Moving Peers in Distributed, Location-based Peer-to-Peer Overlays

Tobias Amft and Kalman Graffi
Technology of Social Networks Group, Heinrich-Heine University Düsseldorf, Germany
Email: [amft, graffi]@cs.uni-duesseldorf.de

***Abstract*—In the past decades, various concepts of peer-to-peer (P2P) systems have been realized in desktop applications for stationary PCs. Last recent years show that computers have evolved to handy devices which can be used nearly everywhere, not only at home. For this reason, location-based services which allow to associate peers in a network with geographical positions, e.g. to track a device in emergency cases, become interesting. In this paper, we introduce *LobSter*, a lightweight location-based service over unmodified DHTs like Chord [24], Kademlia [14] and Pastry [21] which supports the mobility of peers. Our evaluation compares *LobSter* to related work in stationary situations and gives insights about the general limits of distributed location-based services in mobile environments.**

I. Introduction

Since most mobile devices include electronics and software to determine the geographical position of the device, location-based services are interesting nowadays as they enable a user to find peers in a certain area around a location. Older works about location-based P2P overlays mostly assume stationary nodes which are not able to move. Reason for this observation might be the fact that mobility became important in the last years when mobile devices emerged. In the beginning of the era of research on P2P systems (around 2000 - 2007), computers were rather fixed to one location. The recent emergence of mobile devices like tablet computers and smartphones make location-based overlays for mobile peers interesting again.

In this paper, we describe and evaluate *LobSter*, a DHT-based approach of location-based overlays which can be applied on top of every DHT-overlay which implements few simple methods described by the DHT- and KBR-API [4]. Its purpose is to find as many peers as possible located in a geographical area through *range-queries*. In our evaluation, we compare LobSter to Geodemlia [6], another location-based overlay which supports range-queries, in stationary scenarios.

Our contribution in this paper is twofold: first, we show with LobSter that simple location-based overlays which use linearization techniques, like space filling curves, on top of existing DHTs can compete with more specialized approaches for location-based services, such as Geodemlia, although they do not utilize own, dedicated routing tables or maintenance strategies. Second, we focus in our evaluation on a realistic scenario in which peers are moving and show, using LobSter as example, that range-queries under peer movement are generally limited by the size of the searched area.

A. Outline

The remainder of the paper is structured as follows: in Section II we list existing work related to location-based services and briefly discuss their shortcomings. Section III describes how range-queries are conducted in LobSter. We evaluate LobSter in Section IV and show general limits of location-based P2P overlays. Section V concludes our investigation.

II. Related Work on Location-based Overlays

We divide location-based overlays presented over the last years mainly into three categories: dedicated overlays, mappings from a multi-dimensional space into a one dimensional DHT, and geographically structured overlays.

In the first category of solutions, new overlays have been introduced which fulfill the purpose of location-based services by storing links to geographical neighbors. Approaches such as CAN [20] and HyperCuP [22] store less contacts than general DHTs, thus stability under churn and scalability are endangered (routing mostly in $O(\log(N))$ with N being size of the network). Overlays in the category of dedicated location-based overlays can be distinguished in hierarchical and flat approaches. RectNet [7], Globase.KOM [10], Geo-Chord [17] and GeoP2P [2] build hierarchical structures as they recursively split the geographical area into a tree structure. The algorithms mainly differ in the forming of the geographical zones. The connectivity of the nodes is often enhanced through wide interconnections for better performance and through high interconnections for more robustness in case of peer failures. GeoKad [18] and Geodemlia [6] are flat approaches, inspired by the idea of the DHT Kademlia [14]. Similar to [3] and [19] they use routing tables with buckets representing geographical distances to other nodes. Since dedicated overlays only serve the purpose of location-based routing, they are very efficient.

The second category of location-based overlays comprises several linearization approaches, such as [12], [15], [25], [28], [29]. In these works, space filling curves [11], [9], such as S-, Z-, Hilbert or Lebesque-curves, are used to map multi-dimensional points in a geographical representation to a single key which is then maintained in a DHT. Our solution falls into this category of overlays and considers peer movement. Although Lee et al. also assume peers to be moving in [11], they do not describe and evaluate the performance of range-queries in their work.

