KLS07] Aleksandra Kovacevic, Nicolas Liebau, Ralf Steinmetz, **Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search**, Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing, September 2007, S. 82 - 94

Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search

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

Abstract

Location based services are becoming increasingly popular as devices that determine geographical position become more available to end users. The main problem of existing solutions to location-based search is keeping information updated requires centralized maintenance at specific times. Therefore, retrieved results do not include all objects that exist in reality. A peer-to-peer (P2P) 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 P2P overlay that enables fully retrievable location-based overlay operations which proved to be highly efficient and logarithmically scalable.

1 Introduction

Location-based search is becoming increasingly popular and it is a part of many search engines and navigation systems. In existing centrally managed solutions, the search results are often incomplete or outdated. Additional information about the searched object (e.g. opening hours, prices, or menu) is usually not available as such a huge amount of data and frequent updates (e.g. number of free places in restaurant) would overload the server. In a P2P solution, each peer is responsible for the information about the object it represents, therefore updating and publishing information is done directly, bypassing the server and avoiding single point of failure. 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 at low cost, because of the natural scalability and administration-free character of P2P systems, which makes them available to a wide community to join and publish their services. While the P2P research community has been very active in 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 P2P overlay that enables fully retrievable area search, lookup, and finding the geographically closest node.

The paper is organized as follows: In Section 2 we set the goals, requirements, and assumption for our solution. In Section 3 we present Globase.KOM. Performance of overlay operations, as well as protocol overhead, load balance, and effects of overlay parameters on performance are shown in Section 4. We discuss related work in Section 5 and conclude the paper in Section 6.

2 Goal, Requirements, and Assumptions

Our main focus is enabling search over *all* peers in some defined geographical area. The area can be circular or rectangular. Additionally, a peer can search 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. In this paper, we do not consider mobile peers as we assume that users search for static objects - gasoline stations, restaurants etc. However, Globase.KOM can support mobility of the peers to some extent.

We assume that each peer is aware of its exact geographical position (using appropriate devices or database). In order to represent the two-dimensional curved surface of the Earth on a plane, we use the Plate Carée projection. This projection plots 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 introduce some kind of distortion because an ellipsoid can not be mapped to a plane without stretching,

tearing, or shrinking. 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. When the search area lies 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.

3 Design

This section describes our solution – a P2P overlay for *fully retrievable* **G**eographical **LO**cation **BA**sed **SE**arch (Globase.KOM). We describe its overlay structure, overlay operations, and failure recovery.

3.1 Overlay Structure

Globase.KOM is a superpeer-based overlay forming a tree enhanced with interconnections. The world projection is divided in rectangular, non-overlapping zones (Figure 1). Each zone is assigned to a superpeer located inside of the zone which keeps overlay/underlay contact addresses to all peers in that zone. Superpeers form a tree where peer A is called the parent of peer B when B's zone is inside A's zone.

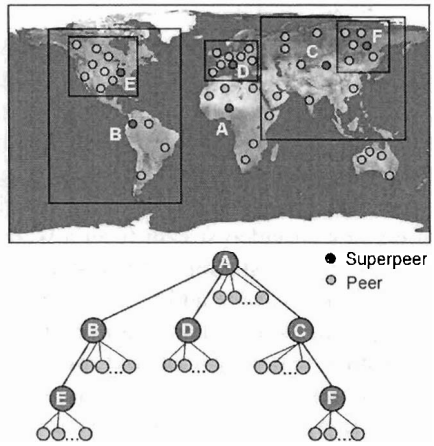


Figure 1. Basic structure of the Globase.KOM

Each superpeer maintains the contact addresses of: 1) the peers inside of its zone, excluding the inner zones, 2) superpeers responsible for inner zones (i.e. child nodes), 3) its parent in the tree, 4) the root superpeer, and 5) interconnected superpeers (see 3.1.2).

Each peer maintains the following contact addresses: 1) the parent superpeer, 2) the root superpeer, 3) an interconnection list, and 4) a cache list of already contacted peers.

Peers/superpeers are identified by their unique ID (Figure 2). The PeerID contains: 1) the GPS coordinate of the node, 2) if it is a supernode, the zone it is responsible for, and 3) a random part in order to support the existence of more than one peer at the same location. A rectangular zone is simply described by the concatenation of its vector representation - left bottom point of the zone and its side lengths.

