Globase.KOM - A Peer-to-Peer Overlay for Fully Retrievable Location-based Search

Aleksandra Kovačević, Nicolas Liebau, and Ralf Steinmetz
Multimedia Communications Lab (KOM)
Technische Universität Darmstadt, Germany
Email: {sandra, liebau, steinmetz}@KOM.tu-darmstadt.de

***Abstract*— Location based services are becoming increasingly popular as devices that determine the geographical position got more available to the end users. The problem of existing solutions to location-based search, e.g. in car navigation systems, is keeping information updated because of centralized maintenance at specific times. Therefore, retrieved results do not include all objects that exist in reality. A peer-to-peer approach can easily overcome this issue as peers are responsible for the information users are searching for. Unfortunately, current state-of-the-art overlays cannot fulfill the requirements for efficient and fully retrievable location-based search. In this paper we present Globase.KOM, a hierarchical tree-based peer-to-peer overlay that enables fully retrievable area search, lookup, and finding the geographically closest node. Simulation results prove that the overlay operations of Globase.KOM are highly efficient, fully retrievable, and logarithmically scalable. Protocol overhead and load of the peers in the overlay have been evaluated as well.**

I. Introduction

Location based services are becoming increasingly popular – navigation systems, tracking and logistic systems, mobile billing (e.g. toll) or mobile advertising are just some of them. Search for objects/information located in some area are part of many search engines or navigation systems. Additionally, devices that determine the geographical position (e.g. a GPS receiver) are already integrated in many Pocket PCs and integration into standard laptops and other end-systems can be expected soon. This location information enables for more convenient, highly personalized services for users, like for example, finding the closest restaurant. In existing solutions for location-based search the search results are often incomplete or outdated. Information in a peer-to-peer-based solution could be kept updated much better than in centrally managed systems even with the fast growth of the system. The cases of passing by a gasoline station while our navigation system shows that the closest gasoline station is 5 km away, or navigating through a blocked road can be avoided. Further, the system could be operated with low costs, because of the natural scalability and administration-free character of peer-to-peer systems. In spite of the fact that the peer-to-peer research community is very active since the last seven years, current state-of-the-art overlays cannot fulfill the requirements for efficient and fully retrievable location-based search. In this paper we present Globase.KOM, a hierarchical tree-based peer-to-peer overlay that enables fully retrievable area search, lookup, and finding the geographically closest node.

This paper is organized as follows. In the Section II we discuss the goals, requirements, and assumption for our solution. The design of Globase.KOM – overlay structure, overlay operations, and failure recovery, is presented in Section III. Evaluation in Section IV shows the performance of overlay operations as well as protocol overhead, load balance, and effects of overlay parameters on the performance. The related work is surveyed in Section V. The paper is concluded in Section VI.

II. Goal, Requirements, and Assumptions

Our main focus is enabling search for *all* peers in some defined geographical area. The area can be a circle with a given center and radius or a point with two lengths for a rectangular area. Peers can search for all peers in the defined geographical area, for a peer with some particular location, or for the geographically closest peer. Together with the information about its geographical location, a peer can publish any other data describing the service it offers (e.g. a video stream from a webcam), the object it represents (e.g. restaurant, police station, sightseeing, gasoline station), or some additional information (e.g. menu, prices, opening hours). For example, users can find the closest gasoline station or can find all restaurants in some area and see their menu or video streams from webcams.

We assume that each peer is aware of its exact geographical position using appropriate devices (e.g. a GPS receivers). In order to represent the two-dimensional curved surface of the Earth on a plane, we use the Plate Carée projection. This projection plots directly latitude-longitude points on a regular X, Y graph assuming the Earth is a sphere. The longitude lines on the graph are spaced using the same scale as the latitude lines, forming a grid of equal rectangles. All map projections have in common that they introduce some kind of distortion, because an ellipsoid can not be mapped without stretching, tearing, or shrinking it to a plane. The distortion introduced by the Plate Carée grows with the latitude. For zones lying on the equator there is little distortion but zones far away from it are strongly distorted. This distortion has to be taken into account when performing geographical calculations. If we want to search for zones lying within a specified radius of a point on the surface of the Earth, this circle is transformed into an ellipse on the overlay's flat projection.

III. Design

This section describes our solution that fulfills the previously given requirements and assumptions – a peer-to-peer overlay for *fully retrievable* **G**eographical **LO**cation **BA**sed **SE**arch (Globase.KOM). We describe its overlay structure, overlay operations, and failure recovery.

A. Overlay Structure

Globase.KOM is a superpeer-based overlay forming a tree enhanced with interconnections. The world projection is divided in rectangular, not overlapping zones, like presented in Figure 1. Each zone is assigned to a superpeer that is located inside of the zone and keeps overlay/underlay contact addresses to all peers in that zone. Superpeers form the tree where peer A is called the parent of peer B when B's zone is inside A's zone.