In the third category of solutions, new more fitting multi-dimensional data structures like [1], [23] are used which support geographical constellations without reducing them to one dimension. One main fraction of these new overlays uses a recursively split (two-dimensional) space-partitioning tree in which nodes maintain geographical zones. Approaches such as [26], use a DHT to map zones to a DHT and thus find a responsible node for a zone. One drawback of these tree-based solutions is their poor behavior under churn. Grid-based approaches, such as [8], organize the nodes' responsibilities in a multi-dimensional data structure on top of a DHT.

Excluding the work of Lee et al. [11], the overlays described in this section are not designed to handle changes in its peer's positions. An investigation under peer movement would require drastic changes in most protocols. For this reason, we focus in this paper on our solution LobSter, which is of category two.

A. Geodemlia and Other Dedicated Approaches using Buckets

In this paper we investigate location-based overlays and their ability to support application scenarios in which peers (e.g. smartphones) are moving. As smartphones and other mobile devices are usually limited in their capacities like CPU, bandwidth, etc. we are mostly interested in approaches in which participating peers are equally utilized. Hierarchical and tree-based approaches require some peers to have special roles, e.g. being root node for a geographical location. For this reason, we limit our evaluation to a solution of category two, to which our approach LobSter belongs, and to flat overlays from category one.

We chose Geodemlia [6] as representative for flat overlays from the first category to demonstrate advantages and disadvantages of such overlays. In Geodemlia, peers split their routing table into n equal parts, where each of the n parts contains neighboring peers that are located in the same direction in relation to a peer's position. Using $n = 4$ for example separates a routing table into the four cardinal directions. For each direction, neighboring peers are associated to buckets which have limited size. Each bucket in Geodemlia represents the peer's geographical distance to its neighbors.

Range-queries in Geodemlia are conducted the following: a peer p contacts the α nearest nodes to the queried area. The contacted peers return their α closest contacts to the area. Peer p then contacts all newly found peers which also return their α closest contacts. This procedure is repeated until a peer has visited all discovered neighbors near the queried area.

Our evaluation in Section IV shows that range-queries in bucket-based approaches perform worse, the smaller α (the number of parallel contacted neighbors) gets. Since buckets are limited in their size, a peer is not able to store nearest contacts for all possible locations that could be queried. Therefore, if the size of a network increases, a peer has to decide which contacts to keep in its buckets. It might happen then, that a peer p which is close to peer q maintains q in one of its buckets, but not vice versa. The resulting problem is that range-queries and queries for the k-closest peers can not be solved deterministically. Section IV shows that, even if peers are stationary, range-queries in Geodemlia lack to find all peers located in an area.

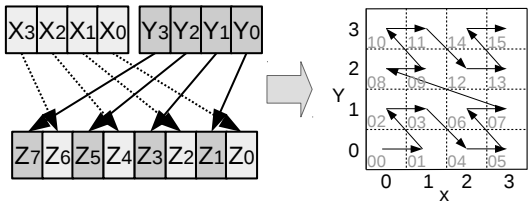


Fig. 1. The one-dimensional z-key of a peer is obtained by interleaving the bits of its coordinates. Example: (1,3) results in 11.

III. LobSter: location-based Overlay Design

In this section we introduce our approach of a distributed location-aware service which can potentially be used on top of every existing DHT-overlay. Purpose of our service is to find as many peers as possible which are located in an area of interest, e.g. to find all shops in a city. Dedicated solutions of distributed location-based services often suffer from having too small routing tables or buckets to execute range-queries deterministically and reliably.

