MobiHide: A Mobilea Peer-to-Peer System for Anonymous Location-Based Queries*

Gabriel Ghinita[1], Panos Kalnis[1], and Spiros Skiadopoulos[2]

[1] Dept. of Computer Science
National University of Singapore
{ghinitag,kalnis}@comp.nus.edu.sg
[2] Dept. of Comp. Science & Technology
University of Peloponnese, Greece
spiros@uop.gr

Abstract. Modern mobile phones and PDAs are equipped with positioning capabilities (e.g., GPS). Users can access public location-based services (e.g., Google Maps) and ask spatial queries. Although communication is encrypted, privacy and confidentiality remain major concerns, since the queries may disclose the location and identity of the user. Commonly, spatial $\mathcal{K}$-anonymity is employed to hide the query initiator among a group of $\mathcal{K}$ users. However, existing work either fails to guarantee privacy, or exhibits unacceptably long response time.
In this paper we propose MobiHide, a Peer-to-Peer system for anonymous location-based queries, which addresses these problems. MobiHide employs the Hilbert space-filling curve to map the 2-D locations of mobile users to 1-D space. The transformed locations are indexed by a Chord-based distributed hash table, which is formed by the mobile devices. The resulting Peer-to-Peer system is used to anonymize a query by mapping it to a random group of $\mathcal{K}$ users that are consecutive in the 1-D space. Compared to existing state-of-the-art, MobiHide does not provide theoretical anonymity guarantees for skewed query distributions. Nevertheless, it achieves strong anonymity in practice, and it eliminates system hotspots. Our experimental evaluation shows that MobiHide has good load balancing and fault tolerance properties, and is applicable to real-life scenarios with numerous mobile users.

1 Introduction

Consider the following scenario: Bob uses his GPS enabled mobile phone (e.g., iPAQ hw6515, Mio A701) to ask the query "Find the nearest AIDS clinic to my present location". This query can be answered by a *Location-Based Service* (LBS), e.g., Google Maps, which is *not* trusted. To preserve his privacy, Bob does not contact the LBS directly. Instead he submits his query via a trusted *pseudonym* service which hides his identity (services for anonymous web surfing are commonly available). Nevertheless, the query still contains the exact

* This work has been partially supported by project PENED 03 funded by the European Social Fund (75%) and the General Secretariat of Research and Technology (25%).

coordinates of Bob. One may reveal sensitive data by combining the location with other publicly available information. If, for instance, Bob uses his mobile phone within his residence, the untrustworthy owner of the LBS may infer Bob's identity (e.g., through a white-pages service) and speculate that he suffers from AIDS. Bob may even hesitate to ask innocuous queries such as "Find the nearest restaurant", in order to avoid unsolicited advertisement.

Recent research on LBS privacy focused on the $\mathcal{K}$-anonymity [17, 20] technique, which is used in relational databases for publishing census, medical and other sensitive data. A dataset is $\mathcal{K}$-anonymous, if each record is indistinguishable from at least $\mathcal{K}-1$ other records with respect to certain identifying attributes. In the LBS domain, a similar idea appears in Ref. [7, 9, 12, 15], which employ *spatial cloaking* to conceal the location of the querying user u: Instead of reporting the coordinates of u, they construct an *Anonymizing Spatial Region* (ASR or $\mathcal{K}$-ASR) which encloses u and $\mathcal{K}-1$ additional users. Typically, a central trusted server (called *location anonymizer*, or simply *anonymizer* in the sequel) exists between the users and the LBS. All users subscribe to the anonymizer and continuously update their position while they move. Each user sends his query to the anonymizer, which constructs the appropriate $\mathcal{K}$-ASR and contacts the LBS. The LBS computes the answer based on the $\mathcal{K}$-ASR, instead of the exact user location; thus, the response may contain false hits. The anonymizer filters the result and returns the exact answer to the user.

The centralized approach has several drawbacks; for example, the anonymizer may become bottleneck since it must handle frequent location updates as users move [8]. Most importantly, the centralized anonymizer poses a serious security threat. If it is compromised by an attacker, or forced to cooperate with a government agency, the history of all user movements and their queries may be revealed. For these reasons, two fully distributed systems emerged: *(i)* CLOAKP2P [5] is a Peer-to-Peer system which constructs $\mathcal{K}$-ASRs by considering users in the neighborhood of the querying user. *(ii)* PRIVÉ [8], on the other hand, clusters users in a hierarchical overlay network, resembling a distributed B^+-tree. Both systems minimize the security risk by distributing the sensitive information in numerous peers. However, we will show that CLOAKP2P fails to provide privacy for many user distributions, whereas PRIVÉ may suffer from slow response time, since root-level nodes constitute potential bottlenecks.

In this paper we propose MOBIHIDE, a Peer-to-Peer (P2P) system for anonymous location-based queries which addresses the problems of existing approaches. In MOBIHIDE the participating mobile devices form a hierarchical distributed hash table, based on the Chord P2P architecture [19], which indexes the locations of all users. In order to map the 2-D user locations to the 1-D Chord space, we employ the Hilbert space-filling curve [16]. $\mathcal{K}$-ASRs are collaboratively assembled by peers in a distributed fashion, by choosing random groups of $\mathcal{K}$ users (including the querying user) that are consecutive in the 1-D space. We prove that for uniform query distribution MOBIHIDE *guarantees* privacy, and we show experimentally that even for skewed distributions, the probability of identifying the querying user is very close to the theoretical bound. A clear trade-

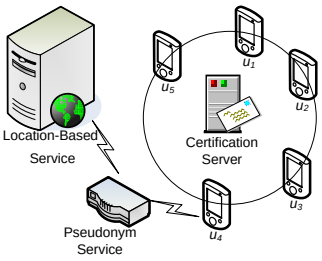


Fig. 1. System architecture

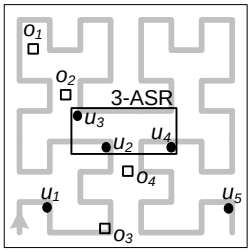


Fig. 2. Anonymized query, $\mathcal{K}$=3

off emerges between MobiHide and existing state-of-the-art Privé: the latter provides anonymity guarantees under any query distribution, but has an hierarchical architecture. On the other hand, MobiHide alleviates system hotspots, and still achieves strong anonymity in practice. Our experiments suggest that MobiHide is resilient to failures, achieves good load balancing and supports efficiently the relocation of users (as users move) and the construction of $\mathcal{K}$-ASRs (while querying); therefore, it is scalable to a large number of mobile users.