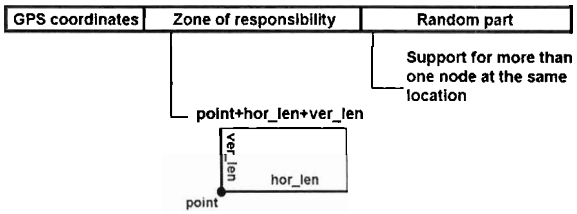


Figure 2. Structure of PeerID

3.1.1 Forming the Zones

When bootstrapping the system, 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 superpeers. As the network grows, highly loaded areas are clustered into rectangular zones using a clustering algorithm and assigned to one peer in that area which becomes a superpeer by taking over the zone. As metric for the load of an area we use the number of peers connected to a 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* (below 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 runs 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 in the formed zone that is marked as a potential superpeer.

3.1.2 Interconnections

As mentioned above, each superpeer maintains connections to other superpeers besides its parent and children. Also, each peer caches the contact information of other superpeers besides its parent. The main purpose of these so-called 'interconnections' is to provide fault-tolerance. Additionally, bypassing the root superpeer makes query responses more efficient especially in the case of a degenerated tree. Reiter [17] presented an algorithm for constructing a fault-tolerant communication structure out of a core tree structure where each node initially only knows 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. Tree reconstruction after

failures is done by using new edges and heavily relies on the root of the tree. Our approach modifies Reiter's approach 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 (with similar maintenance as in [13]). 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 adds the sender to its interconnection list. The size of an interconnection list allows at least one contact per subtree. Interconnections provide for each peer a rough view of the tree structure in order to optimize tree recovery actions and searches. They are most valuable when the root superpeer fails because they can recover peers from the affected zone (Section 3.3).

3.2 Overlay Operations

Besides joining and leaving the network, peers in Globase.KOM can do area searches (3.2.2), lookups (3.2.1), or finding the geographically closest node (3.2.3).

3.2.1 Lookup

In our case, the lookup operation is used to determine the underlay address (IP address and port) of a peer from its geographical location. Each superpeer knows the IDs/locations of all nodes it is responsible for. Therefore, 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 by sending 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 it is inside of the zone of one of its children, the superpeer will forward the request to the 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 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 contact address of the node if it exists or *null* if it does not exist.

3.2.2 Area Search

Area search is performed using the SEARCH message which includes a description of the geographical area (center and radius) plus metadata describing the targeted service/object. When a superpeer receives a SEARCH query from one of its peers, it calculates the searched ellipse onto the map projection. Next, it checks if that ellipse intersects the zone it is responsible for. All further actions are the same as 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 peers. The search is considered finished after a specific timeout. Simulation studies showed that the optimal value for this timeout is 2 seconds. For the each received message, interconnections are updated as described in Section 3.1.2.

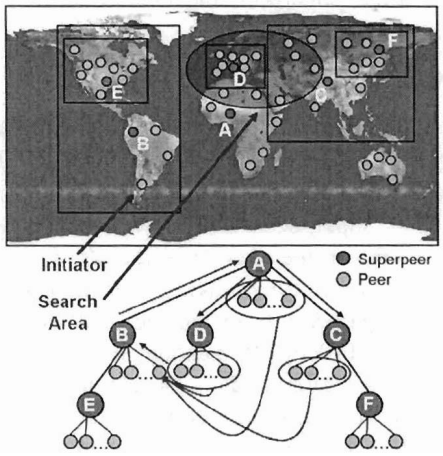


Figure 3. Example of an area search

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. In the end, superpeers A, C, and D will reply with the list of the matching results.

3.2.3 Find the Closest Peer

When a peer wants to find the closest peer, it first calculates the closest border of the zone it belongs to. This is possible by using the ID of the parent superpeer, which contains a vector representation of the zone. Then, the peer sends a 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 the radius of the given distance, the superpeer calculates the closest and includes it in a FIND_CLOSEST_RESULT message. Otherwise, it sends back FIND_CLOSEST_NEXT

message which includes the address of its parent superpeer. The peer again calculates the closest border of the zone of the retrieved superpeer and send it the newly formed FIND_CLOSEST message. The steps are then repeated until the peer receives a FIND_CLOSEST_RESULT message containing the contact of the closest peer.