A. DHT as a Basis for LobSter

LobSter is a service directly built on top of a DHT, therefore, each peer in LobSter is participating in a common underlying DHT like Chord or Pastry. The advantage of a common DHT as basis is that it already provides routing algorithm, lookup methods and storage mechanism that can be leveraged to retrieve information (*meta-data*) about other peers are in the network in $O(\log(N))$ steps with N being the number of nodes in the system. LobSter can be put on top of every structured overlay which supports the DHT- and KBR-API described in [4]. Changing the underlying DHT influences the behavior and performance of LobSter, using Chord for example would lead to a resilient system which has high maintenance costs, whereas Pastry as basis would result in a less robust system which uses a lazy maintenance strategy.

B. Using Z-order Curves or Other Space Filling Curves

In LobSter, each node is associated with an overlay identifier (*o-id*) which is obtained from the underlying DHT and Cartesian coordinates which represent the peer's geographical location. Peers use a z-order curve, introduced by Herzog and Tropf in [27], to calculate a one-dimensional key out of their location which is presented as multi-dimensional point. In specific, the one-dimensional key (*z-key*) is obtained by interleaving the bits of the x-coordinate and the bits of the y-coordinate of the peer's location (Fig. 1). Doing this, all peers can be associated with a one-dimensional identifier, and, at the same time keep information about their geographical location.

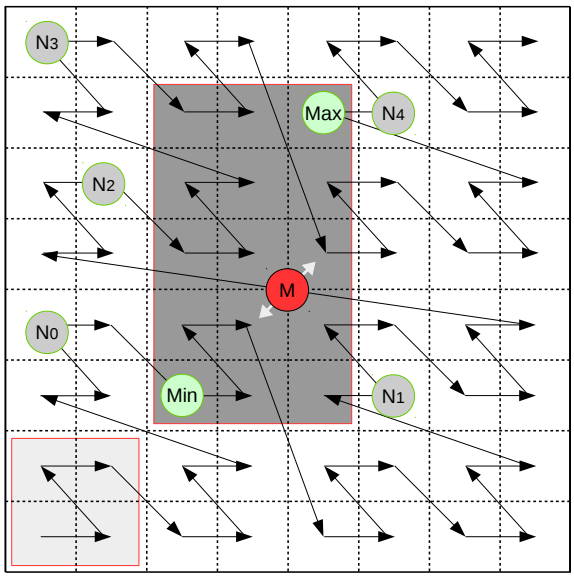


Fig. 2. In space filling curves, neighboring identifiers can be physically close (light-grey area). On the other hand, physically close peers can be far away on the space filling curve (nodes around red circle in middle).

C. Position Updates

Peers in LobSter periodically compute their *z-key* out of their positions and store a time-to-live (*ttl*), their overlay identifier (*o-id*) and IP-address into the DHT. This meta-data containing *ttl*, *o-id* and IP-address is stored on the DHT node which is responsible for the computed z-key. With this method, a peer is responsible for all meta-data which is associated with z-keys that fall into the peer's responsibility range in the DHT. To avoid outdated information to be spread, meta-data are only valid until the given time-to-live has elapsed. To reduce further costs, nodes could update their position adoptively e.g. feed the DHT only if they leave a specific geographical area and change their position. The update frequency could be set with respect to the motion of the peer: high update frequency, if peer moves fast, low frequency if movement is slow. In our simulations, peers move randomly without a pause, so that we do not consider adoptive position updates here. For better resilience in case of failures, the underlying DHT runs a simple replication service. Using Chord for example allows to pass copies of the meta-data to succeeding peers which take over the responsibility for the meta-data in case a peer leaves the DHT unexpectedly.

D. Range-queries in LobSter

Purpose of a *range-query* is to find as many peers as possible located in a geographical area. To search for nodes located in a specific area, we exploit the characteristics of a typical DHT in the way that initiators of a request retrieve meta-deta directly from the DHT. Using the z-order curve, a peer is able to compute a z-key for any geographical position. With this information at hand, a peer can check if any other node is currently located on this position by directly asking the DHT node which is responsible for the computed z-key of the position. Since meta-data is directly stored in the DHT, there is always one node responsible for the computed z-key which can answer the request.