The rest of the paper is organized as follows: Section 2 presents an overview of MobiHide. Section 3 surveys the related work. Section 4 introduces our Hilbert-based randomized $\mathcal{K}$-ASR construction algorithm, whereas Section 5 describes the implementation of our system on top of Chord. Section 6 presents the experimental evaluation of MobiHide. Finally, Section 7 concludes the paper and discusses directions for future work.

2 Overview of MobiHide

We assume a large number of users who carry mobile devices (e.g., mobile phones, PDAs) with embedded positioning capabilities (e.g., GPS). The devices have processing power and access the network through a wireless protocol such as WiFi, GPRS or 3G. Moreover, each device has an IP address and can establish point-to-point communication with any other device in the system through a base station (i.e., the two devices do not need to be within the range of each other). For security reasons, all communication links are encrypted. In addition, we assume the existence of a trusted central *Certification Server* (CS), where users are registered. Prior to entering the system, a user u must authenticate against the CS and obtain a certificate. Users having a certificate are trusted by all other users. Typically, a certificate is valid for several hours; it can be renewed by recontacting the CS. Apart from the certificate, the CS returns to u a list of possible entry points to the P2P network (i.e., IP addresses of on-line users). Note that the CS does not know the locations of the users and does not participate in the anonymization process; therefore, it is not a security threat or a bottleneck.

The mobile users self-organize into a P2P system (see Figure 1) based on the Chord [19] distributed hash table architecture, well-known for its good scalability and fault-tolerance properties. The P2P system defines a 1-D space of index keys; MobiHide uses the Hilbert space-filling curve to map the 2-D user coordinates to 1-D space. The Hilbert curve is a continuous fractal (see Figure 2) which maps each region of the space to an integer. With high probability, if two points are close in the 2-D space, they will also be close in the Hilbert transformation [16].

Typically users ask Range or Nearest-Neighbor (NN) queries with respect to their location. In the example of Figure 2, user u_4 asks for the nearest object to his location (i.e., o_4). Assume that the required degree of anonymity is $\mathcal{K} = 3$ ($\mathcal{K}$ may vary among users). MobiHide identifies in a distributed manner a random set of 3 users (including u_4) that are consecutive in the 1-D space (i.e., u_2, u_3 and u_4 in the example), and constructs the corresponding 3-ASR (i.e., the rectangle which encloses the 3 users). Next, u_4 submits the 3-ASR NN query to the LBS through any existing pseudonym service [2]. Note that the pseudonym service hides the IP address of u_4 but is not aware of the users' locations. Furthermore, it does not become a bottleneck, since each user may choose his preferred pseudonym service.

The LBS returns to u_4 (through the pseudonym service) the NN of every point of the 3-ASR. Intuitively, the nearest neighbors of a region are all data objects inside the region plus the NNs of every point in the perimeter of the region [11]. In our example, these are objects o_2 and o_4. Finally, u_4 filters the false hits and determines his true NN (i.e., o_4). Note that the number of false hits depends on the $\mathcal{K}$-ASR; therefore we aim to minimize the size of the $\mathcal{K}$-ASR. Query processing at the LBS [11, 12, 15] is orthogonal to our work but outside the scope of this paper.

3 Background and Related Work

$\mathcal{K}$-anonymity was first discussed in relational databases, where sensitive published data (e.g., census, medical) should not be linked to specific persons. Samarati and Sweeney [17, 20] proposed the following definition: A relation satisfies $\mathcal{K}$-anonymity if every tuple is indistinguishable from at least $\mathcal{K}-1$ other tuples with respect to every set of quasi-identifier attributes. Quasi-identifiers are sets of attributes (e.g., date of birth, gender, zip code) which can be linked to publicly available data to identify individuals. Machanavajjhala et al. [14] proposed ℓ-diversity, an anonymization method that extends $\mathcal{K}$-anonymity by providing diversity among the sensitive attribute values of the anonymized set.

Privacy in location-based services has recently attracted a lot of attention. Spatial $\mathcal{K}$-anonymity is defined as [8]:

Definition 1 (Spatial $\mathcal{K}$-anonymity). *Let A be a set of $\mathcal{K}$ users with locations enclosed in an arbitrary spatial region $\mathcal{K}$-ASR. User $u \in A$ is said to possess $\mathcal{K}$-anonymity, if the probability of distinguishing u among the other users in A does not exceed $1/\mathcal{K}$, where $\mathcal{K}$ is the required degree of anonymity.*

Note that: *(i)* The definition assumes a *snapshot* of the users' locations. Although we support user mobility, $\mathcal{K}$-anonymity is undefined across multiple snapshots. *(ii)* Spatial $\mathcal{K}$-anonymity does *not* depend on the size of the $\mathcal{K}$-ASR. In the extreme case, the $\mathcal{K}$-ASR can degenerate to a point, if $\mathcal{K}$ users are at the same location. In general, we prefer small $\mathcal{K}$-ASRs, in order to minimize the processing cost at the LBS and the communication cost between the LBS and the mobile user. Nevertheless, some applications impose a lower bound on the size of the $\mathcal{K}$-ASR [15]. In such cases, the $\mathcal{K}$-ASR can be trivially scaled to satisfy the lower bound. The same procedure can also be used to avoid having users on the perimeter of the $\mathcal{K}$-ASR.

Also observe that the naïve solution of generating an arbitrary $\mathcal{K}$-ASR around the querying user, is not applicable. If, for instance, the user resides in a rural area, the $\mathcal{K}$-ASR may include only himself, whereas in a densely populated area, a too large $\mathcal{K}$-ASR will affect the query processing cost. Moreover, we cannot select $\mathcal{K}-1$ random users and send $\mathcal{K}$ distinct queries, because this would reveal the *exact* locations of $\mathcal{K}$ users; this is not desirable for any anonymization technique.

Ref. [7] considers mobile users who send queries to the anonymizer together with a spatial cloaking range δ_x, δ_y and a temporal cloaking interval δ_t. If $\mathcal{K}$-1 other users generate queries within this cloaking box, the query is issued, otherwise it is dropped. Ref. [9], on the other hand, assumes that users report periodically their location to the anonymizer, and focuses on concealing the exact location without considering query processing. The anonymizer indexes the locations of all users with a Quad-tree [18]. For a user u, it traverses the Quad-tree until it encounters a quadrant which includes u and less than $\mathcal{K}-1$ additional users. Then it selects the parent of that quadrant as $\mathcal{K}$-ASR. Casper [15] also employs a variation of the Quad-tree anonymization method. Casper attempts to build $\mathcal{K}$-ASRs by combining two neighboring quadrants, rather than going one level up in the tree every time more users are required. Finally, Ref. [12] introduces the HILBASR algorithm based on space-filling curves, and proposes an integrated framework for $\mathcal{K}$-ASR construction and query processing in LBS.

