends out an AREA_SEARCH(l_s, r, s, b) message. Each peer q receiving the message, first checks whether its position l_q falls into the search area. If so, it adds all stored data objects to his answer that match the search term s and that have peer been included in the bloom filter b already. Furthermore, the peer q adds at most k additional peers it knows being closest to the search area that have not been included in the bloom filter. Both, the matched data objects and the peers found are sent back to the querying peer p. After receiving the response from q, peer p first checks its list $\overline{C}$ whether it already has contacted the additionally found peers that were included in the response. If so, the additional found peers will be discarded. Otherwise, the peer p adds them to its list C of peers to be queried. In addition, it adds the received data objects o to its results list R, removes peer q from the list C, and adds it to list of already contacted peers $\overline{C}$. Finally, peer p recomputes the bloom filter including data objects from the result list R and peers from the list C. The requesting peer p continues to query peers from its list C, until it has contacted all found peers $q \in C$.

E. Join and Leave

Whenever a peer p in Geodemlia wants to join into the network, it first determines its position l_p on the sphere. This can be done by using common localization techniques such as IP address locator services, GPS, or WiFi footprints. Subsequently, the joining peer contacts a bootstrap peer it knows about. In order to find a suitable bootstrap peer, common bootstrapping approaches can be used. An overview on existing bootstrap protocols has been presented by Dickey et al. [7] and, therefore, this issue will do not be further discussed in detail in this paper.

For joining into the overlay, the joining peer adds the bootstrap peer to its routing table and issues a FIND_NODES(l_p, k, b) request using its own position l_p as query position. During this process, the peer successively discovers new peers for which the peer performs the following steps for adding each received peer q to its routing table: First, peer p determines the bucket K_i^j that q belongs to by calculating the distance between itself and the peer q as well as the bearing angle θ using Equation 4. The bearing angle θ is needed in order to determine the direction j. Subsequently, peer p checks whether bucket K_i^j has less then k peers in its table. If so, peer q is added to the bucket. Otherwise, peer p checks the liveness of the least recently contacted peer in the bucket. If the peer fails to respond, the peer is removed and the new peer is added to the bucket.

For leaving the system, a peer does not have to notify its neighbors. Surrounding peers storing a reference to the leaving peer will notice the absence of the peer, the next time they update their routing table. In Geodemlia, peers leaving the system will not delete their stored objects. The next time a peer rejoins the overlay, the information will be available again in the system.

F. Maintenance

Due to the system dynamics of joining and leaving peers, the maintenance of the routing table and the replication of stored data objects is necessary.

1) Routing Table Maintenance: A peer p's routing table gets updated in the following two cases: (i) Whenever a peer discovers a new peer contact q during lookups or requests and (ii) by regularly querying for a random location l_s lying within a given bucket K_i^j. For each newly discovered peer q a peer p performs the following steps: (i) It calculates the bearing angle θ and distance $d(p, q)$ to that peer in order to determine the bucket K_i^j. (ii) If the bucket K_i^j has less than k entries, the peer is added to the bucket. If it is already full, peer p pings the least recently seen peer s. If the the peer fails to respond, s is removed from the routing table and the newly discovered peer q is added to the tail of the bucket, similar to the update procedure in Kademlia. In case that s responds, s is moved to the tail of the bucket and the newly discovered peer q is put into a cache list of unused overlay contacts. By sorting peers in the bucket according to the time of the last interaction, communication is biased towards long living and more stable peers in the system. In addition, the system becomes more robust against routing table flooding attacks.

2) Replication of Data: In order to ensure the long-term availability of data in the system, stored location-based information is replicated. Therefore, each peer periodically starts the replication procedure every Δt_r minutes. For each stored data object o it determines the set of k closest peers form its routing table with respect to its location l_o. A peer p starting the replication procedure, contacts the set of k closest peers, whether they have already stored the object o to be replicated. All peers q from the list of k closest peers that do not have the data object respond accordingly and get a copy of it. In order to avoid all k peers storing a particular data object to start the replication procedure simultaneously, a peer only replicates a certain data object o that has not been replicated by any other of the k other peers storing the data object o within the last Δt_r minutes. A peer p belonging to the set of k closest peers storing a given data object o, will notice the replication of that data object as it will be contacted by the peer first starting the replication procedure and, thus, it will not replicate the data object again. Hence, the system parameter k determines the bucket size as well as the number of replicas of a data object o. With respect to the consistency of data, in Geodemlia each data object is generated only ones without the support for consistent updates. In case that a peer wants to modify a data object o, it creates a copy of it and stores it under a new randomly chosen ID i_o.

IV. Evaluation

In order to evaluate the performance of Geodemlia, the overlay was implemented in the discrete event-based overlay network simulator PeerfactSim.KOM [10], [32]. On top of the overlay, a location-based application was developed, that produces a workload for the overlay by generating area search requests. The goal of the evaluation is to determine

Table I
ENVIRONMENTAL AND SYSTEM PARAMETER SETUP. DEFAULT VALUES USED IN SIMULATIONS ARE UNDERLINED.