The main idea is now to search successively for nodes in the DHT which are responsible for calculated z-keys. To explain it more in detail, the first step of a range-query in LobSter is to calculate the minimum and maximum z-keys which fall into the requested area on the space filling curve. An example can be seen in Figure 2: the dark-gray area contains z-keys which lie between, and include, the minimum and maximum keys (green circles).

The initiator peer starts the range query by starting a DHT lookup to find the node which is responsible for the maximum key inside the search area. The responsible node returns its range and a list of meta-data which is associated to the part of the area it is responsible for. Two possibilities exist now, either the node was responsible for all identifiers inside the queried area, or not. In the latter case, the information about its range tells the initiator which parts of the area are already found. The initiator calculates the next biggest z-key inside the area and starts a lookup to its responsible node. Executing a range-query, a peer consecutively searches for DHT nodes which are responsible for z-keys located inside the queried area and asks them for information about other peers they store.

The challenge is to calculate the next z-key located in the queried area. As can be seen in Figure 2, not all values between the minimum and maximum z-key lie in the queried area. Z-order curves, as well as other space filling curves, do not necessarily preserve locality of the keys. Physically close peers can be distant on the space filling curve and vice versa. In [27], Herzog and Tropf suggested an algorithm to search all elements inside a binary-tree structure which fall inside a given area. We utilize their algorithm for our purpose so that we are able to calculate the next biggest value on the z-order curve which falls into a given area.

IV. Evaluation of Location-based Overlays

To obtain realistic results and insights into the overlays, we use the event-based simulator PeerfactSim.KOM [5] for our simulations which contains implemented of Chord [24], Kademlia [14], Pastry [21] and further structured DHT overlays. Each simulation uses GNP coordinates [16] to estimate delays in the fundamental network and measurements from the PingEr project [13] for reasonable approximations of jitter and packet loss. Each of our simulation has been run with 20 different random seeds, therefore the values we obtained represent the average of 20 different outcomes.

In order to value the performance of the simulated overlays roughly, we focus on two metrics: *a)* - the ratio of missing nodes in a query-response in relation to the total number of nodes which are currently inside the queried area and *b)* - the amount of nodes which had to be visited during a range-query. The metrics are compared for different sizes of the queried area which will be given as ratio of range-query size to the size

of the whole terrain in which nodes can move ($400km^2$). Due to limited space, we do not further discuss the network traffic costs an overlay produces, for it can be roughly calculated by summing up the traffic costs for join messages, maintenance strategies and lookups (hops).

A. Simulation Setup

In our simulations, we consider a city with the size of 20 km x 20 km which is similar to a city at the size of Cologne (approximately $405km^2$). Except for the stationary scenario in which we compare Geodemlia to LobSter (Section IV-B), nodes follow a random movement model inside the urban area. As soon as peers reach the destination of their actual movement step, they choose a new point on the map and move towards this point. Assuming that nodes are able to walk by feet or drive by car, bicycle, etc. we limit random movements to a velocity between $3.6km$ per hour (pedestrian) and $54km$ per hour (car), since in most cities the speed limit is set to approximately $50km/h$. Anyway, in most cities it is not possible to drive faster than $50km/h$. With these settings, we consider an extreme case to stress our overlay and give a rough overview. More thorough simulation setups are planned.

During simulations, 10 percent of the simulated peers start range-queries which cover (rectangular) areas with a search-radius of 100 meters to 5 kilometers. This means that in extreme cases, all nodes in an area with the size of $100km^2$, (a quarter of the size of the simulated city), are searched. Peers follow an exponential churn model in which the frequency of leaving and joining peers is relatively high. Following the exponential churn model, only few nodes stay online for a relatively long time before they leave the network. The offline time of the peers on the other hand is much shorter so that peers join the network shortly after they have left it. In this evaluation, we run LobSter on top of Chord using a 160-bit identifier space. A Simple replication mechanism is used in which meta-data is replicated to successor nodes in Chord.

B. Comparison of Geodemlia and LobSter

With our first setup, we compare LobSter to Geodemlia [6] in a stationary scenario with 10000 peers without movement. Doing this, we try to find out in how far LobSter can compete with Geodemlia in terms of successful range-queries and lookup costs.