3.2.4 Join

When a peer wants to join, his bootstrapping superpeer routes a NEW_PEER message in order to find the zone the peer is located in. In the case that the peer is first to join, the process is described in Section 3.1.1. The found superpeer forms the peerID for the new peer, adds it to its peer list, marking it appropriately if it is a potential superpeer. It sends back a PEER_OK message with the formed PeerID, the contact of root superpeer, and an interconnection.

3.2.5 Leave

When leaving the network, a superpeer has to inform its parent superpeer by sending a REMOVE_SUPERPEER message with all the contacts it maintains - the list of the peers, children superpeers, and interconnection lists. The informed superpeer then removes the leaving superpeer from the list of its children, taking over the responsibility for all peers/superpeers from the received lists. It sends a REFRESH_SUPERPEER message to its new children peers/superpeers. Afterwards, it checks its load; if it exceeds L_2, it forms a new zone as described in 3.1.1. In the case where 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.

3.3 Failure Recovery

Failure of a peer/superpeer in Globase.KOM is detected when an appropriate peer/superpeer misses a sequence of 3 KEEP_ALIVE messages. In the case of a peer, its superpeer will detect the failure and simply remove the contact address of the failed peer. Here we will discuss more in detail failure recovery of a superpeer and the root superpeer. The measured introduced overhead is shown in Section 4.5.

3.3.1 Failure of a Superpeer

Each superpeer sends a KEEP_ALIVE message periodically 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 a KILL message to the failed superpeer and the parent superpeer removes the 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 a TAKE_ME message to the interconnected superpeer, otherwise the message is sent to the root superpeer. The receiver of a TAKE_ME message routes the appropriate message to the smallest zone containing the location of the sender. The reached superpeer (most likely the parent of the failed superpeer, if it is still online) adds the peer to its peer list or sets the superpeer as a new child superpeer and sends in both cases a REFRESH_SUPERPEER message to it. This mechanism also works well in the case when multiple superpeers fail simultaneously, as the peers contact either interconnected superpeers or the root superpeer for recovery. If none of its contacts are alive, the superpeer/peer will simply rejoin.

3.3.2 Failure of the Root Superpeer

In the case where the root superpeer fails, one of the children superpeers takes over the responsibility of the failed root superpeer. In order to avoid forming several independent trees, we apply the *Election on Bully* algorithm [6]. Therefore, all children superpeers of the root superpeer keep connections to each other. The contacts are kept refreshed through a REFRESH_BROTHERS message from the root superpeer. Using the election algorithm, when a child superpeer notices the failure of the 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, the elected superpeer takes over the zone of the root superpeer and sends REFRESH_ROOT messages to all superpeers.

4 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 by observing overlay operations - lookup and area search with a focus on retrievability. An additional gain of Globase.KOM is better underlying topology awareness, reflected by 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 are included in evaluation as well.

4.1 Evaluation Setup

Simulation is the most appropriate evaluation method for large-scale P2P overlay networks. We used PeerfactSim

[2], a simulator for P2P 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, congestion, retransmission, and packet loss. 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 [1].

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 commonly used metrics for evaluation of P2P overlay network performance, relative delay penalty deserves some more explanation. The **R**elative **D**elay **P**enalty **(RDP)** describes how well the overlay structure matches 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 [10].