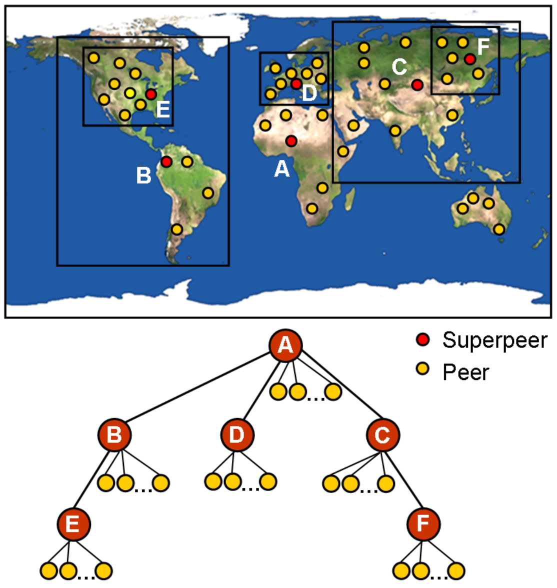


Fig. 1. Basic structure of the Globase.KOM overlay

Each superpeer maintains the overlay/underlay contact addresses of:

- the peers inside of its zone, excluding the inner zones,
- superpeers responsible for inner zones (children in a tree),
- the parent in a tree,
- the root superpeer, and
- interconnected superpeers (see III-A.2).

Each peer maintains the following contact addresses:

- the parent superpeer,
- the root superpeer,
- an interconnection list, and
- a cache list of already contacted peers.

Peers/superpeers are identified by its unique ID with the structure presented on figure 2. The PeerID contains the following information:

- the GPS coordinate of the node,

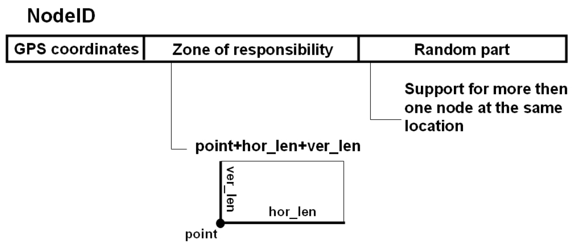


Fig. 2. Structure of PeerID

- if it is a supernode, the zone it is responsible for, and
- random part in order to support the existence of more than one peer on the same location.

A rectangular zone is simply described as concatenation of its vector representation - left bottom point of the zone and its side lengths.

1) Forming the Zones: When bootstrapping the system[1], there is just one zone, the whole world, which is assigned to the first superpeer. Peers with high CPU power, with good network connection, and with a history of long online times, are marked as potential superpeer. When the network grows, highly loaded areas are clustered into rectangular zones using a clustering algorithm and assigned to one peer (preferably to one in that area) that then becomes a superpeer by taking over the zone. As metric for the load of an area we use the number of peers connected to the superpeer as this directly influences the number of messages a superpeer receives on average and how many contacts it has to maintain. There are three load levels - *normal* (bellow a threshold L_1), *overloaded* (between thresholds L_1 and L_2), and *critically overloaded* (above L_2). Once a superpeer's load exceeds the threshold L_2, it is running a single linkage clustering algorithm in order to create a new zone inside its own zone. The new zone is then assigned to one of the peers from the formed zone that are marked as potential superpeer.

2) Interconnections: As mentioned above, each superpeer maintains interconnections to other superpeers besides its parent and children. Also, each peer caches contact information of other superpeers besides the one it is connected to. The main purpose of these so-called 'interconnections' is to provide robustness and fault-tolerance. Additionally, we can use them for more efficient query responses as well as in the case of a degenerated tree. Reiter [1] presented an algorithm for constructing a fault-tolerant communication structure out of a core tree structure where each node initially knows only its parent and children. Their focus is the construction of an expander graph from a tree, using a random walk for collecting new edges such that the nodes in the graph have node degrees close to some constant. The tree reconstruction after failures is done using new edges and it highly relies on the root of the tree. Our approach modifies Reiter's approach in order

[1]More sophisticated bootstrapping mechanisms are envisioned for the system but for the purpose of understanding the basic idea of the overlay this simple bootstrapping mechanism is sufficient

to avoid relying on the root superpeer in tree reconstruction. Instead, we use interconnections which can direct us to new parents/children. As a random walk introduces additional protocol overhead and traffic, our approach learns about new contacts through received messages instead (a similar approach is used successfully in the Kademlia overlay network [2]). Each query message includes the address of the query initiator and the address of the responsible superpeer. Upon receiving a message, each superpeer/peer checks if the initiator is its parent or child and if it is part of its subtree. Checking is done with simple calculation of the described zone in the sender's ID. If the sender is not a parent or a child, then the recipient is adding an appropriate contact to its interconnection list. The size of an interconnection list allows at least one contact per subtree. This number represents the number of children of the root superpeer. Since the maximum number of inner, not overlapping zones is fixed, size of an interconnection list takes the same value. Interconnections enables for each peer a rough view on the tree structure so that tree recovery actions and searches are more optimized. They are most valuable when the root superpeer fails because they can recover the peers from the affected zone and reconstruct the rest of the tree. Simulation results also show that interconnections have a significant role in eliminating the effect of an unbalanced tree.