The previous approaches assume a centralized anonymizer. Recall from Section 1 that centralized approaches, among other drawbacks, are potential security threats. Closer to our approach is CLOAKP2P [5], which addresses the drawbacks of centralized anonymization by employing a fully distributed mobile Peer-to-Peer (P2P) system. In CLOAKP2P, the querying user u initiates $\mathcal{K}$-ASR construction by contacting all peers within a given physical radius r, which is a fixed system parameter. If the set of peers S_0 found in the initial iteration is larger than $\mathcal{K}$, the closest $\mathcal{K}$ of them are chosen to form the $\mathcal{K}$-ASR; otherwise, the process continues recursively, and all peers in S_0 issue a request to all peers within radius r. Intuitively, CLOAKP2P determines a $\mathcal{K}$-ASR by finding the $\mathcal{K}-1$ users closest to u. Unfortunately, this heuristic fails to achieve anonymity in many cases, since u tends to be closest to the center of the $\mathcal{K}$-ASR. We call this "*center-of-$\mathcal{K}$-ASR*" attack. In Section 6 we demonstrate that, in many cases, an attacker can identify u with probability much higher that $1/\mathcal{K}$. The experiments show that MOBIHIDE is considerably more secure compared to CLOAKP2P.

PRIVÉ [8] is another P2P system, which uses the Hilbert transformation to generate a sorted 1-D sequence of all users. PRIVÉ constructs *fixed* partitions of $\mathcal{K}$ users each (except the last one, which may have up to $2\mathcal{K}-1$ users). It is formally proved that this method *guarantees* anonymity (i.e., prevents identification of the query source) against any location-based attack, for any distribution of users and queries, even if an attacker knows the exact location of users. To generate fixed partitions, PRIVÉ must determine the absolute rank of each user in the sorted Hilbert sequence. To achieve this, it implements an overlay network which resembles a distributed B^+-tree. For each query the search must start at the root of the tree; this can overload the root peer. Although PRIVÉ has a load-balancing mechanism, its purpose is to equally share load among users during long periods of time, but it cannot avoid the root hotspot when the number of users or the query rate increases. In Section 6 we will show that even with 10,000 users and a moderate query rate, the response time is almost 10 minutes, while as many as 60% of the queries are rejected due to buffer overflows. In contrast, MOBIHIDE does not maintain fixed partitions, therefore it is much faster than PRIVÉ.

The privacy of user locations has also been studied in the context of related problems. Probabilistic Cloaking [4] does not apply spatial $\mathcal{K}$-anonymity. Instead, given an ASR, the LBS returns the probability that each candidate result satisfies the query, based on its location with respect to the ASR. Kamat et al. [13] focus on sensor networks and examine the privacy characteristics of different routing protocols. Hoh and Gruteser [10] describe techniques for hiding the trajectory of users in applications that continuously collect location samples.

MOBIHIDE is built on top of Chord [19], a Distributed Hash Table (DHT) protocol that supports scalable, fully decentralized key searching. Chord has a flat structure, where all peers have equal responsibilities and equally share the system load among themselves. Our work is also related to CANON [6], which is a framework for building hierarchical DHTs, while retaining the homogeneity of load and functionality offered by flat designs.

4 The MobiHide Spatial Anonymization Algorithm

We introduce MOBIHIDE, a P2P system which employs a randomized $\mathcal{K}$-ASR construction technique to offer query source anonymity, and is scalable to a large number of mobile users. Similar to PRIVÉ, MOBIHIDE is using the Hilbert ordering of the users' locations. However, instead of grouping users into fixed partitions, it forms a $\mathcal{K}$-ASR by randomly choosing $\mathcal{K}$ consecutive users, including the querying user.

Let $[u_1, \ldots, u_N]$ be the sequence of all users, ordered by their Hilbert value. To allow random $\mathcal{K}$-ASR selection for the users at the start and end of the sequence, the 1-D space becomes a ring (or torus), instead of an array. Therefore, u_1 is after u_N (and u_N is before u_1). Figure 3 presents an example, where u_q is the user who issues a query. There are $\mathcal{K}$ ways to choose a set of consecutive $\mathcal{K}$ users which includes u_q: $[u_{q-\mathcal{K}+1} : u_q], [u_{q-\mathcal{K}+2} : u_{q+1}], \ldots, [u_q : u_{q+\mathcal{K}-1}]$. This is equivalent to choosing a random offset $l \in [0, \mathcal{K}-1]$, representing the offset

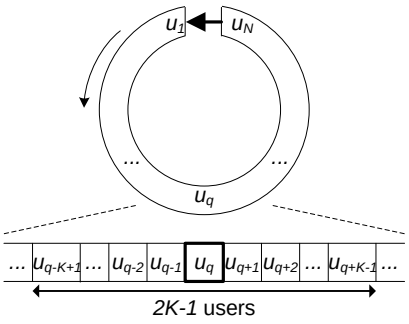


Fig. 3. Hilbert sequence ring

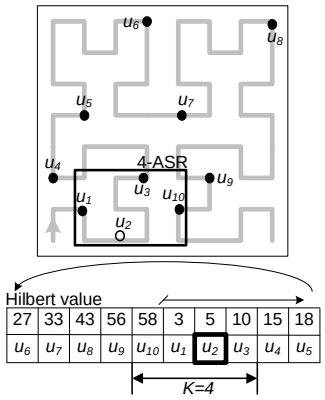


Fig. 4. $\mathcal{K}$-ASR construction in MobiHide

of u_q in the resulting sequence. For example, if $l = 0$, the resulting sequence is $[u_q : u_{q+\mathcal{K}-1}]$. Observe that we only need information in the neighborhood of u_q in order to select the sequence (as opposed to Privé, which needs the global ranking). Therefore, MobiHide works in a fully decentralized manner, and can be deployed on top of a scalable structure such as Chord.