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

4.2 Lookup Performance

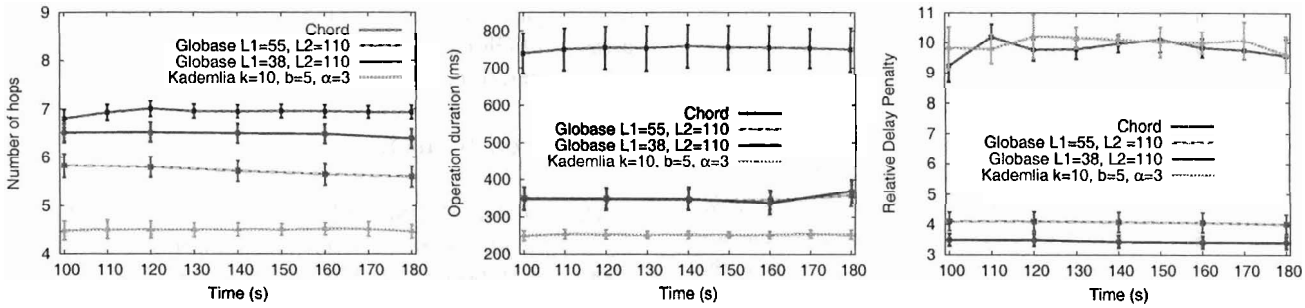
In spite of the fact that Chord [18] and Kademlia [13] were designed for lookup rather than for retrievable search, the performance of their modified lookup operation will be used here as reference for a comparison (see Figures 4). The experiments are run with 10 000 peers, and parameters are $L_1 = 55$, $L_2 = 110$ and $L_1 = 38$, $L_2 = 110$, 10 successors in Chord with stabilization interval of 650ms, and $k = 10$, $b = 5$, and $\alpha = 3$ for Kademlia. In Figure 4(a) we can see that number of hops in 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. This also reflects on operation duration (Figure 4(b)), where Globase.KOM needs 38.4% longer time to respond to a lookup than Kademlia. However, the lookup performance difference between Chord and Globase.KOM is even bigger with regard to operation duration – Globase.KOM performs 53.5% better than Chord. The reason is better underlay-awareness of the Globase.KOM overlay, which significantly reduces RDP (Figure 4(c)). That is also the reason why both configurations of Globase.KOM have almost the same duration of lookup operation.

4.3 Area Search

Here Globase.KOM is observed with $L_1 = 10, 25, 55$, $L_2 = 20, 50, 110$ respectively for experiments with 100, 1000, and 10 000 peers. We have considered two cases of area search based on the distance of the searched area from the peer who initiated the area search – local (from 60s to 110s in simulation scenario) and distant area search (from 0s to 60s). Local area search performs better as the peer often can get all results just by contacting its own superpeer (Figures 5(a) and 5(b)). The steep decrease of the operation time during the simulation of distant area search proves the significance of interconnections, which are built by learning from received messages and therefore do not exist at the beginning of the simulation, resulting in longer durations for search operations. 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 these experiments is the time-out, set to 2s. As we can see, there is no difference in retrievability between local and distant area search. Globase.KOM scales logarithmically for overlay operations according to the simulation results which are not presented here due to the lack of space.

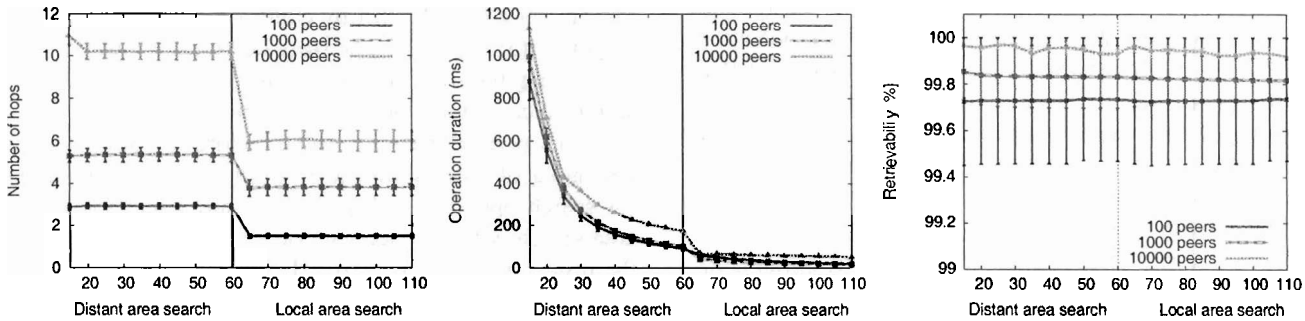
4.4 Load Balancing

These experiments are run with 1000 peers, $L_1 = 25$, $L_2 = 50$, and 10 successors in Chord with stabilization interval of 650ms. In order to measure the load of the peers, each received message per peer is counted during the simulation. Peers are then sorted from the most to the least loaded peer to present the proportional distribution of messages in the overlay (Figure 7(a)). 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 varies between 0.05% and 0.31%. There are no severe differences in load distribution, though around 40 peers have significantly bigger load than 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 varies between 0.04% and 0.32%. Figure 7(b) shows steep load reduction after the first 10 most loaded peers. More exact insight shows that those peers are 10 superpeers



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

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



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

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

which form the overlay. The bigger load of superpeers is 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. Thus, heterogeneity of the peers is taken into account when selecting superpeers.