B. Overlay Operations

Besides joining and leaving the network, in Globase.KOM peers can do area searches (III-B.2), lookups for a particular peer (III-B.1), or finding the geographically closest node (III-B.3).

1) Lookup: The lookup operation is used to determine the underlay address (IP address and port) of an overlay peer from its overlay ID. In our case, LOOKUP message contains the geographical location of a requested peer. Each superpeer knows the IDs/locations of all nodes it is responsible for. A lookup operation basically means routing the LOOKUP message to the superpeer responsible for the peer with the given location. A peer that performs the lookup will first contact its superpeer, sending him a LOOKUP message, with a sequence number, the reply address and the address of the responsible superpeer. The superpeer then checks whether the given location is inside of its zone. If this is the case, then it checks whether it is responsible for this location or its child. If it is not inside of its zone, it first checks if it is inside of the zones of its interconnections in order to forward the LOOKUP message directly. Otherwise, it forwards the LOOKUP message further to its parent superpeer who repeats the same actions. Finally, the superpeer who is responsible for the queried location sends a LOOKUP_RESULT including the overlay/underlay address of the node if it exist or *null* if it does not exist.

2) Area Search: Area search is performed using the SEARCH message that includes the description of the geographical area (center and radius) plus metadata describing the service/object that is searched. When a superpeer receives a SEARCH query from one of its peers, it calculates the searched eclipse onto the map projection. Further, it checks if that eclipse intersects the zone it is responsible for. All further actions are the same like in LOOKUP with the difference that all superpeers responsible for the peers inside of the searched area send a SEARCH_RESULT with a list of matching nodes. The search is considered finished after a specific timeout. Simulation studies showed that optimal values for this timeout is 2 seconds. For the each received message, interconnections are kept like described in Section III-A.2.

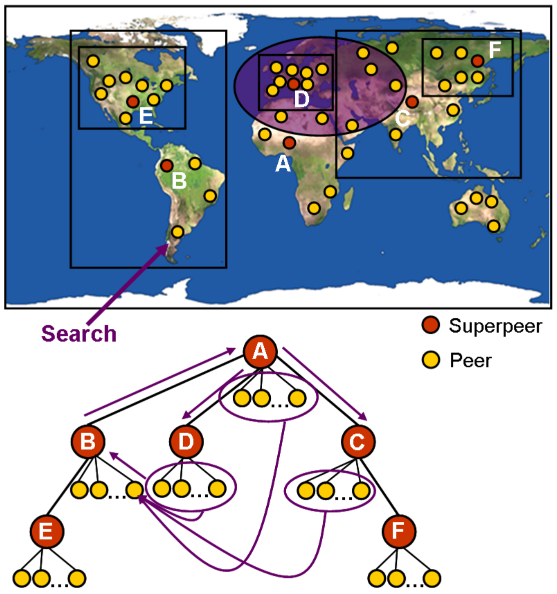


Fig. 3. Example of performing an area search

An example of an area search is given in Figure 3. A peer in the zone of superpeer B sends a SEARCH message containing a description of the marked zone. As the zone does not intersect the zone superpeer B is responsible for, the SEARCH message will be forwarded to the superpeer A. At the end, superpeers A, C, and D retrieve the list of the matching results.

3) Find the Closest Peer: When peer wants to find the closest peer, it first calculate the closest border of the zone it is belonging to. It is possible using ID of parent superpeer, which contains the vector representation of the zone. Then peer sends FIND_CLOSEST message to its parent superpeer, containing the calculated distance to the closest border of the zone. If there are some peers in the area around the initiator, with radius of given distance, superpeer calculates the closest and include it in FIND_CLOSEST_RESULT message. Otherwise, it sends FIND_CLOSEST_NEXT message containing the address of its parent superpeer. Then peer again calculates the closest border of the zone retrieved superpeer is responsible for and send it again the message FIND_CLOSEST. The steps are then analogous continuing until peer receives FIND_CLOSEST_RESULT message containing the contact of the closest peer.

4) Join: When a peer wants to join, his bootstrapping superpeer routes NEW_PEER message in order to find the zone peer is located in. In the case that the peer is first to join, the process is described in Section III-A.1. The found superpeer forms the peerID for the new peer, adds it to its peer list, and marking it if it is a potential superpeer. It sends back PEER_OK message containing the formed PeerID, contact of root superpeer, and a contact of an interconnected superpeer.

5) Leave: When leaving the network, a superpeer has to inform its parent superpeer by sending it REMOVE_SUPERPEER message with all contacts it maintains - the list of the peers, children superpeers, and interconnection lists. Informed superpeer then removes the leaving superpeer from the list of its children, taking over the responsibility for all received peers and children superpeers. It sends REFRESH_SUPERPEER message to its new children superpeers. Afterwards, it checks its load and if it exceeds L_2, it forms a new zone as described in III-A.1. In the case the root superpeer leaves the network, it sends NEW_ROOT to an appropriate marked children peer which takes over the responsibility for all peers and children superpeers of the leaving root superpeer. The new root superpeer sends REFRESH_ROOT to all superpeers, through the tree.