Parameter	Value
Environmental Parameters	
Peer Distribution	Germany
Size of Area	700 km x 900 km
Number of Peers	5,000
Number of Data Items	50,000
Payload per Data Item	10 KB
Underlay Delay Model	Distance-based Delay Model [16]
Session Duration	Weibull(λ_s, k_s), $\lambda_s = 169.5385$ min, $k_s = 0.61511$
Intersession Times	Weibull(λ_i, k_i), $\lambda_i = 413.6765$ min, $k_i = 0.47648$
Simulation Duration	12 h
Radius of Query Area	2 km
System Parameters	
Parallel Lookups α	$\underline{3}$, 6, 9, 12
Bucket Size k	2, 3, 5, $\underline{10}$, 20
Number of Directions d	$\underline{4}$, 6, 8
Republish Interval Δt_r	60 min
Length of the bloom filter b	160 bits

the performance and cost of Geodemlia during runtime. In addition, the evaluation should demonstrate that the developed overlay is robust in terms of high churn rates and that data is kept persistently in the overlay.

Therefore, the evaluation consists of two parts: First, a detailed performance and cost analysis is conducted, while varying different system parameters under a constant workload. Subsequently, we investigate the performance of Geodemlia under varying environmental conditions and compare the resulting performance and costs with the hierarchical tree-based approach Globase [18]. In order to obtain statistically significant results, all simulations are repeated five times using different seeds. Out of the series of different runs the average as well as confidence intervals are computed.

A. System Parameter Analysis

For investigating the impact of different system parameters on the performance and costs of Geodemlia, the environmental parameters are set to the values shown in Table I. For generating a certain dynamic in the system, the churn model in [24] is used, which is partially based on measurements conduced by Steiner et al. [31]. For determining the delay between peers, the distance-based method proposed in [16] is used. Table I also shows the different values for the varied system parameters. The underlined values denote the default value used in cases where the corresponding system parameter is not varied.

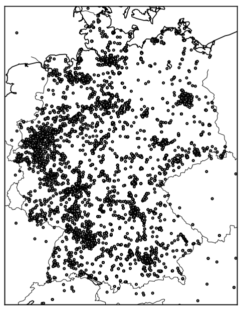
In the evaluation scenario, 5,000 peers are distributed over Germany using the distribution of peers shown in Figure 2(a), which was extracted from measurements done by Cheng et al. [5], who measured the logins of 220,000 Twitter users over half a year resulting in 22 million checkins.

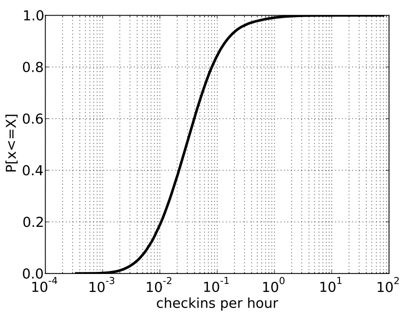
For setting up the system, the following procedure is used: First, the 5,000 peers join the system in the first hour. Afterwards, each peer publishes ten data objects according to the distribution of peers with each object having a size of 10 KB. After publishing all data objects in the system, the peer churn is enabled using the KAD churn model derived by Steiner et al. [31]. Finally, the measurement and query period of eight hours is started. For generating workload on the overlay, peers execute area searches on the system. Query requests generated by each peer are modeled as a Poisson process. For determining the mean arrival rate of requests made per hour by a particular peer, the CDF shown in Figure 2(b) is used. The value is determined once per peer at the beginning of the simulation and remains constant during the rest of the simulation. For creating an area search request, a peer p first has to chose a query location l_s, for which the following procedure is used: First, the peer draws a distance value $d_s(l_p, l_s)$ relative to its location l_p from the CDF shown in Figure 2(c), which was extracted from measurements done by Cheng et al [5]. Afterwards, it chooses a bearing angle $\theta \sim U(-\pi, \pi)$ denoting the direction in which the search location l_s is located. Based on these two parameters the search location $l_s = (\phi_s, \psi_s)$ is calculated.

1) Number of Parallel Lookups: First, the impact of a variation of the number of parallel lookups is investigated. Therefore, the value for the number of parallel lookups is set to 3, 6, 9, and 12. Figure 3(a) shows the resulting average recall for the area search operation together with the 95th confidence intervals. Thereby, the recall is defined as the ratio of correctly found data objects in the area divided by the total number of data objects that should have been found in the area. As shown in the figure, an increase in the number of parallel lookups shows no impact on the recall.

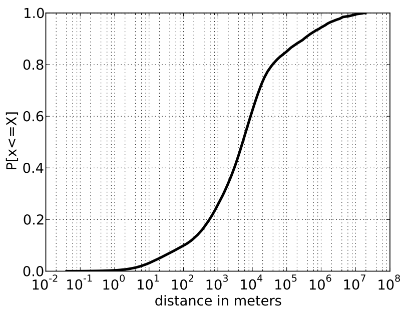
Subsequently, in order to quantify the responsiveness of the overlay the response time of executed area search requests is measured using the following method: As Geodemlia uses an iterative search approach, arriving responses to an area search request are distributed over time. Therefore, the time until the *first* data object arrives at the querying peer is measured as well as the time until the *last* data object arrives. The response times for the first and last data object to arrive at a peer are shown in Figure 3(b) and Figure 3(c). Increasing the number of parallel lookups reduces both the time for the first and last data object being returned from the system as the list of peers to be contacted during the search process can be processed faster. Dealing with the costs shown in Figure 3(d), increasing the number of parallel lookups only leads to small increase in the overlay traffic per peer. But, with an average traffic of about 0.13 kB/s the traffic of the overlay is almost negligible. In summary, the query performance can be increased by increasing the number of parallel lookups causing only a marginal increase in the costs.