Figure 4 shows an example of $\mathcal{K}$-ASR construction, where u_2 is the querying user. Let $\mathcal{K} = 4$ and assume that u_2 randomly selects offset $l = 2$. According to the Hilbert ordering, the resulting sequence of users is $[u_{10}, u_1, u_2, u_3]$. The corresponding $\mathcal{K}$-ASR is the *minimum bounding rectangle* (MBR) which encloses these four users. In this particular example it was necessary to wrap around the Hilbert sequence (from u_{10} to u_1). Observe that the "jump" in Euclidean distance due to wrapping, is not necessarily larger than other "jumps" that may occur within the sequence (e.g., from user u_8 to u_9). Therefore, the average size of the $\mathcal{K}$-ASRs (thus the query cost) is not affected significantly by wrapping. We investigate further this issue in Section 6.

Theorem 1. *If all users issue queries with the same probability (i.e., uniform distribution),* MobiHide `guarantees` *query anonymity.*

Proof. Denote by P_Q the probability of a user issuing a query (same for all users). The query source generates a random offset $l \in [0, \mathcal{K}-1]$; we denote by $\langle u, l\rangle$ the event of user u generating a set of users with offset l. The probability $P_{\langle u,l\rangle} = P_Q/K$. Refer to Figure 3, where u_q is issuing a query. Obviously, u_q must belong to the set associated to his query. To guarantee anonymity, the probability of identifying u_q as the query source must not exceed $1/\mathcal{K}$. We denote by A_q any set of users that includes u_q, and by P_{A_q} the probability of such a set being generated. We denote by P_{u_i} the probability of user u_i being the source of the query associated with A_q. Then, $P_{u_i} > 0$ only for users $[u_{q-\mathcal{K}+1} : u_{q+\mathcal{K}-1}]$, and by symmetry, $P_{u_{q-j}} = P_{u_{q+j}}$. We have:

$$P_{u_q} = \sum_{l=0}^{\mathcal{K}-1} P_{\langle u_q, l\rangle} = P_Q, \qquad P_{u_i} = \sum_{l=i-q}^{\mathcal{K}-1} P_{\langle u_i, l\rangle} = \frac{\mathcal{K}-i+q}{\mathcal{K}} P_Q,\ i > q$$

$$P_{u_q} + 2 \sum_{i=q+1}^{q+\mathcal{K}-1} P_{u_i} = P_{A_q}$$

The probability of pinpointing u_q as the query source is

$$\frac{P_{u_q}}{P_{A_q}} = \frac{P_Q}{\left(1 + 2\sum_{i=1}^{\mathcal{K}-1} \frac{\mathcal{K}-i}{\mathcal{K}}\right) P_Q} = \frac{1}{\mathcal{K}}, \tag{1}$$

hence user u_q is $\mathcal{K}$-anonymous.

4.1 The Correlation Attack

In practice, the query distribution is not always uniform, hence Theorem 1 may not hold. In the extreme case, the same user (e.g., u_q) would send all queries and he would be included in all $\mathcal{K}$-ASRs. An attacker can intersect the $\mathcal{K}$-ASRs and pinpoint u_q as the querying user with high probability. It is more realistic, however, that many users ask queries, even if the query distribution is skewed. In this case, intersecting the $\mathcal{K}$-ASRs is unlikely to compromise the system, since the random sequence selection in MobiHide distributes the anonymized regions in the entire space. In order to succeed, the attacker should know the *exact locations of all users*, to be able to reconstruct the Hilbert sequence. Then, he could find the users included in each $\mathcal{K}$-ASR by reverse-engineering the $\mathcal{K}$-ASR construction mechanism, and speculate that the users who appear more frequently are the ones who issued the queries.

Consider the extreme case where the attacker knows the exact location of all users and intercepts the set $\mathcal{R}$ of $\mathcal{K}$-ASRs. We formalize the correlation attack as follows: *(i)* Construct a histogram F with the number of occurrences of every user in any of the queries. *(ii)* For each $R \in \mathcal{R}$: infer the query source as the user in R with the highest number of occurrences in F.

The correlation attack gives an attacker powerful means to infer the query source. Privé guarantees anonymization against this type of attack, but as discussed in Section 3, scales poorly as the number of users increases. MobiHide cannot offer theoretical guarantees when the query distribution is extremely skewed. However, we believe that in practice this attack is hard to stage, since it is difficult for an attacker to know the exact locations of all users at each snapshot. Furthermore, we show experimentally (Section 6) that the probability of identifying the querying user in MobiHide is very close to the theoretical bound $1/\mathcal{K}$, even if the attacker knows all users' locations and the query distribution is skewed. Finally, observe that MobiHide does not suffer from the "*center-of-$\mathcal{K}$-ASR*" attack (see Section 3) because, by construction, the probability of u_q to be closest to the center of the $\mathcal{K}$-ASR is $1/\mathcal{K}$.

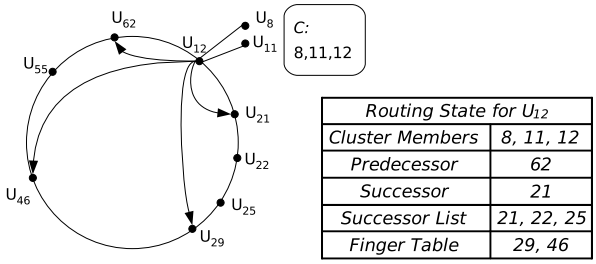


Fig. 5. MobiHide implementation over Chord

5 Implementation of MobiHide

MobiHide users organize themselves into a Chord [19] P2P system. Chord is a Distributed Hash Table (DHT), where each peer (or node) has an m-bit key (the Hilbert value in our case), and it stores a routing table with pointers to other nodes (see Figure 5). The routing table at peer n with key key_n consists of:

- a *successor* and *predecessor* pointer to the node with the key that immediately follows (respectively, precedes) key_n on the ring
- a *successor list*, used mainly for fault tolerance, with a list of consecutive peers that follow n on the ring
- a *finger table*, with m pointers to nodes that are situated at 2^i distances from n ($i = 0, 1, .., m-1$}.

We denote by $\mathcal{H}(u)$ the Chord key of user u. Assume that each user is mapped to a distinct Chord node. When user u wants to ask a query, he initiates the $\mathcal{K}$-ASR construction procedure, denoted by $\mathcal{K}$*-request*. u generates a random offset $l \in [0, \mathcal{K}-1]$, and contacts the set P of l predecessors and the set S of $\mathcal{K}-1-l$ successors on the Chord ring. The resulting $\mathcal{K}$-ASR is the MBR that encloses users in $P \cup S \cup \{u\}$. The complexity of a $\mathcal{K}$-request is $O(\mathcal{K})$ overlay hops.