C. Failure Recovery

As the structure of the overlay is made of three kinds of peers, the failure of them leads to different effects and therefore their failure has to be handled appropriately. Thus, we distinguish three kinds of failure recoveries - failure of the peers, failure of the superpeers, and failure of the root superpeer. The introduced overhead is measured in Section IV-E.

1) Failure of a Peer: Each peer sends periodically KEEP_ALIVE message (without body fields) to its superpeer notifying him that it is still online. When a superpeer misses a sequence of 3 KEEP_ALIVE messages, the peer is simply removed from the superpeer's list.

2) Failure of a Superpeer: Each superpeer sends periodically a KEEP_ALIVE message to the parent and children superpeers, as well as to its peers. When a superpeer fails, its peers, children and parent superpeer do not receive the appropriate KEEP_ALIVE message. Superpeers send to failed superpeer KILL message and parent superpeer remove failed superpeer from the list of children superpeers. Children of failed superpeer (both peers and superpeers) check if their location is inside of the zone of their interconnections. If it is the case, then they send TAKE_ME message to interconnected superpeer, otherwise to root superpeer. The receiver of a TAKE_ME message is routing the appropriate message to the smallest zone containing the location of the sender. Reached superpeer (most probably the parent of the failed superpeer, if it is still online) adds peer to its peer list or set superpeer as new child superpeer and sends in both cases a REFRESH_SUPERPEER message to it.

3) Failure of the Root Superpeer: In the case the root superpeer fails, one of the children superpeers takes over the responsibility of the failed root superpeer. In order to avoid the case of forming several independent trees, we apply the *Election on Bully* algorithm [3]. Therefore, all children superpeers of the root superpeer keep connections to each other. The contacts kept refreshed through REFRESH_BROTHERS message from the root superpeer. Using the election algorithm, when a child superpeer notices the failure of root superpeer, it starts the election where it chooses the brother with the highest ID and sends him an election message with a sequence number. As soon as the election is finished according to the algorithm, the elected superpeer takes over the zone of the root superpeer and sends REFRESH_ROOT messages to all superpeers.

IV. PERFORMANCE EVALUATION

The main goal of the evaluation of Globase.KOM is to show its efficiency – the ratio of the achieved performance to the introduced costs. The achieved performance is evaluated observing overlay operations - lookup and area search with the focus on retrievability. An additional gain of Globase.KOM is better underlying topology awareness, reflected on the so called 'relative delay penalty'. The introduced costs are observed by measuring the protocol overhead and the load balance among the peers. The effects of the overlay parameters, thresholds L_1 and L_2 on the overall performance of the overlay are included in evaluation as well.

A. Evaluation Setup

Simulation is the most appropriate evaluation method for large-scale peer-to-peer overlay networks. We used PeerfactSim.KOM [4], a simulator for peer-to-peer systems that models geographical-location based peer distribution and churn. The underlying network model abstracts geographical distance between peers, the processing delay of intermediate systems, signal propagation, congestions, retransmission, and packet loss. The resource model includes peers' bandwidth model and support for message priorities with appropriate scheduling mechanisms. In order to get a realistic model of the peer distribution over the world, a grayscale colored bitmap of the world is used. Sparser areas are lighter in grayscale and darker areas are corresponding to the denser populated areas. Therefore, the darker a point in the bitmap is, the higher is the probability that a peer will be mapped to this location. The bitmap is created using the world map of Internet users by country [5].

The metrics that are used for the evaluation of overlay operations are number of hops, operation duration, and relative delay penalty. Whereas number of hops and operation duration are the common used metrics for evaluation of peer-to-peer overlay network performance, relative delay penalty deserves some more explanation. The **R**elative **D**elay **P**enalty **(RDP)** describes how well the overlay structure match the underlying network topology. It is defined as the ratio $RDP = \frac{d_{overlay}(A,B)}{d_{underlay}(A,B)}$ of the measured latency introduced by sending a message from point A to B through the overlay structure and the corresponding latency when sending it directly through the underlay [6].

In the following experiments, all the peers join and after the stabilization phase, they do appropriate overlay operations. Churn rate is mixed log-normal. Experiments were done through 20 simulation runs each. The results are presented using 95% confidence intervals.