Geodemlia is initialized with $n = 4$ directions, $b = 32$ buckets per direction. In order to find out fair values for Geodemlia's bucket-size and α (number of parallel queries during range-queries), we conducted a parameter study with different values $\in [3, 6, 9, 12, 15]$ for the bucket-size and α. The result in Figure 3 is a bit surprising, since a smaller bucket-size leads to less nodes missing in the result of the range-query. Reason is that, smaller buckets lead to more diversity in the routing table. If buckets are too big, all contacted nodes return similar nodes, which forces Geodemlia's range-query to determine early.

Next, we compare Geodemlia with $n = 4$, $b = 32$, $\alpha = 15$ and buckets-size $= 3$ to LobSter. Figure 4 shows the ratio of missing nodes in a range-query to the total number of nodes inside the requested area for changing sizes of the queried area in percent. It can be seen that in LobSter no peers are missed in a range query, whereas range-queries in Geodemlia start to miss peers if the queried area exceeds approximately 0.02 percent. Reason is that, due the limited bucket-size, the number of visited nodes in Geodemlia does not increase with the size of the queried area, as can be seen in Figure 5.

As we see from this comparison, LobSter's performance and effectiveness are comparable to those of Geodemlia for relatively small range-queries.

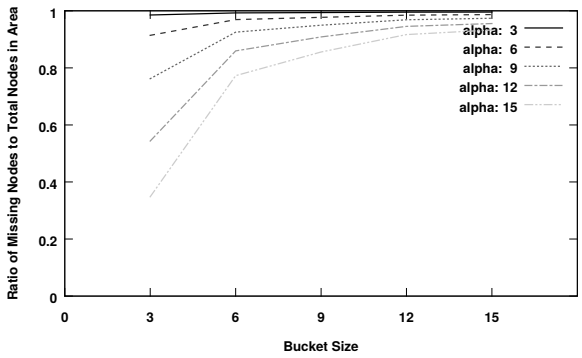


Fig. 3. Ratio of missing nodes to total nodes in area depends on the bucket-size and α.

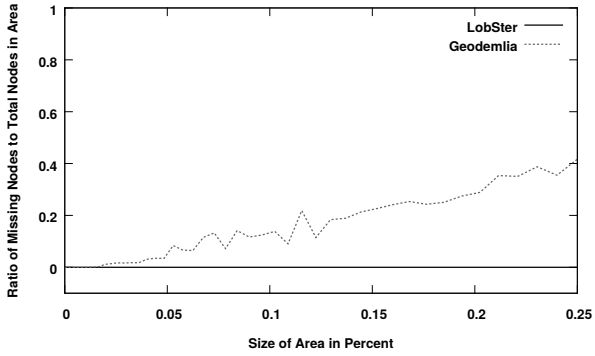


Fig. 4. Comparison of ratio of missing nodes to total nodes for LobSter and Geodemlia.

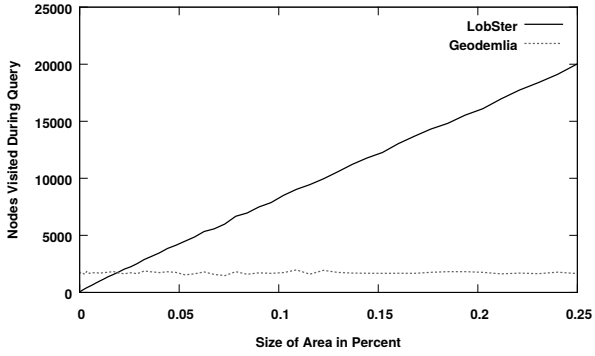


Fig. 5. Comparison of query costs in terms of hops for LobSter and Geodemlia.