2) Size of Buckets: Next, the impact of the bucket size is evaluated by varying the bucket size between 2 and 20. Geodemlia shows a good performance with respect to the recall because even with a bucket size of 2, Geodemlia provides a recall close to one as shown in Figure 4(a). A further increase in the bucket size k does not lead to a better recall. A small bucket size also leads to a reduced response time for the first data object being returned as shown in Figure 4(b). The reason for this increase in the response time


(a) Distribution of peers.

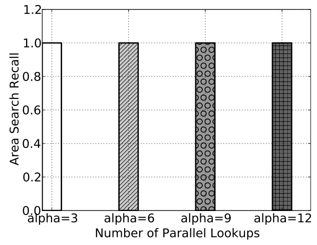


(b) CDF of the requests per hour per peer.

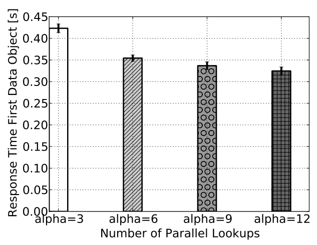


(c) CDF of the distances between search position and peer position.

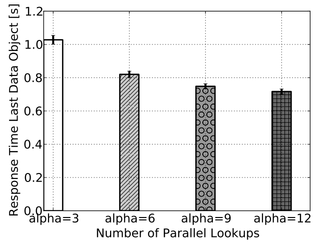
Figure 2. Peer distribution, peer activity, and distance of location-based queries extracted from the measurement data reported in [5].



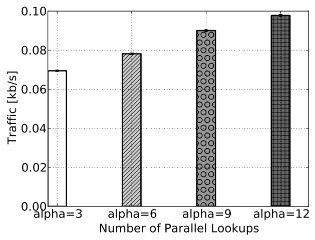
(a) Recall of area search requests.



(b) Duration of area search requests until the first data object is found.

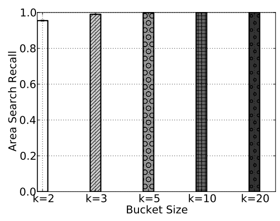


(c) Duration of area search requests until the last data object is found.

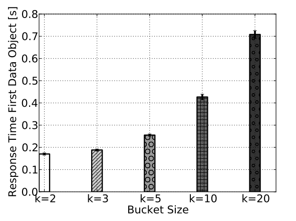


(d) Overlay Traffic per Host.

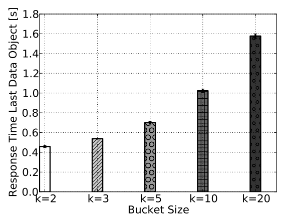
Figure 3. Performance and costs of Geodemlia for a varying number of parallel lookups ($\alpha = 3, 6, 9, 12$).



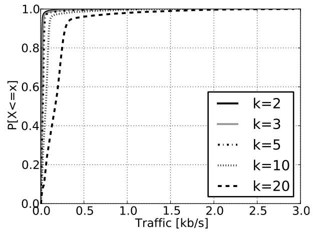
(a) Recall of area search requests.



(b) Duration of area search requests until the first data object is found.



(c) Duration of area search requests until the last data object is found.



(d) Overlay Traffic per Host.

Figure 4. Performance and costs of Geodemlia for a varying bucket sizes ($k = 2, 3, 5, 10, 20$).

is that in the current version of the overlay peers found during the area search process are sorted according to their distance to the search location l_s and not according to their stability. With a smaller bucket factor, peers kept in the buckets are those who stay longer online due to Geodemlia preference towards stable peers. With an increasing bucket factor, more unstable peers are added to the routing table, which results in a more stale neighbors being contacted during the search process. A similar behavior can be observed for the last data object being returned as shown in Figure 4(c). From the costs point of view, a larger bucket factor k causes more traffic in the overlay as shown in Figure 4(d). The reason for this traffic increase is that a peer has to ping more peers in its routing table in order to check whether they are still online. In addition, stored data objects are replicated to a larger set of k closest peers which requires more messages. Furthermore, a higher bucket factor leads to a slightly more unbalanced distribution of traffic over the peers in the overlay. To improve the load balancing capabilities of the system, the virtual servers concept presented in [25] can be applied by future versions of Geodemlia.

3) Number of Bucket Directions: Finally, the impact of the number of bucket directions on the performance and costs is investigated by increasing the number of directions in the routing table from 4 over 6 to 8. With an increasing number of dimensions Geodemlia shows a similar behavior with respect to performance and costs then with an increasing number of parallel lookups α. Therefore, the detailed results are not presented in this paper.