B. Lookup Performance

In spite of the fact that Chord [7] and Kademlia [2] were designed for lookup rather than for retrievable search, the performance of their lookup operation will be used here as reference for a comparison (see Figures 4). This experiments are run with 10 000 peers, Globase.KOM is observed with $L_1 = 55$, $L_2 = 110$ and $L_1 = 38$, $L_2 = 110$, and Chord with 10 successors and stabilization interval of 650ms. In Figure 4(a) we can see that number of hops of Globase.KOM is 18% better in the case of parameters $L_1 = 55$ and $L_2 = 110$. Chord needs on average 22.8% more hops per lookup operation then Globase.KOM with $L_1 = 55$ and $L_2 = 110$ while Kademlia performs 21% better due to paralel lookup queries and big contact lists. That also reflects on operation duration as shown in Figure 4(b), where Globase.KOM needs 38.4% longer time to respond on lookup query than Kademlia. However, the lookup perfromance difference between Chord and Globase.KOM is even bigger regarding to operation duration – Globase.KOM performs 53.5% better than Chord. The reason is more underlay-aware building of the Globase.KOM overlay, which significantly reduces RDP like shown in Figure 4(c). That is the reason why both configurations of Globase.KOM have almost the same duration of lookup operation.

C. Area Search

In this experiments, Globase.KOM is observed with $L_1 = 10, 25, 55$, $L_2 = 20, 50, 110$ respectively for the experiments with 100, 1000, 10 000 peers. We have considered two cases of area search based on the distance of the searched area and the peer who initiated the area search – local and distant area search. Local area search performs better as the peer can get all results by contacting just its own superpeer or by forwarding the search request some levels higher (figures 5(a) and 5(b)). The steep decrease of the operation time during the simulation in the case of distance area search proves the significance of interconnections. They are built by learning from received messages and therefore are not existing in the beginning of the simulation. This results in longer durations for resolving search operation. Figure 5(c) shows that for distant area search with 100 peers in the worst case only 0.3% of the results are not delivered. This percentage decreases to 0.1% for 1000 peers and 0.05% for 10 000 peers. The main reason for not delivering all results in this experiments is that the Area Search Time-out is set to 2s. As we can see, there is no difference in retrievability between local and distant area search. Regarding the scalability of overlay operations, Globase.KOM scales logarithmically according to the simulation results which are not presented here because of the lack of space.

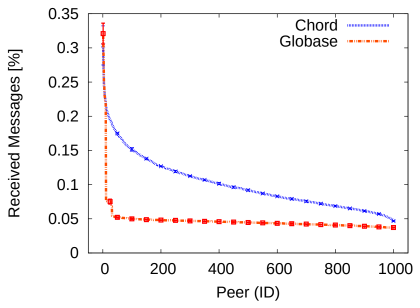


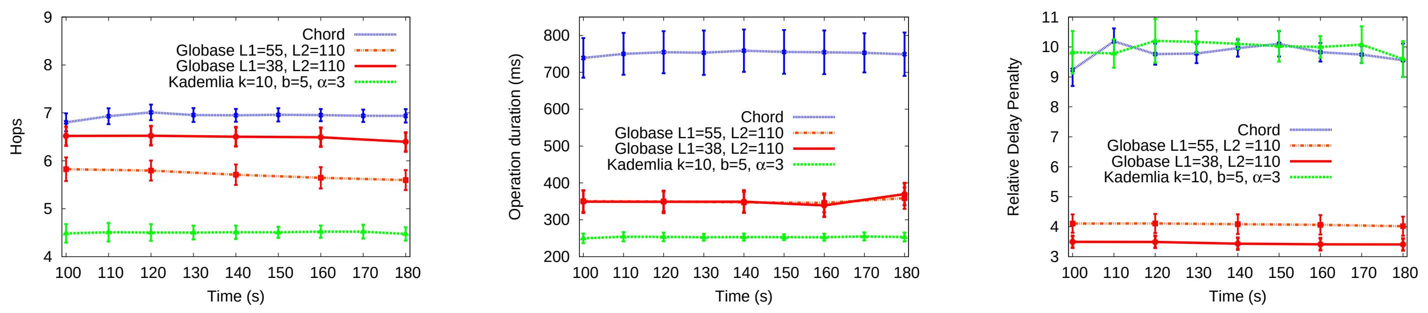
Fig. 6. Precentage of received messages of all peers in Chord and Globase.KOM

D. Load Balancing

This experiments are run with 1000 peers, Globase.KOM is observed with $L_1 = 25$, $L_2 = 50$, and Chord with 10 successors and stabilization interval of 650ms. In order to measure the load of the peers, each received message during the simulation is counted per peer. Peers are then sorted from the most to the least loaded peers in order to present the proportional distribution of messages in the overlay, like depicted in Figure 6. We can see the load distribution of Globase.KOM and Chord under the identical simulation conditions. The average load of the peers in Chord is 0.1% and vary between 0.05% and 0.31%. There are no severe differences in load distribution, though around 40 peers have significantly bigger load then other participants. The explanation is that in the beginning of the simulation, the Chord ring is built over just a few peers and therefore the most of the peers have fingers to those peers. Through stabilization messages, those peers are periodically contacted from all peers which have fingers to them. The average load in Globase.KOM is 0.05% and vary between 0.04% and 0.32%. Figure 7 shows steep reduction of the load happens after the first 10 most loaded peers. More exact insight shows that those peers are 10 superpeers which form the overlay. Bigger load of superpeers is according to expectations because of maintenance messages from children-peers as well as routing messages. The root superpeer has the highest load (0.32%), and its direct children-superpeers have around 0.26%. On average, a superpeer receives 0.21% of all produced messages in the overlay. Therefore, heterogeneity of the peers is taken into account when selecting superpeers.