4.5 Protocol Overhead

These experiments are run with 2000 peers, parameters $L_1 = 25, 50, 75, 100, 125, 150, 175$ and $L_2 = 50, 100, 150, 200, 250, 300, 350$, which vary the number of superpeers. Figure 6 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. 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 are exchanged between peers and their responsible superpeer. The overall number M of maintenance

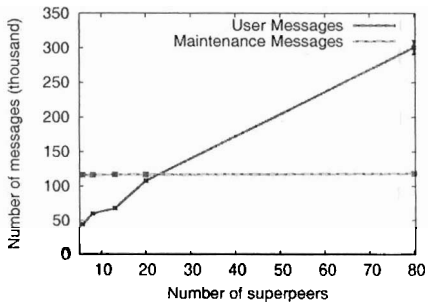
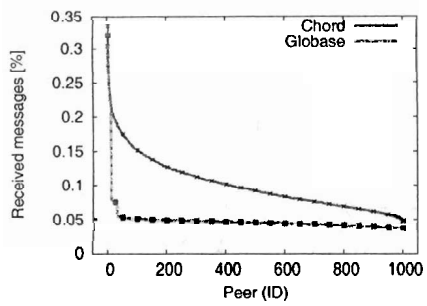


Figure 6. Protocol Overhead

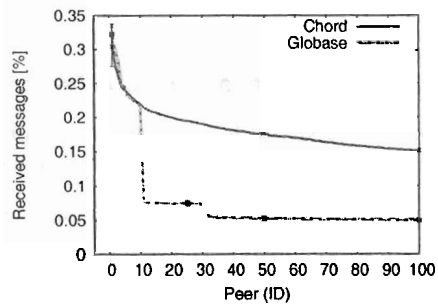
messages per keep-alive-interval is $M = 3 \cdot (S_p - 1) + P$ where S_p is the number of superpeers and P is the number of peers in overlay. As an optimal value for the threshold L_2 in the case of a network with 2000 peers, is around 100 (see Section 4.6), we can assume between 20 - 40 superpeers. In that case, the number of maintenance messages is equal or up to 50% bigger than the number of user messages.

4.6 Effects of Protocol Parameters on Performance

These experiments are run with 5000 peers and parameters $L_1 = 10, 20, 30, 40, 50, 60, 70, 80, 90$ and $L_2 = 100$. Here we observe the influence of the overlay parameters L_1



(a) For all peers



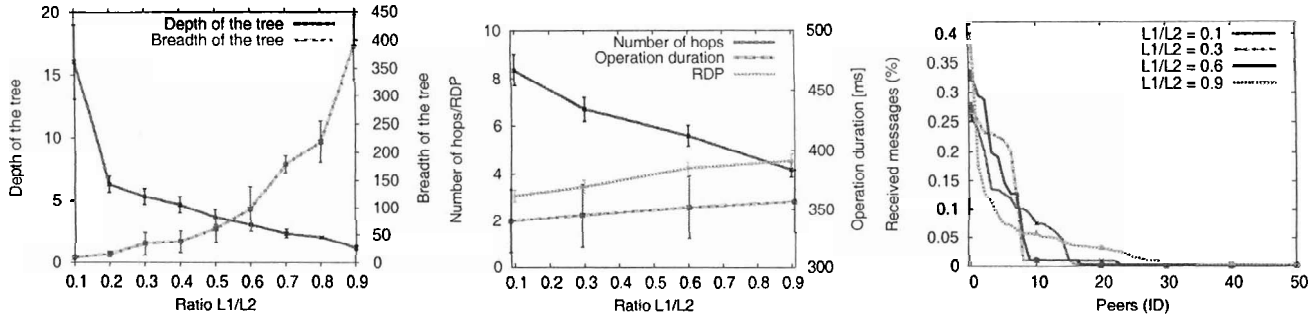
(b) For the first 100 most loaded peers

Figure 7. Percentage of received messages

and L_2 on the performance of area search. Since resolving an 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 8(a)). Smaller ν makes the tree deeper, but with less breadth. When ν is small, which means L_1 is considerably smaller than 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 hand, the newly assigned superpeer got a high load ($L_1 - L_2$) and will likely soon create a new inner zone. Therefore, the tree will grow in depth rather than in breadth. In the case where ν is big, which means that the values $L1$ and $L2$ are close to each other, newly created inner zones contain 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 than newly created superpeers. Figure 8(b) shows the impact of ν on the number of hops and duration of search. 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 8(c) 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 higher in the tree hierarchy and thus their load is significantly higher than in the case of deeper tree (smaller ν).