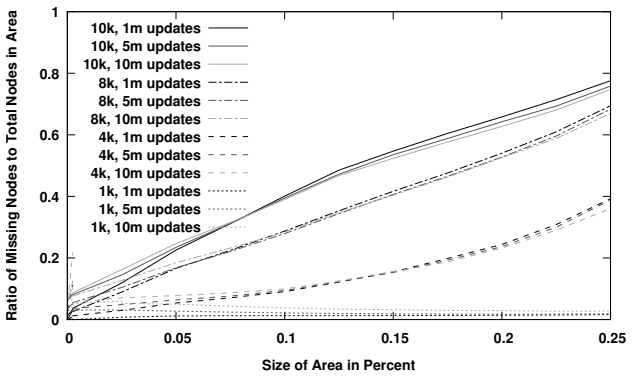


Fig. 6. The ratio of missing nodes to available online nodes in a query depends on the area size and the size of the network.

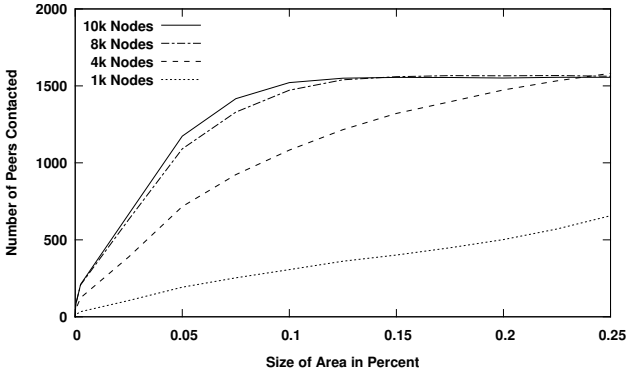


Fig. 7. The number of contacted peers during a range-query illustrate the costs of the query.

C. Evaluation of LobSter with Moving Peers

In this section, we investigate the performance of LobSter with different network sizes (1k, 4k, 8k, 10k) and movement of peers. We are not able to consider other solutions from related work here, as they are not designed to handle node movement naturally.

Figure 6 depicts for different area-sizes and different DHT update intervals (1,5,10 minutes) the ratio of missing nodes in a range-query to the total number of nodes which are currently in the searched area and present in the network. The result comprising 1000, 4000, 8000 and 10000 nodes shows that the number of missing nodes in a query grows with the area that is searched for nodes. The larger the area, the more nodes are missing in the response. The reason is obvious: the larger the area, the more nodes have to be visited in the DHT to find meta-data of nodes located in the area, the longer the request takes to be completed. Missing meta-data of nodes are a result of changes in the DHT during the lookup. From this point of view, small lookups are more reliable and provide better chance to find all nodes inside the requested area.

The number of nodes which are visited during a range-query is presented in Figure 7. Hops from lookups in Chord are not included in this figure, but lie around $O(\log(N))$ average per lookup for N nodes in the network. If small areas (up

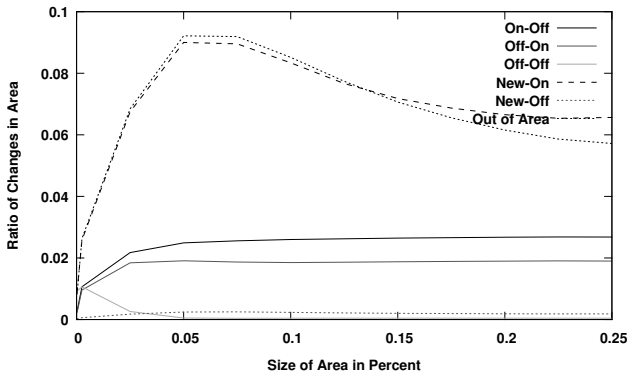


Fig. 8. A closer lookup to the changes in the system reveals that a fluctuation of peers exists during the query process. We have identified different types of changes during a query, such as peers that join or leave either the network or the area.

to 5 percent of the $400km^2$) are searched, the number of visited nodes increases linear with the size of the area that is requested. Then, above this threshold, the number of visited hops seems (for 8000, 10000 nodes) to reach a saturation point so that not more than 15 percent of all participants are visited during the range-query. For smaller networks, the point of saturation seems to be shifted to the right and only exist for big areas greater than 20 to 25 percent of the size of the simulated city ($400km^2$). The smaller the requested area is, the faster a query can be resolved and therefore less errors occur during the process of the range-query.