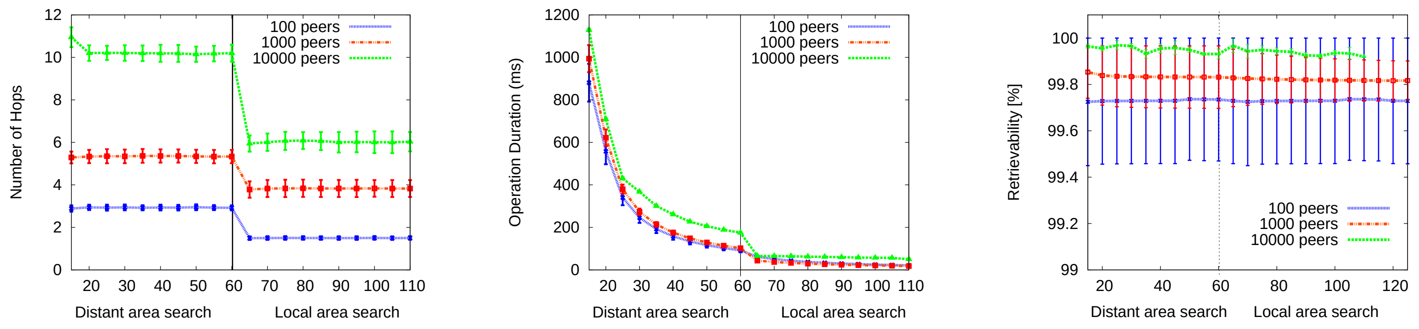
E. Protocol Overhead

This experiments are run with 2000 peers and Globase.KOM is observed with $L_1 = 25, 50, 75, 100, 125, 150, 175$ and $L_2 = 50, 100, 150, 200, 250, 300, 350$ in order to vary the number of superpeers. Figure 8 shows the number of sent user messages (resulted from lookup and area-search operations) and maintenance (keep alive, refresh) messages, with various network sizes (number of superpeers). The ratio L_1/L_2 is changed so that the overlay contains from 5 to 80 superpeers.



(a) Number of hops (b) Operation duration (c) RDP

Fig. 4. Lookup performance of Globase.KOM, Chord, and Kademlia with 10 000 peers



(a) Number of hops (b) Operation duration (c) Retrievability

Fig. 5. Performance of a local and distant area search operation in Globase.KOM

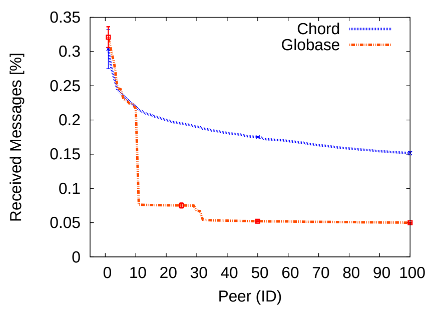


Fig. 7. Precentage of received messages of the first 100 most loaded peers in in Chord and Globase.KOM

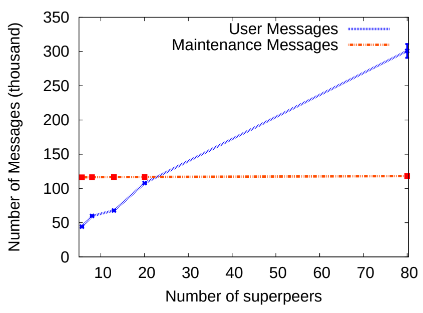


Fig. 8. Protocol Overhead

The increase of the number of superpeers in the overlay is linearly followed by an increase of number of user messages. More superpeers in the overlay means smaller zones and more superpeers involved in resolving the queries. The number of maintenance messages stays constant. The reason is that more than 97.3% of the sent maintenance messages is exchanged between peers and their responsible superpeer. The overall number M of maintenance (Keep Alive) messages per Keep Alive Interval is $M = 3 \cdot (S_p - 1) + P$ where S_p is the number of superpeers and P is the number of peers in an overlay. As optimal value for threshold L_2 in the case of network with 2000 peers, is around 100 (see Section IV-F) we can assume between 20 - 40 superpeers. In that case, the number of maintenance messages is equal or up to 50% bigger than the number of user messages.

F. Effects of Protocol Parameters on Performance

This experiments are run with 5000 peers and Globase.KOM is observed with $L_1 = 10, 20, 30, 40, 50, 60, 70, 80, 90$ and $L_2 = 100$. Here we observe the influence of the overlay parameters L_1 and L_2 on the performance of area search. Since resolving area search involves routing through the tree, first we discuss the influence of the ratio $\nu = L_1/L_2$ on the form of the superpeer tree (see Figure 9). Smaller ν makes the tree deeper but with less breadth. When ν is small, which means L_1 is considerably smaller then L_2, the threshold L_2 is reached faster, new inner zones are formed and assigned to a new superpeer sooner. However, a load balanced superpeer has very small load now and it will take a long time until it is overloaded again. On the other side, a newly assigned superpeer got high load $(L_1 - L_2)$ and will soon create a new inner zone with high probability. Therefore, the tree will grow in depth rather than in breadth. In the case ν is big, which means that the values $L1$ and $L2$ are close to each other, newly created inner zones contains a very small amount of peers. However, load balanced superpeers will reach the threshold $L2$ faster than in the previous case and therefore create more children-superpeers then newly created superpeers.