Since $\mathcal{K}$ can be large (e.g., 50-100) in practice, we wish to reduce the number of hops, and hence the latency of $\mathcal{K}$-request. We introduce an additional level of hierarchy, such that each overlay node represents a *cluster of users*, rather than a single user. Each cluster has between α and 3α-1 users, where α is a system parameter. If the cluster reaches 3α, a split is performed and an additional ring node is created. If the size falls below α, a merge operation with another overlay node is performed[3]. We chose 3α, instead of 2α, as the upper bound on size, to minimize frequent merge and split operations. Each cluster has a representative, or cluster *head*, which is part of the Chord ring. In the example of Figure 5, u_{12} is the head of cluster $\{u_8, u_{11}, u_{12}\}$. The head's key on the ring is the maximum of all keys inside its cluster, in order to preserve the key ordering on the ring. The cluster membership is maintained by the head, and is replicated to all cluster members, to enhance fault-tolerance. Heads are rotated periodically to achieve

[3] Obviously, if more keys fall within a Chord segment, there will also be proportionally more nodes in that segment; therefore, hot-spots are avoided.

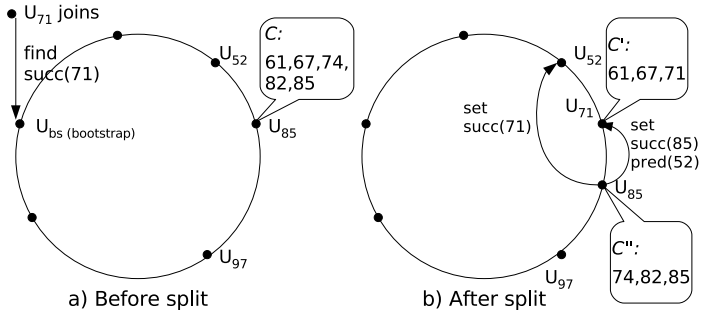


Fig. 6. Join and Split, α=2

load-balancing. We denote by C_u the cluster that contains user u, and by CH_u the head of C_u.

We further describe how various operations are performed in MobiHide. For each operation, we consider two performance metrics:

- *latency*: the time to completion, measured as the number of overlay hops on the longest path followed. Multiple paths may be followed in parallel.
- *cost*: the communication cost of an operation, measured as the number of transmitted messages (communication cost typically prevails over CPU cost).

Join and Departure. User join is illustrated in Figure 6a. User u with key $\mathcal{H}(u) = 71$ authenticates at the certification server and receives the address of some user u_{bs} inside the system. u_{bs} issues a search for key $\mathcal{H}(u)$, which returns the address of u_{85}, the successor of 71 on the ring. u contacts u_{85} and joins cluster C. Hence, $C_u \equiv C$ and $CH_u \equiv u_{85}$. Upon u's join, CH_u checks the new size of cluster C_u, and if $size(C_u) = 3\alpha$, CH_u splits his cluster into two halves, in increasing order of key values. He appoints one of his cluster members, CH'_u, as head of the newly formed cluster. All nodes in the initial cluster are notified. CH_u and CH'_u also notify their predecessor and successor on the ring. CH'_u inherits a large part (if not all) of the finger table of CH_u; the rest of the table is determined through the Chord stabilization process [19].

In our example, the new size of C is 6 and $\alpha = 2$, so u_{85} triggers a split operation (Figure 6b). u_{85} divides his cluster C into two halves, C' with members 61, 67 and 71, and C'' with members 74, 82 and 85. u_{71} is appointed as head of C', while u_{85} remains head for C''. u_{85} sets his predecessor pointer to u_{71}, and notifies the former predecessor u_{52} to change its successor from u_{85} to u_{71}. The complexity of join is $O(\log N - \log \alpha)$ latency and $O(\log N - \log \alpha + \alpha)$ communication cost (the last term stands for notifying all cluster members).

User u can depart gracefully, or fail; failure is addressed in Section 5.1. When u departs gracefully, he notifies his cluster head CH_u, who updates the cluster membership. If the departing node is cluster head, he appoints one of his members as new head. A merge can be triggered by departure. In this case, user CH_u triggering the merge contacts randomly either his successor s or predecessor p

```
u.findASR(H,K)
  compute rank_H in sorted order of C_u
  generate random offset l
  before = max(0,l - rank_H)
  after = max(0,K-l + rank_H − size(C_u))
  if (after > 0)
    succ.FwdReq(after,1)
  if (before > 0)
    pred.FwdReq(before,−1)
  wait for partial MBRs
  K-ASR = union of all received MBR
```

```
u.K-request(K)
  call CH_u.findASR(H(u),K)
u.FwdReq(count,direction)
  if (direction == 1) /*Look Forward*/
    return MBR of first count keys
    if (count > size(C_u))
      succ.FwdReq(count − size(C_u),1)
  else                /*Look Backward*/
    return MBR of last count keys
    if (count > size(C_u))
      pred.FwdReq(count − size(C_u),−1)
```

Fig. 7. Pseudocode for $\mathcal{K}$-Request

on the Chord ring to merge[4]. CH_u transfers his members (including himself) to the merging peer and ceases to be cluster head. All members are notified and the successor and predecessor pointers are updated.

Relocation. When user u moves to a new location, his Hilbert value $\mathcal{H}(u)$ changes. If the new $\mathcal{H}'(u)$ falls within the key range of other users in cluster C_u, u only needs to inform his cluster head of the key change. Otherwise, u performs a graceful departure followed by a join. Since Hilbert ordering preserves locality, it is likely that the relocation will be within a small distance from the initial ring position. The worst case complexity of relocation is $O(\log N - \log \alpha)$ latency and $O(\log N - \log \alpha + \alpha)$ communication cost.

$\mathcal{K}$-request. To generate a $\mathcal{K}$-ASR, u forwards a $\mathcal{K}$-request to his cluster head CH_u (unless u himself is the cluster head). CH_u generates a random offset $l \in [0, \mathcal{K}-1]$. Then, CH_u examines the membership list of his cluster C_u and determines how many users in C_u will belong to the $\mathcal{K}$-ASR. CH_u computes the values $before$ and $after$ corresponding to the number of users in $\mathcal{K}$-ASR that are *outside* C_u and precede (respectively, follow) the set of keys in C_u. CH_u issues a request for the MBR[5] of these members to his predecessor p and successor s. In p and s the same procedure is followed recursively, until $\mathcal{K}$ users are found. CH_u waits for all answers, and assembles the $\mathcal{K}$-ASR as the union of the received MBRs. The pseudocode[6] for $\mathcal{K}$-request is given in Figure 7. The complexity is $O(\mathcal{K}/\alpha)$ in terms of both latency and communication cost. Once the $\mathcal{K}$-ASR is assembled, u can submit it to the LBS using his preferred pseudonym service.