5 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 [5] [21]. The linearization of two-dimensional map projections is achieved using different space filling curves. The suitability of different space-filling curves is discussed in [12]. The focus of [5] in developing Prefix Hash Trees (PHTs) was to meet the needs of an end-user positioning system, without modifying the underlying DHT. It is able to perform two dimensional geographical range queries by 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. This results in inefficient query replies which introduce additional delay into the communication. Another important point is that most of these are using DHTs which do not provide complete retrievability of a search request. Search for spatial content was the focus of [7] and [14]. In [7], Harwood and Tanin recursively divide a 2D space into smaller zones and using a distributed quadtree index to assign 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 [19] that such an approach can lead to load balancing problems and therefore they introduced a mapping of the physical space into the CAN [15] overlay instead of Chord [18]. Identification of spatial data is created by a concatenation of the respective location, a random part, and the identification of the content of object. Similar work is done in [20] based on K-D trees [4]. 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. LL-Net [11] uses the same zone division and assigns an R-Peer to each area to be the root of tree topology formed of N-Peers contained in that area. Besides the zone division and routing, this work differs from our approach by using a central instance, the S-Peer, which manages contacts of all R-Peers, bringing all drawbacks of central management. A binary tree as a distributed space partitioning tree is used in RectNet [9]. 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, but significantly reduces the search performance. GeoPeer is a location-aware P2P system [3]



(a) Breadth and depth of the tree

(b) Number of hops, duration, and relative delay penalty of area search operations

(c) Load balancing

Figure 8. Influence of the L1/L2 ratio on the form of the tree, performance, and load balancing

that is using Delaunay triangulation to build a connected lattice of nodes and it implements a DHT for geographical routing, similar to GHT [16]. Neither provide support for complete retrievable search.

6 Conclusion and Future Work

In this paper we address the challenge of fully retrievable location-based search using the P2P paradigm in order to overcome the problems of existing solutions, i.e. retrieval of all up-to-date information related to an area. We presented the overlay structure, operations and failure recovery mechanisms of Globase.KOM, a superpeer tree-based overlay. Simulation results proved 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 less balanced than it is the case in flat structures, such as Chord. Future work will be focused on improving the interconnection strategies and developing a tree with multiple roots in order to decrease the load of the superpeers in the higher levels.

References

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

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

[3] F. Araújo and L. Rodrigues. GeoPeer: A Location-Aware Peer-to-Peer System. In *Proc. of NCA '04*, 2004.

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

[5] Y. Chawathe et al. A case study in building layered DHT applications. In *SIGCOMM '05*, pages 97–108, 2005.

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

[7] A. Harwood and E. Tanin. Hashing Spatial Content over Peer-to-Peer Networks. In *ATNAC*, page 5, 2003.

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

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

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

[11] Y. Kaneko et al. A location-based peer-to-peer network for context-aware services in a ubiquitous environment. In *SAINT-W'05*, 2005.

[12] M. Knoll and T. Weis. Optimizing Locality for Self-Organizing Context-based Systems. In *IWSOS'06*, 2006.

[13] 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`.

[14] 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.

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

[16] S. Ratnasamy et al. GHT: A geographic hash table for data-centric storage. In *Proc of WSNA*, Atlanta, GA, Sept. 2002.

[17] M. K. Reiter, A. Samar, and C. Wang. Distributed Construction of a Fault-Tolerant Network from a Tree. In *SRDS '05*, pages 155–165, Washington, DC, USA, 2005.

[18] I. Stoica et al. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.

[19] H. Wang, R. Zimmermann, and W.-S. Ku. ASPEN: an adaptive spatial peer-to-peer network. In *GIS '05*, 2005.

[20] C. Zhang, A. Krishnamurthy, and R. Y. Wang. Brushwood: Distributed Trees in Peer-to-Peer Systems. In *4th Int. Workshop on P2P Systems*, 2005.

[21] 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, 2003.