B. Performance Comparison

After investigating the effect of different system parameters, a performance and cost comparison between Geodemlia and Globase is conducted. Both systems provide means for searching for data or peers within a given area. Unfortunately, the original Globase implementation does not include a mechanism for replicating data, which results in a loss of data over time in the presence of churn. In order to allow for a fair comparison between the two systems, only the peer recall is calculated instead of the data object recall in order to quantify the performance. For comparing both systems the following scenarios are used: (i) Two scalability scenarios where on the one hand the number of peers is increased from $1,000$ over $\underline{5,000}$ to $10,000$ peers and on the other hand (ii) the size of the query area is varied between 1km, 2km, $\underline{\text{5km}}$, 10km and 20km. (iii) The stability of both systems is tested under different levels of churn by varying the mean session time λ_s and inter-session time λ_i, which both are linearly decreased from $\underline{\lambda}$ to $\frac{1}{16}\lambda$. (iv) Finally, both systems are compared using a uniform random peer distribution for peer locations and search requests. Again, the underlined values denote the default values. For the comparison of both systems, the system parameter configurations shown in Table II are used. The values for the system parameter configuration of Globase have been taken from [19]. For the configuration of Geodemlia, the best parameter setup derived in Section IV-A is used.

Table II
SYSTEM PARAMETER CONFIGURATIONS.

System	System Parameter	Value
Globase	Load Threshold L_1	60
	Load Threshold L_2	120
	Number of Interconnections S_1	20
	Size of Cache S_2	10
	Timeout of Operations T_1, T_2	2 s
Geodemlia	Number of Parallel Lookups α	9
	Bucket Size k	3
	Number of Directions d	4

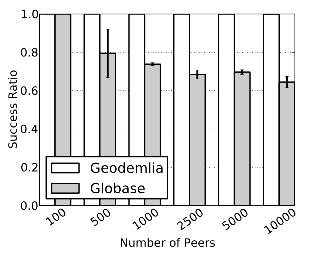
1) Number of Peers: In several experiments the number of peers is increased from 100 to 10,000 peers. The corresponding success ratio and recall of Globase and Geodemlia are shown in Figure 5(a) and Figure 5(b). For a small numbers of peers, Globase delivers a success ratio and recall close to one but with an increasing number of peers, the success ratio and recall of Globase are dropping. The reason for the good performance of Globase with 100 peers is that only one super-peer is responsible for the whole ID space as the load threshold L_2 is not exceeded. With 500 peers the load threshold L_2 is exceeded, which causes Globase to split up the ID space and to assign super-peers to the resulting regions of the ID space. This leads to the conclusion that Globase has problems in reorganizing its tree structure in case that the load threshold is exceeded. In contrast, the success ratio and recall of Geodemlia both remain close to 1. The high recall and success ratio of Geodemlia, however, come at slightly higher costs with respect to an increased response time and traffic overhead as shown in Figure 5(c) and 5(d). The reason for this is that a querying peer in Globase only has to contact the single peer responsible for the area that matches the given search area, thus, resulting in significantly less traffic. The response time as well as the traffic of Geodemlia increase logarithmically with the number of peers in the system as the measured traffic matches the logarithmic fitting curve, which shows that Geodemlia is scalable. With both systems providing response times below 100 ms and producing traffic in the range of only a couple of bytes per second, they both demonstrate that they are responsive and produce very low costs.

2) Radius of the Search Area: The impact of the size of the search area radius is investigated by increasing it from 1 km to 20 km. The success ratio and recall for different radii of the query area are shown in Figure 6(a) and 6(b). Globase shows a significant reduction in its performance as only 60% of the queries are finished successfully and, those that return a response only deliver only 63% of the data objects that should have been found in the search area with an area search radius of 20 km. The reason for this performance degradation is that with a larger query area more nodes need to be contacted in Globase to solve a query request. This involvement of a higher number of nodes is more susceptible to churn leading to a decreased recall. In contrast to the performance of Globase, the recall of Geodemlia remains close to one.

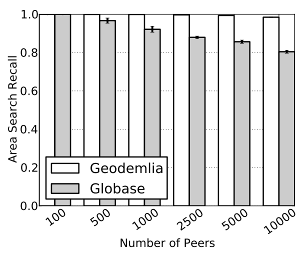
Figure 6(c) shows the response time for the first data object being returned, which for Geodemlia is rapidly dropping with an increasing size of the query area. The reason for this drop is that with a larger query area chances are higher that the querying peer itself can partially answer a request on its own due to the fact that most search requests are focusing on data nearby. With a query area of 20 km, Geodemlia is even capable of delivering the first data object faster than Globase. Globase on the other hand, shows no change in the response time with an increasing area search radius. From the area search traffic point of view Geodemlia requires more bandwidth with an increasing area size due to more peers being queried as shown in Figure 6(d). With an increasing radius of the search area, peers in Globase spent less traffic because querying peers only have to contact a few peers that are responsible for the area that intersects with the query area. In summary, the increased performance of Geodemlia comes at the costs of an increasing traffic for area search requests. On the other hand, as the traffic is in the order of magnitude of a couple of bytes per second per peer, this increase is tolerable. In addition, creating area search requests with an radius of 20 km already exceeds the typical workload of most location-based applications, as user tend to query for information directly next to them.