Figures 6 and 7 support our findings from Section IV-B and show that LobSter performs best for small areas (below 5 percent of the simulated city). Computing the corresponding radius of an area 4 percent the size of the city gives us $r = sqrt(0.04 * 400km^2)/2 = 2km$ which is quite fair.

Figure 8 reveals why LobSter's performance is limited (and probably why other solutions are limited, too). During simulation, we observe the positions of nodes in the area before and after a range-query and compare the result Doing so, we obtain a snapshot which shows changes in the constellations of the nodes. Those changes in the area effect the outcome of the range-query massively. We categorized our observations the following:

Nodes stay inside the area:

- Nodes might stay online during the query (We observed approx. 90 percent of all nodes to be in this category)
- Present nodes might leave the system (On-Off)
- Offline nodes could join the network again (Off-On)
- Nodes which do not participate do not join (Off-Off)

Nodes might enter the area during the query:

- Nodes might move into the area and participate in the system (New-On)
- Nodes might move into the area but be offline (New-Off)

Nodes (it does not matter whether online or offline) might leave the area during the query (Out of Area).

One can see that many active peers enter the area during the query (New-On), almost the same number of nodes leaves the

area during this time (Out of Area). It can be further observed that the amount of nodes which change from online to offline and remain inside the area is similar to those nodes switching from offline to online inside the area. The results conclude that, the smaller the queried area is, the higher is the fluctuation inside this area.

V. Conclusion

In this work, we studied the behavior of DHT-based, distributed location-based services, under nearly realistic conditions. We simulated LobSter, our solution of a DHT-based overlay, under the assumption that participants move permanently inside an urban area which has the size of $400km^2$. The speed with which nodes move is limited to $54km$ per hour which is realistic under the assumption that participants drive a car within the city. With our simulations, we roughly proof the performance of location-based service on top existing DHTs and show that linearization approaches such as space filling curves are suitable techniques to handle peer movement in overlays.

Comparing LobSter to Geodemlia, our simulation shows that obtaining good range-query results is costly in terms of nodes to be visited. We show further, that LobSter is able to solve small range-queries if peers are moving. In our evaluation, we only tested the performance of single range-queries with different sizes of the search area. With the knowledge that smaller range-queries are more successful than extensive searches, we can propose a mechanism that allows to separate one range-query into multiple parallel queries. Since LobSter processes nodes in the DHT successively and lookups to different parts of the DHT can be done individually, a range-query can be simply divided into multiple queries which are started simultaneously.

A deeper investigation shows that node fluctuation in the queried area limits the performance of range-queries in location-based overlays generally. Current solutions of location-based services still have to be extended with new techniques to support peer movement fully.

References

[1] F. Araújo and L. Rodrigues, "GeoPeer: A Location-Aware Peer-to-Peer System," in *Proc. of IEEE Int. Symposium on Network Computing and Applications (NCA)*, 2004.

[2] S. Asaduzzaman and G. von Bochmann, "GeoP2P: An Adaptive Peer-to-Peer Overlay for Efficient Search and Update of Spatial Information," *Computing Research Repository*, 2009.

[3] G. Brambilla, M. Picone, M. Amoretti, and F. Zanichelli, "An Adaptive Peer-to-Peer Overlay Scheme for Location-Based Services," in *Proc. of IEEE Int. Conf. on Network Computing and Applications (NCA)*, 2014.

[4] F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Proc. of Int. Workshop on Peer-to-Peer Systems (IPTPS)*, ser. LNCS, vol. 2735. Springer, 2003.

[5] K. Graffi, "PeerfactSim.KOM: A P2P System Simulator Experiences and Lessons Learned," in *Proc. of the Int. Conf. on Peer-to-Peer Computing (P2P)*. IEEE, 2011.