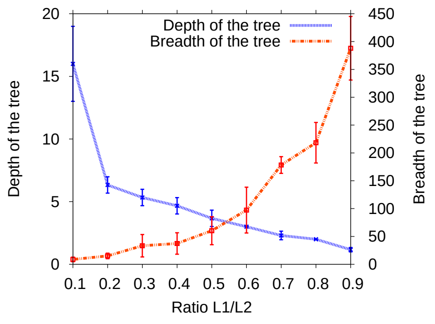


Fig. 9. Breadth and depth of the tree depending on the L1/L2 Ratio

Figure 10 shows the impact of ν on the number of hops and duration of search operation. With an increase of ν and thus an decrease of depth of the tree, the overlay needs less hops to resolve the search queries – it decreases from 8.5 to 4.5 hops. The operation duration increases slightly due to increase of relative delay penalty.

Figure 11 shows the average load distribution of the 50 most loaded peers for different ν. In a broader tree (with bigger ν) the most of the communication goes through superpeers which are placed higher in the tree hierarchy and thus their load is significantly higher than in the case of deeper tree (smaller ν).

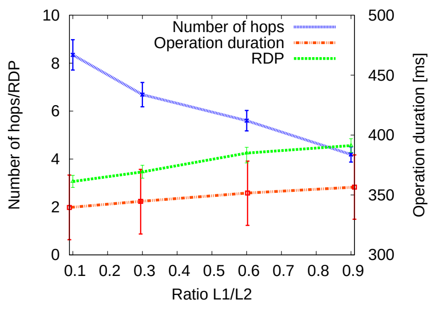


Fig. 10. Influence of the L1/L2 ratio on number of hops, duration and relative delay penalty of area search operations

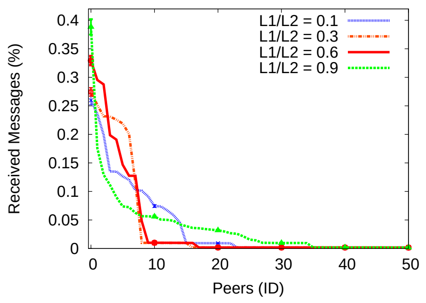


Fig. 11. Load balancing depending on the L1/L2 Ratio

V. Related work

So far, location-based search in P2P networks is mainly approached by re-using existing structured overlays that are used to provide efficient one dimensional lookups [8] [9]. The linearization of two-dimensional map projections is achieved using different space filling curves. The suitability of different space-filling curves is discussed in [10]. The focus of [8] in developing Prefix Hash Trees (PHTs) was to meet the needs of PlaceLab's [11], end-user positioning system, without modifying the underlying DHT. It is able to perform two dimensional geographical range queries applying a Z-curve linearization of the 2D space. All approaches with space-filling curves suffer from not matching the geographical distance with the distance in the overlay ID space. That results in inefficient query replies which introduce additional delay into the communication. Another important point is that most of these are using DHTs which are not providing a guarantee for complete retrieval of all results matching a search request. Search for spatial content was the focus of [12] and [13]. In [12], Harwood and Tanin are recursively dividing a 2D space into smaller zones and using a distributed quadtree index for assigning responsibilities for regions of space to peers. For each zone a control point is assigned and hashed into the node ID on the Chord ring. Copies of the objects associated with a region are stored on the node which was assigned the control point. As a result, the 2D space is transformed to a tree structure. Zimmermann et al. discuss in [14] and [15] that such an approach can lead to load balancing problems and therefore they introduced a mapping of the physical space into the CAN [16] overlay instead of Chord [7]. Identification of spatial data is created with a concatenation of the respective location, a random part, and the identification of the content of object. Similar work is done in [17] based on K-D trees [18]. The search space is repeatedly hierarchically partitioned into smaller zones and each internal node splits its zone into two subzones. The data points are stored in leaf nodes. This solution creates a performance bottleneck at the higher level nodes since a query has to be propagated to the nodes close to root of the tree. A binary tree as a distributed space partitioning tree is used in RectNet [19]. It dynamically adapts to the geographical distribution of the workload caused by the storage of (location, object)-pairs and the processing of queries. This tree has a binary structure which simplifies the recovery of the structure in the case of node failures and significantly reduces the search performance. All of these approaches do not consider explicit load balancing when creating/updating the zone assignment. Another problem is the recovery which is hindered by keeping the zone rectangular so that zones cannot be simply concatenated if the resulting zone is not rectangular. GeoPeer is location-aware peer-to-peer system [20] that is using Delaunay triangulation to build a connected lattice of nodes and it implements a DHT for geographical routing, similar to GHT [21]. The focus of both systems is to lookup and to route. Both do not provide support for complete retrievable search.