3) Churn Rate: Finally, the stability of both overlays is tested under an increasing churn rate. Therefore, the mean session time λ_s and inter-session time λ_i are stepwise decreased from λ to $\frac{1}{16}\lambda$. For this experiment it was necessary to exclude the root peer of Globase from the churn as the original implementation was not able to deal with a change of the root peer, which resulted in an entire collapse of the overlay. Both the success ratio and recall of Globase are slightly dropping with an decreasing session time of the peers as shown in Figure 7(a) and 7(b). In contrast, Geodemlia shows a stable behavior with a high success ratio and recall even at a high churn level with nodes having a mean session time of $\frac{1}{16}\lambda = 10$ min. The response time of Geodemlia for the first data object being found increases with a decreasing session time of the peers as shown in Figure 7(c), as potentially more stale peers are contacted during the lookup process. Furthermore, the overlay traffic slightly increases with a decreasing session time as peers join the system more often, causing additional traffic.

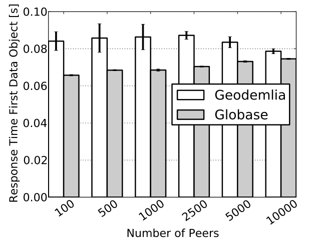
The search performance of Globase highly depends on



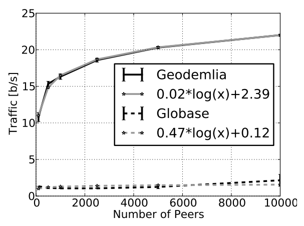
(a) Success ratio of area search requests.



(b) Recall of area search requests.

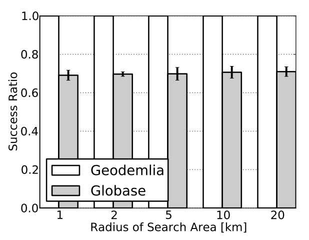


(c) Duration of area search requests until the last data object is found.

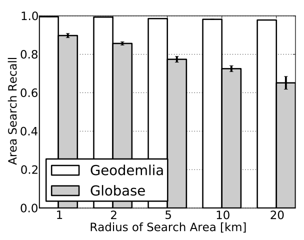


(d) Overlay traffic per peer.

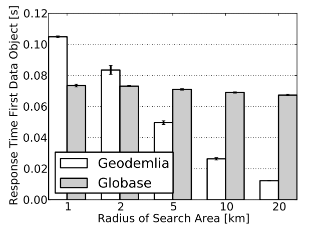
Figure 5. Comparison of Geodemlia and Globase for a varying number of peers.



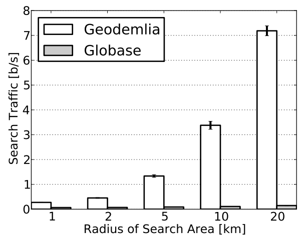
(a) Success ratio of area search requests.



(b) Recall of area search requests.

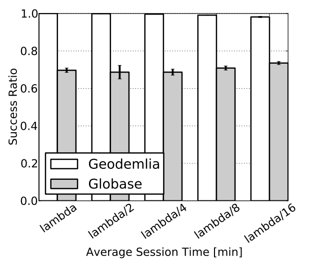


(c) Duration of area search requests until the first data object is found.

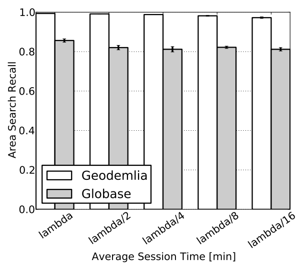


(d) Area search traffic per peer.

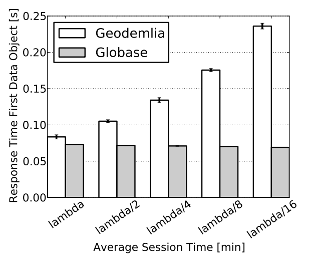
Figure 6. Comparison of Geodemlia and Globase for varying sizes of the search area.



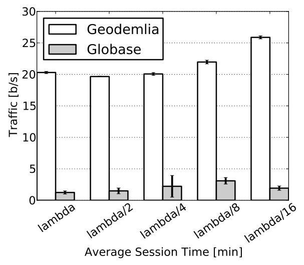
(a) Success ratio of area search requests.



(b) Recall of area search requests.



(c) Duration of area search requests until the first data object is found.



(d) Overlay traffic per peer.

Figure 7. Performance and costs of Geodemlia and Globase for varying levels of churn.

whether normal or super-peers are effected by the churn. If only normal peers join and leave the overlay, the performance remains stable as only the corresponding super-peer needs to update its routing table. In case that super-peers leave the system, a restructuring of the tree becomes necessary, which causes a much higher traffic overhead. This also explains the fluctuation in the traffic of Globase with a mean session time of $\frac{1}{2}\lambda$ as shown in Figure 7(d).