5.1 Fault-tolerance and Load Balancing

MobiHide inherits the good fault-tolerance properties of Chord [19]. Similar to Chord, some of the pointers to other peers (i.e., successor and predecessor pointers, the successor list and the finger table) may be temporarily corrupted (e.g.,

[4] Alternatively, an interrogation phase can find which of s or p has fewer members, and merge with that one (to avoid cascaded splits and to equalize cluster sizes).

[5] CH_u only acquires the MBR, not the exact location of users in other clusters.

[6] We use the Remote Procedure Call convention $u.routine()$; i.e., u is the node where *routine* is executed

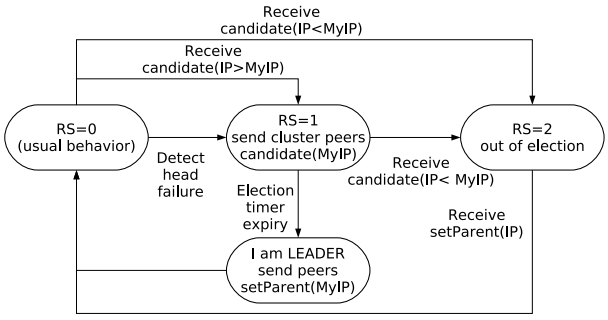


Fig. 8. Leader Election Protocol

when a user fails). Such pointers are corrected periodically through a stabilization process. In addition to stabilization, MobiHide implements an intra-cluster maintenance mechanism. Each cluster head periodically (i.e., every δt seconds) checks if all cluster members are alive, by sending *beacon* messages; beacons contain the current cluster membership in addition to the successor and predecessor nodes of the head on the Chord ring. If a user fails to respond for $2\delta t$ seconds, he is considered failed and is removed from the cluster. Similarly, a non-head node that does not receive a beacon from his head for $2\delta t$ seconds, concludes that the head has failed and initiates a leader election protocol (see Figure 8). The RecoveryState (RS) variable of each node indicates whether the node is in normal operation ($RS = 0$) or participates in the election protocol. Since the cluster membership is replicated at all cluster nodes, recovery is facilitated. Upon detecting leader failure, node n enters the $RS = 1$ state, sends a $candidate(n.IP)$ message to all peers in the cluster and sets an election timer large enough to allow other peers to respond to the candidature proposal. When a node receives the $candidate(IP)$ message, it initiates its own candidature only if its address is smaller than IP; otherwise, it enters the $RS = 2$ state and waits for a $setParent$ message. The user with the smallest address declares himself leader and notifies all other cluster members, as well as the predecessor and successor on the ring.

To prevent unequal load sharing, a simple rotation mechanism is enforced among cluster members. The rotation is triggered when a certain load threshold is reached. This threshold is measured in terms of number of messages sent/received, since the communication cost is predominant in terms of both energy consumption and fees payed to the service provider. When the cluster head CH transfers leadership to another cluster member CH', he transfers his routing state on the Chord ring and the cluster membership to CH'. Observe that the Chord key does not change, since it is the maximum key among all cluster members. Therefore, the overhead for the P2P network is minimal.

6 Experimental Evaluation

We implemented MobiHide on top of Chord in the p2psim [1] suite, a packet-level simulator for P2P systems. We consider topologies with $1sec$ average round-

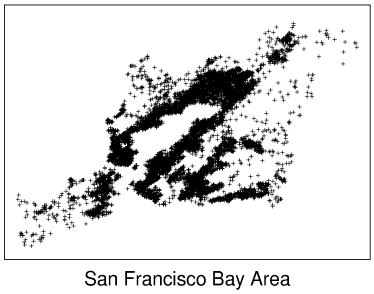


Fig. 9. Dataset

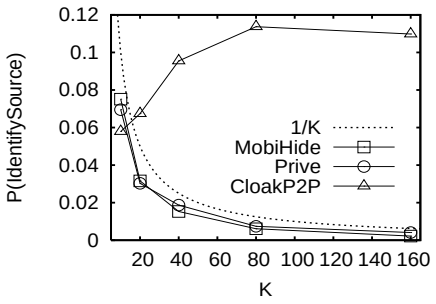


Fig. 10. "*center-of-$\mathcal{K}$-ASR*" attack

trip delay, a typical value for wireless devices. To highlight the behavior of our system, we only consider packet loss as an effect of queueing at the processing nodes, and not as a result of link faults. Our dataset corresponds to the San Francisco Bay Area (Figure 9) and is constructed with the *Network-based Generator of Moving Objects* [3], which models the movement of mobile users on public road infrastructures. We consider scenarios with 1,000 to 10,000 users and anonymization degree $\mathcal{K}$ between 10 and 160. If not stated differently, we set $\alpha = 5$ (see Section 5). We compare MobiHide against the two existing distributed spatial anonymization systems (i.e., cloakP2P and Privé).

Anonymization Strength. Theorem 1 theoretical demonstrates that MobiHide guarantees $\mathcal{K}$-anonymity for a uniform query distribution. To complete our study, we also evaluate the anonymity strength of MobiHide for skewed query distributions. To this end, we assume 10K users and 10K queries and consider a Zipfian query distributions with $\vartheta = 0.8$. We start by focusing on the "*center-of-$\mathcal{K}$-ASR*" attack. We revisit the query scenario from Ref [8], and present a comparison of MobiHide against Privé and cloakP2P. Let us denote by u_c the closest user to the center of the $\mathcal{K}$-ASR. Figure 10 illustrates the probability of u_c being the query source (i.e., the user initiating the query). Theoretically, to achieve anonymity, the above probability should be bounded by $1/\mathcal{K}$. In other words, the performance of the evaluated algorithms should be *under* line $1/\mathcal{K}$ (the dotted line of Figure 10). cloakP2P, for $\mathcal{K} \geq 20$, does not satisfy the theoretical bound. For instance, for $\mathcal{K}$=40, the probability of u_c being the query source is 10%, i.e., four times the maximum allowed bound ($1/\mathcal{K}$=2.5%). The users are likely to come uniformly from all directions; hence, u_c is disclosed as the query source. Contrary, Privé and MobiHide always satisfy the theoretical bound. Notice that in some cases, the MBR of the $\mathcal{K}$-ASR may contain a few more than $\mathcal{K}$ users. This is why the results for Privé and MobiHide are not identical to the $1/\mathcal{K}$ line.