[6] C. Gross, D. Stingl, B. Richerzhagen, A. Hemel, R. Steinmetz, and D. Hausheer, "Geodemlia: A Robust Peer-to-Peer Overlay Supporting Location-based Search," in *Proc. of IEEE Int. Conf. on Peer-to-Peer Computing (P2P)*, 2012.

[7] D. Heutelbeck and M. Hemmje, "A Peer-to-Peer Data Structure for Dynamic Location Data," in *Proc. of IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, 2006.

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

[9] M. Knoll and T. Weis, "Optimizing Locality for Self-organizing Context-Based Systems," in *Proc. of Int. Workshop on Self-Organizing Systems (IWSOS)*, 2006.

[10] A. Kovacevic, N. Liebau, and R. Steinmetz, "Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search," in *Proc. of IEEE Int. Conf. on Peer-to-Peer Computing (P2P)*, 2007.

[11] J. Lee, H. Lee, S. Kang, S. M. Kim, and J. Song, "CISS: an efficient object clustering framework for dht-based peer-to-peer applications," *Computer Networks*, vol. 51, no. 4, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2006.07.005

[12] N. Lopes and C. Baquero, "Implementing range queries with a decentralized balanced tree over distributed hash tables," in *Proc. of Int. Conf. Network-Based Information Systems (NBiS)*, 2007.

[13] W. Matthews and L. Cottrell, "The PingER Project: Active Internet Performance Monitoring For the HENP Community," *Communications Magazine*, vol. 38, no. 5, 2000.

[14] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Proc. of the Int. Workshop on Peer-To-Peer Systems (IPTPS)*, ser. LNCS, vol. 2429. Springer, 2002.

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

[16] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-based Approaches," in *Proc. of the Twenty-First Annual Joint Conf. of the IEEE Computer and Communications Societies*, vol. 1, 2002.

[17] A. Pethalakshmi and C. Jeyabharathi, "Geo-Chord: Geographical Location-based Chord Protocol in Grid Computing," *International Journal of Computer Applications*, vol. 94, no. 3, 2014.

[18] M. Picone, M. Amoretti, and F. Zanichelli, "GeoKad: A P2P Distributed Localization Protocol," in *Proc. of IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, 2010.

[19] M. Picone, M. Amoretti, and F. Zanichelli, "Proactive Neighbor Localization Based on Distributed Geographic Table," *International Journal of Pervasive Computing and Communications*, vol. 7, no. 3, 2011.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," in *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, vol. 31. ACM, 2001.

[21] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. of the Int. Conf. on Distributed Systems Platforms (IFIP/ACM Middleware)*. Springer, 2001.

[22] M. T. Schlosser, M. Sintek, S. Decker, and W. Nejdl, "HyperCuP - Hypercubes, Ontologies, and Efficient Search on Peer-to-Peer Networks," in *Proc. of Int. Workshop on Agents and Peer-to-Peer Computing (AP2PC)*, 2002.

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

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proc. of the Int. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. ACM, 2001.

[25] Y. Tang, S. Zhou, and J. Xu, "LIGHT: A Query-Efficient Yet Low-Maintenance Indexing Scheme over DHTs," *IEEE Transactions on Knowledge Data Engineering*, vol. 22, no. 1, 2010.

[26] E. Tanin, A. Harwood, and H. Samet, "Using a Distributed Quadtree Index in Peer-to-Peer Networks," *VLDB Journal*, vol. 16, no. 2, 2007.

[27] H. Tropf and H. Herzog, "Multidimensional range search in dynamically balanced trees." *ANGEWANDTE INFO.*, no. 2, 1981.

[28] T. Zahn and J. H. Schiller, "DHT-based Unicast for Mobile Ad Hoc Networks," in *Proc. of IEEE Conf. on Pervasive Computing and Communications Workshops (PerCom)*, 2006.

[29] C. Zheng, G. Shen, S. Li, and S. Shenker, "Distributed Segment Tree: Support of Range Query and Cover Query over DHT," in *Proc. of Int. workshop on Peer-To-Peer Systems (IPTPS)*, 2006.