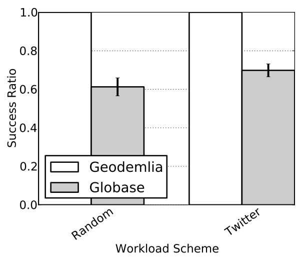
4) Peer and Request Distribution: Finally, the impact of the peer distribution on the performance and costs is investigated for both systems with 5,000 peers. Therefore, the scheme for placing peers and requests is varied such that requests and peers are uniformly distributed. Peers generate requests with a radius of 5 km. The resulting performance and costs are compared to the scenario with peers and requests being generated according to the Twitter trace files. With respect to recall and success ratio, Geodemlia outperforms Globase with recall and success ratio values close to 1 as shown in Figure 8(a) and 8(b).

From the response time point of view, both system provide and equal performance as shown in Figure 8(c). For the Twitter scenario, however, Geodemlia delivers search results slightly faster than Globase. On the other hand, Geodemlia produces more traffic due to its iterative routing and search scheme as shown in Figure 8(d).

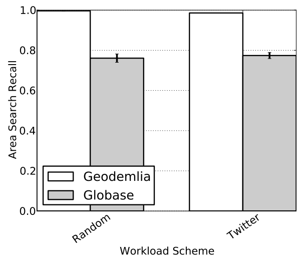
Averaging the performance and costs of Geodemlia and Globase observed in all three scenarios leads to the conclusion that Geodemlia provides on average a 46% better success ratio and 18% better recall than Globase at the cost of a minor increase in the response time of 0.2 s for the first result being returned and a 13 bytes/s higher overlay traffic per peer.

V. Related Work

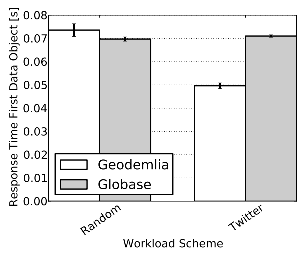
In the recent years a lot of research work has been done in the area of location-based services, resulting a plethora of different approaches for location-based search. These approaches can be grouped into two categories: (i) approaches that utilize an underlying Distributed Hash Table (DHT) using the key-value lookup functionality to support location-based search [11], [15], [20], [22], [33], [36], [37]. In order to do so, linearization techniques are required in order to map the multi-dimensional space onto the one dimensional ID space of the DHT. (ii) Approaches that were designed from scratch for solving the problem of location-based search [2], [3], [12],



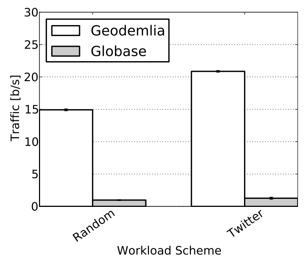
(a) Success ratio of area search requests.



(b) Recall of area search requests.



(c) Duration of area search requests until the first data object is found.



(d) Overlay traffic per peer.

Figure 8. Performance and costs of Geodemlia and Globase for varying peer placement and request distributions.

[18], [23], [30], [34] from which most of them utilize a tree-structure. According to Asaduzzaman et al. the unique combination of location-based search and the geographic distribution of information providing peers suggests the development of such a dedicated overlay supporting the locality of peers [3].

With respect to the first category, a variety of approaches have been developed that utilize space filling curves [4], [17] in order to solve the two elementary problems: (i) mapping a multi-dimensional space onto a one dimensional ID space of a DHT, while (ii) preserving the locality of peers. According to Knoll et al. [17] finding an optimal mapping that solves both problems is impossible. Therefore, the authors conducted a performance study on various approaches for space filling curves and found out that S-shaped curves as well as Lebesque curves perform poorly whereas more complex approaches such as the Hilbert curve show a better locality property. But still, most of the approaches suffer from a poor locality preserving property such as the Z-filling curves used in PlaceLab [4].

In addition to the space filling curves, a variety of approaches have been developed that utilize a tree structure and map this structure onto a DHT. Harwood [11], Tang [33], Nam [22], and Tanin [34] recursively split up the two dimensional space using a space partitioning tree. To each region a control point is assigned which is hashed onto the DHT identifier space. The peer responsible for that ID is responsible for that particular area of the geographical space. While tree structures allow for a fast an efficient access to data, they are susceptible to system dynamics such as the frequent join an leaving of peers. Other approaches [15] present a grid based splitting scheme for organizing the peer responsibilities of the multi-dimensional space and map it onto the one dimensional space of a DHT. While this scheme allows for the reuse of the well known DHTs, it suffers from performance drawbacks [3]. Lopes et al. [20] proposes a space partitioning scheme using the B^+-algorithm for addressing the load balancing problems of tree-based approaches such as PHT [4] and DST [37]. Although addressing the load balancing problem, the authors did not evaluate their system under churn leaving the stability and robustness characteristics of their system unclear.