In Figure 11 we consider the correlation attack (see Section 4.1). We assume the extreme case, where the attacker knows the exact locations of all users (recall that this attack is unlikely to occur in practice). We show the results for

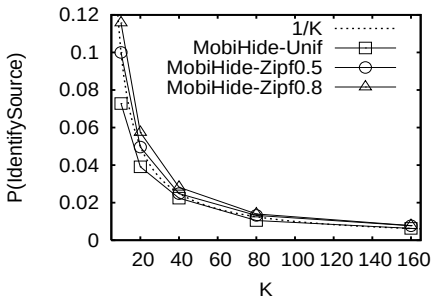


Fig. 11. Correlation attack (MobiHide)

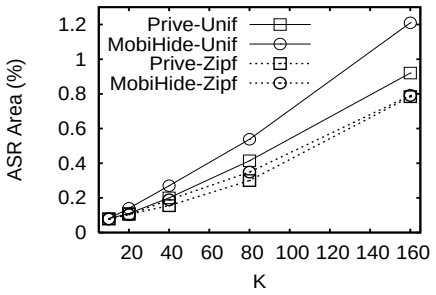


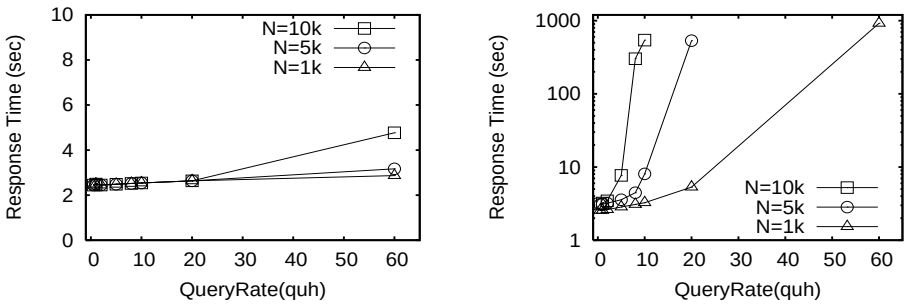
Fig. 12. $\mathcal{K}$-ASR Area

uniform and Zipf query distribution, with $\vartheta = 0.5$ and $\vartheta = 0.8$. As expected, for uniform distribution anonymity is always preserved. Actually, in this case MobiHide behaves almost identical to Privé (not shown in the graph). Anonymity is also entirely preserved for $\vartheta = 0.5$. As the distribution becomes more skewed, MobiHide may fail to preserve anonymity by a small margin. In most cases, however, the probability of identifying the query source is very close to the theoretical bound $1/\mathcal{K}$. In the worst case, for $\mathcal{K}$= 160, ϑ = 0.8, the probability of identifying the query source was $1.2/\mathcal{K}$. Observe that in Figure 11 we did not consider cloakP2P, as it can be easily compromised by the much simpler "*center-of-$\mathcal{K}$-ASR*" attack. Since it fails to provide anonymity in many cases, we will not consider cloakP2P any further.

$\mathcal{K}$-ASR Size. MobiHide wraps around the Hilbert sequence in order to handle users near the start/end of the sequence. In some cases, this may yield $\mathcal{K}$-ASRs with larger area, compared to Privé; consequently, the query processing cost will increase. To investigate this issue, we considered uniform and Zipf ($\vartheta = 0.8$) query distributions over a set of 10K users and varying $\mathcal{K}$. In Figure 12 we plot the average area of the $\mathcal{K}$-ASRs as a percentage of the entire dataspace. Observe that for the Zipf distribution the two systems behave almost identical, while for uniform distribution MobiHide generates 25% larger $\mathcal{K}$-ASRs in the worst case. Therefore, we tradeoff at most 25% in additional query processing cost, but we obtain far superior system scalability as we will show next.

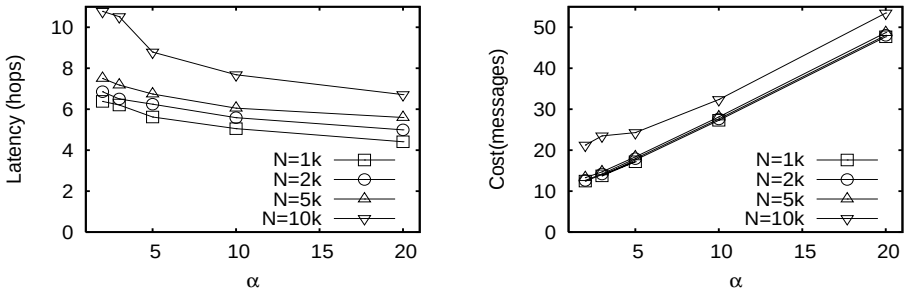
Scalability (response time). The most important advantage of MobiHide is its increased scalability due to the highly decentralized structure. Here, we evaluate the response time of the system for 1K, 5K and 10K users. The querying users are selected with a Zipf ($\vartheta = 0.8$) distribution[7]. We use exponential distribution to model the query rate, and the mean is varied between 0.5 and 60*quh* (*Queries per User per Hour*). Processing time at each node is exponentially distributed with mean 50*ms*. This is a realistic processing time that includes CPU processing and network buffer access. We set $\mathcal{K}$=40 and inject queries for a period of 600*sec*. From Figure 13(a), we can see that the response time is short (i.e,

[7] MobiHide behaves even better for uniform distribution of the querying user.



(a) MobiHide (b) Privé

Fig. 13. Scalability, $\mathcal{K} = 40$



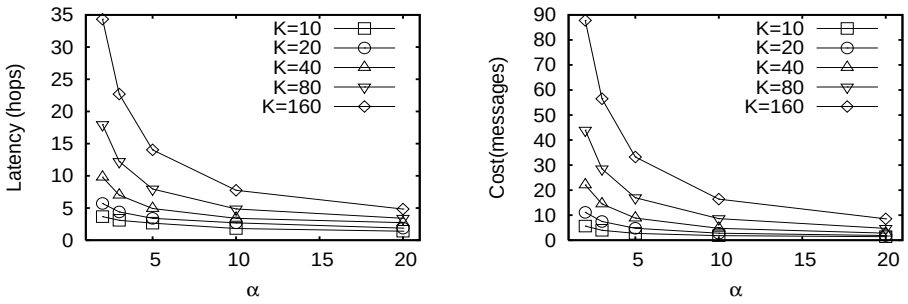
(a) Latency (b) Communication cost

Fig. 14. Join

does not exceed $5sec$) even for large user populations and high query rates. Note that the experiment assumes unbounded message queues at the nodes; therefore the drop rate of requests is 0. We also considered bounded queues (size = 100); in the worst case, the drop rate was 3.4%.