Summarizing the related work there are many works in the area of location-based search with many different advantages and disadvantages. To our best knowledge a solution has not been developed yet that meets our requirements (section II).

VI. Conclusion and Future Work

In this paper we research the challenge of fully retrievable location-based search following the peer-to-peer paradigm in order to overcome the problems of existing solutions, i.e. to provide a mechanisms to retrieve all up-to-date information related to an area. We presented overlay structure and operations as well as failure recovery mechanisms of Globase.KOM, a superpeer tree-based overlay. Simulation results proved that Globase.KOM provides full retrievability of area searches, a high degree of underlay topology awareness, short response time, and logarithmical scalability. The load of the peers is just slightly worse balanced than it is the case in flat structures, such as Chord. We showed how the overlay parameters influence the performance of the overlay operations. Future work will be focused on improving the interconnection strategy in order to avoid the effects of an unbalanced, degenerated tree. Additionally, we will focus on developing a tree with multiple roots in order to decrease the load of the superpeers in the higher levels.

Acknowledgment

The authors would like to thank Oliver Heckmann for his great help with initial ideas, André Mink for his support in evaluation, and Patrick Mukherjee on valuable discussions and comments.

References

[1] M. K. Reiter, A. Samar, and C. Wang. Distributed Construction of a Fault-Tolerant Network from a Tree. In *SRDS '05: Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 155–165, Washington, DC, USA, 2005. IEEE Computer Society.

[2] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. `http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.ps`.

[3] H. Garcia-Molina. Elections in a Distributed Computing System. *IEEE Transactions on Computers*, C-31:48–59, 1982.

[4] PeerfactSim.KOM: A Simulator for Large-Scale Peer-to-Peer Networks. `http://www.peerfactsim.com`.

[5] Internet users by country world map. `http://upload.wikimedia.org/wikipedia/commons/7/7f/Internet_users_by_country_world_map.PNG`.

[6] S. Jain, R.l Mahajan, and D. Wetherall. A Study of the Performance Potential of DHT-based Overlays. In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[7] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[8] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, S. Shenker, and J. Hellerstein. A case study in building layered DHT applications. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 97–108, New York, NY, USA, 2005. ACM Press.

[9] S. Zhou, G. Ganger, and P. Steenkiste. Location-based Node IDs: Enabling Explicit Locality in DHTs. Technical Report CMU-CS-03-171, Carnegie Mellon University, September 2003 2003.

[10] M. Knoll and T. Weis. Optimizing Locality for Self-Organizing Context-based Systems. In *International Workshop on Self-Organizing Systems (IWSOS 2006)*, September 18-20 2006.

[11] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place Lab: Device Positioning Using Radio Beacons in the Wild. In *Proceedings of PERVASIVE 2005*, Munich, Germany, 2005.

[12] A. Harwood and E. Tanin. Hashing Spatial Content over Peer-to-Peer Networks. In *Australian Telecommunications, Networks and Applications Conference*, page 5. ATNAC, 2003.

[13] A. Mondal, Y. Lifu, and M. Kitsuregawa. P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments. In *EDBT Workshops*, pages 516–525, 2004.

[14] R. Zimmermann, W.-S. Ku, and H. Wang. Spatial Data Query Support in Peer-to-Peer Systems. In *Proc. of COMPSAC '04*, pages 82–85, Washington, DC, USA, 2004. IEEE Computer Society.

[15] H. Wang, R. Zimmermann, and W.-S. Ku. ASPEN: an adaptive spatial peer-to-peer network. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 230–239, New York, NY, USA, 2005. ACM Press.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. of SIGCOMM '01*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[17] C. Zhang, A. Krishnamurthy, and R. Y. Wang. Brushwood: Distributed Trees in Peer-to-Peer Systems. In *4th International Workshop on Peer-To-Peer Systems*, pages 47–57, Ithaca, New York, USA, February 2005.

[18] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[19] D. Heutelbeck. *Distributed Space Partitioning Trees and their Application in Mobile Computing*. PhD thesis, Fernuniversität Hagen, Hagen, Germany, 2005.

[20] F. Araújo and L. Rodrigues. GeoPeer: A Location-Aware Peer-to-Peer System. In *Proc. of NCA '04*, pages 39–46, Washington, DC, USA, 2004. IEEE Computer Society.

[21] S. Ratnasamy, B. Karp, L. Yin, F. Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: A geographic hash table for data-centric storage. In *Proc of WSNA*, Atlanta, GA, September 2002.

[22] O. Heckmann, M. G. Sanchis, A. Kovačević, N. Liebau, and R. Steinmetz. A Peer-to-Peer System for Location-based Services. In *Proc. of Peer-to-Peer Paradigm (PTPP) Track at AMCIS*, August 2006.