The second category, containing the stand-alone approaches for location-based search, can be further divided into the hierarchical and flat approaches. One of the most cited approaches for location-based search has been developed by Kovacevic et al. [18]. Globase is a hierarchically structured overlay using a super-peer concept. While the tree-structure allows for a fast access within $O(log(n))$ routing steps, the tree structure causes a higher maintenance overhead and is less robust in terms of high churn rates. Especially, in terms of super-peers failing, network partitions become likely. In addition, weak peers might get selected as a super-peers which easily get overloaded, resulting in the overlay to become unstable.

An approach very similar to Globase is RectNet developed by Heutelbeck et al. [12]. RectNet uses a binary distributed space partitioning tree which simplifies the recovery in case of peer failures but reduces the search performance. Furthermore, RectNet does not include any load-balancing capabilities, resulting in overloaded peers in the higher levels of the tree.

Asaduzzaman et al. [3] propose an overlay called GeoP2P, which uses a hierarchical splitting of the 2D-space. Based on the splitting a binary tree is constructed. The splitting either is done taking cluster information of peers into consideration or by splitting the 2D space in equally sized regions. The approach has the disadvantage that the splitting has to be recomputed whenever larger amounts of peers join or leave the system, which causes additional overhead. Furthermore, the split and merge operation require some consensus algorithm to determine the responsible peer, which leaves doubts of the robustness of the system. Finally, the authors do not present any evaluation results of their proposed system and just show a brief theoretical analysis of the system performance.

Another location-aware overlay called GeoPeer has been proposed by Araujo et al. [2], which uses a Delaunay triangulation to build a connected lattice of peers. The use of a Delaunay triangulation causes additional overhead as it has to be recomputed every time a peer joins or leaves the system. Furthermore, the system does not support persistent storage of data due to the lack of a replication scheme. Geodemlia on the other hand, comes with a built-in replication scheme, which ensures the long-term availability of data. In addition, the system causes less overhead in terms of high system dynamics as routing tables are recomputed periodically and not every time peers join or leave the system.

Picone et al. [23] proposed an overlay approach for location-based search similar to the Geodemlia overlay as its design is also inspired by the prominent Kademlia overlay. Unlike Geodemlia, the overlay focuses on pure mobile scenarios, which results in a different construction of the routing table. In addition, the overlay focuses only on finding the closest peer for a given location. Geodemlia extends this functionality by also addressing area search functionality for finding peers and data objects in a given region.

Song et al. present a system called FAN [30] supporting multi-dimensional attribute search. Therefore, peers are mapped onto a d-dimensional Cartesian space. Subspaces in FAN are managed by super-peers, which require higher computational resources than regular peers. Searching in FAN corresponds to finding the particular subspace that matches the query criteria. The super-peer concept organizes the regions in a hierarchical way, such that super peers might become a bottleneck.

Finally, several attempts have been made in the area of multi-dimensional DHTs such as CAN [26] and hypercube-based systems like HyperCuP [27], which either suffer from stability issues in terms of a small number of dimensions being used or do not scale well. In contrast, Geodemlia has proven to be scalable and robust in terms of churn.

VI. Conclusion

In this paper a novel overlay supporting location-based search termed Geodemlia was presented, whose design is inspired by the well known Kademlia overlay. The evaluation results revealed that Geodemlia provides better robustness and stability capabilities in comparison to a tree-based approach. The latter causes less traffic, but maintaining the tree-structure in the presence of churn is a difficult task.

For future work it is planned to further optimize Geodemlia such that the overlay traffic can be further reduced. Additionally, it is planned to extend the performance and cost comparison and include other approaches such as space filling curves. Finally, it is planned to implement Geodemlia as a real prototype and to evaluate it in a testbed such as G-Lab or PlanetLab.

VII. Acknowledgments

The authors would like to thank Zhiyuan Cheng from Texas A&M University for providing them with the location sharing services dataset.

References

[1] K. Aberer, L. Alima, A. Ghodsi, S. Girdzijauskas, S. Haridi, and M. Hauswirth, "The Essence of P2P: A Reference Architecture for Overlay Networks," in *P2P*. IEEE, 2005.

[2] F. Araujio and L. Rodrigues, "Geopeer: A Location-Aware Peer-to-Peer System," in *Network Computing and Applications,*. IEEE, 2004.

[3] S. Asaduzzaman and G. Bochmann, "GeoP2P: an Adaptive and Fault-tolerant Peer-to-Peer Overlay for Location-Based Search," in *ICDCS*, 2009.

[4] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, S. Shenker, and J. Hellerstein, "A Case Study in Building Layered DHT Applications," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 97–108, 2005.

[5] Z. Cheng, J. Caverlee, K. Lee, and D. Sui, "Exploring Millions of Footprints in Location Sharing Services," in *ICWSM*. AAAI, 2011.

[6] I. Constandache, R. Choudhury, and I. Rhee, "Towards Mobile Phone Localization without War-Driving," in *INFOCOM*. IEEE, 2010.

[7] C. Dickey and C. Grothoff, "Bootstrapping of Peer-to-Peer Networks," in *SAINT*. IEEE, 2008.

[8] T. D'Roza and G. Bilchev, "An Overview of Location-Based Services," *BT Technology Journal*, vol. 21, no. 1, pp. 20–27, 2003.

[9] geobytes.com, "IP Locator Service," 2012, http://www.geobytes.com.

[10] C. Groß, M. Lehn, D. Stingl, A. Kovacevic, A. Buchmann, and R. Steinmetz, "Towards a Common Interface for Overlay Network Simulators," in *ICPADS*. IEEE, 2010.

[11] A. Harwood and E. Tanin, "Hashing Spatial Content over Peer-to-Peer Networks," in *Australian Telecommunications, Networks and Applications Conference*. Citeseer, 2003.

[12] D. Heutelbeck and M. Hemmje, "A Peer-to-Peer Data Structure for Dynamic Location Data," *PERCOM*, 2006.

[13] B. A. Huberman, D. M. Romero, and F. Wu, "Social Networks that Matter: Twitter under the Microscope," *CoRR*, vol. abs/0812.1045, 2008.

[14] I. A. Junglas and R. T. Watson, "Location-Based Services," *Communications of the ACM*, vol. 51, no. 3, pp. 65–69, 2008.

[15] V. Kantere, S. Skiadopoulos, and T. Sellis, "Storing and Indexing Spatial Data in P2P Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 2, pp. 287–300, 2009.

[16] S. Kaune, M. Wählisch, and K. Pussep, *Modeling and Tools for Network Simulation: Modeling the Internet Delay Space and its Application in Large Scale P2P Simulations*. Springer, 2010, pp. 427–446.

[17] M. Knoll and T. Weis, "Optimizing Locality for Self-Organizing Context-Based Systems," *Self-Organizing Systems*, pp. 62–73, 2006.

[18] A. Kovacevic, N. Liebau, and R. Steinmetz, "Globase.KOM - A P2P Overlay for Fully Retrievable Location-Based Search," in *P2P*. IEEE, 2007.

[19] A. Kovacevic, "Peer-to-Peer Location-Based Search: Engineering a Novel Peer-to-Peer Overlay Network," Ph.D. dissertation, Technische Universität Darmstadt, 2009.

[20] N. Lopes and C. Baquero, "Implementing Range Queries with a Decentralized Balanced Tree over Distributed Hash Tables," in *International Conference on Network-based Information Systems*. Springer, 2007.

[21] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," *Peer-to-Peer Systems*, vol. 1, pp. 53–65, 2002.

[22] B. Nam and A. Sussman, "DiST: Fully Decentralized Indexing for Querying Distributed Multidimensional Datasets," in *Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2006.

[23] M. Picone, M. Amoretti, and F. Zanichelli, "Geokad: A P2P Distributed Localization Protocol," in *PERCOM Workshop*. IEEE, 2010.

[24] K. Pussep, C. Leng, and S. Kaune, "Modeling User Behavior in P2P Systems," in *Modeling Tools for Network Simulation*. Springer, 2010, no. July, ch. Modeling User Behavior, pp. 447–461.

[25] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I, "Load Balancing in Structured P2P Systems," *Peer-to-Peer Systems*, vol. 1, no. 4, pp. 100–103, Feb. 2003.

[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 161–172, 2001.

[27] M. Schlosser, M. Sintek, and S. Decker, "HyperCuP - Hypercubes, Ontologies and Efficient Search on Peer-to-peer Networks," *Agents and Peer-to-Peer*, 2003.

[28] R. Sinnott, "Virtues of the Haversine," *Sky and Telescope*, vol. 68, p. 158, 1984.

[29] E. Snekkenes, "Concepts for Personal Location Privacy Policies," in *ACM Conference on Electronic Commerce*, 2001.

[30] W. Song, R. Li, Z. Lu, and G. Yu, "FAN: A Scalable Flabellate P2P Overlay Supporting Multi-Dimensional Attributes," in *Advanced Information Networking and Applications (AINA)*. IEEE, 2008.

[31] M. Steiner, T. En-Najjary, and E. Biersack, "A Global View of KAD," in *ACM SIGCOMM Conference on Internet Measurement*. ACM, 2007.

[32] D. Stingl, C. Groß, J. Rückert, L. Nobach, A. Kovacevic, and R. Steinmetz, "PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems," in *HPCS*. IEEE, 2011.

[33] Y. Tang and S. Zhou, "LHT: A Low-Maintenance Indexing Scheme over DHTs," in *ICDCS*. IEEE, 2008.

[34] E. Tanin and A. Harwood, "Using a Distributed Quadtree Index in Peer-to-Peer Networks," *Journal on Very Large Data Bases*, vol. 16, no. 2, pp. 165–178, Apr. 2007.

[35] D. Tran and T. Nguyen, "Hierarchical Multidimensional Search in Peer-to-Peer Networks," *Computer Communications*, vol. 31, no. 2, pp. 346–357, Feb. 2008.

[36] T. Zahn, G. Wittenburg, and J. Schiller, "Towards Efficient Range Queries in Mobile Ad hoc Networks using DHTs," in *MobiShare*. ACM, 2006.

[37] C. Zheng, G. Shen, S. Li, and S. Shenker, "Distributed Segment Tree: Support of Range Query and Cover Query over DHT," in *IPTPS*, 2006.