In Figure 13(b) we repeated the same experiment for Privé. Observe that the response time grows sharply with the query rate, due to delays at the root node. For 10K users and $10quh$ the response time is almost $600sec$ (whereas, MobiHide needs only $2.5sec$). Again, these results are for unbounded queues. For the bounded case (queue size = 100), the drop rate was 26% for $8quh$; for $10quh$ the drop rate surges as high as 60%. From the previous experiments it is obvious that MobiHide outperforms Privé.

Join. In this experiment, we measure the latency (i.e., number of hops) and communication cost (i.e., total number of messages) for the user join operation. Starting from a stable system, an additional 10% of the initial user population joins randomly the system. Figure 14(a) shows the latency for N = 1K, 2K, 5K and 10K users, for varying α (recall that the cluster size is between α and 3α). The plot confirms the theoretical expected complexity $O(\log N - log\alpha)$. For low α values, we observe a slight increase, due to the increasing proportion of split



(a) Latency (b) Communication cost

Fig. 15. $\mathcal{K}$-Request Operation

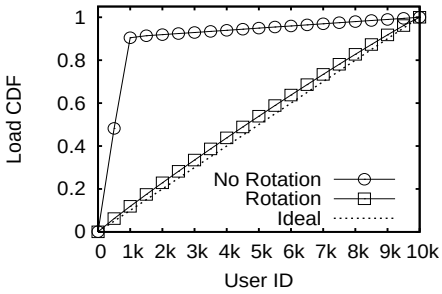


Fig. 16. Load Balancing

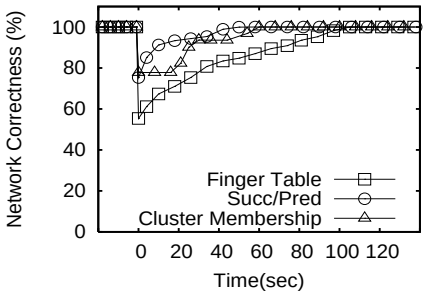


Fig. 17. Fault Tolerance

operations. In terms of communication cost (see Figure 14(b)), the dominant factor is $O(\alpha)$ due to the intra-cluster notification. There is a tradeoff between join latency and communication cost in terms of α. For low α values, the cluster maintenance cost is lower, but the latency increases. Furthermore, a low α also causes increased latency and communication cost during $\mathcal{K}$-requests, as we will show shortly. Our experiments suggest that a value $5 < \alpha < 10$ is likely to yield good results in practice.

$\mathcal{K}$-Request. We consider a 10K user population with 10K uniformly distributed queries (note that the distribution does not influence the latency or communication cost in the absence of queuing). Figure 15(a) and 15(b) show the average latency and communication for constructing the $\mathcal{K}$-ASRs (α is varied). Both the latency and communication cost are favored by larger α values. However, a compromise must be reached among the $\mathcal{K}$-Request performance, maintenance cost and system scalability. Larger α determines higher maintenance cost and also yields a more centralized system, with inferior peak-load performance.

Load Balancing. Due to the hierarchical nature of MobiHide, the cluster heads that participate on the Chord ring bear more load than other cluster members. Here, we evaluate the rotation mechanism of MobiHide which aims at distributing the load evenly. We set α=5, $\mathcal{K}$=20 and simulated a 10K user network, where an average of $3.6quh$ are generated. The total simulated time is

3 hours, and a rotation is triggered at every 300 messages received by a node. Figure 16 shows the cumulative distribution function (CDF) of the *sorted* node loads. Without rotation, the roughly 1,000 cluster heads (i.e., $10000/2\alpha$ as 2α is the average cluster size) bear 90% of the system load. With rotation, the load balancing is very close to the ideal (i.e., linear CDF, plotted as dotted line). Note that, for a load unit setting of 300 and a rotation cost of 2α messages, the rotation overhead is only $2\alpha/300 = 3\%$. This overhead can be decreased further by increasing the load unit.

Fault Tolerance. In this experiment we evaluate the fault-tolerance features of MobiHide. We consider 10K users and α=5. Chord performs periodical maintenance for its pointers. The respective timers are set at $3sec$ for the successor/predecessor, $10sec$ for the successor list and $30sec$ for the finger table pointers. The intra-cluster beacon timer $\delta t = 10sec$. We consider three network correctness metrics: *(i)* the intra-cluster correctness, measured as the ratio of correct cluster membership entries out of the total entries, *(ii)* the succ/pred correctness, measured as the ratio of correct successors/predecessors over the total number of successor/predecessor pointers, and *(iii)* we define similarly the correctness of finger tables. Note that, for correct execution of $\mathcal{K}$-request operations, only the successor/predecessor and intra-cluster membership need to be 100% accurate; the finger table pointers are only used for join and relocation operations, and their inaccuracy can only cause a slight increase in latency. Figure 17 shows the evolution in time of the three metrics, starting with a correct network, when 25% of the users fail simultaneously; $t = 0$ is the time of failure. We observe that the succ/pred and intra-cluster correctness are established after $60sec$. For the intra-cluster correctness, it takes the system roughly three purge intervals ($6\delta t$) to detect head failure, elect new leaders and establish correct cluster membership. The finger table is restored after $120sec$.

7 Conclusions

While location-based services become essential in supporting a broad area of applications (navigation systems, emergency services, etc), new privacy concerns arise for LBS users (e.g. in the near future GSM phones will be equipped with a "clipper" chip that accurately tracks users). In this paper, we propose MobiHide, a scalable P2P system for anonymous LBS queries. MobiHide indexes users into a hierarchical Chord network, according to the 1-D Hilbert ordering of their coordinates, and builds $\mathcal{K}$-ASRs by randomly choosing Hilbert sequences of $\mathcal{K}$ users. Our results confirm that in practice, MobiHide outperforms existing solutions: our system provides strong anonymity, it is fault-tolerant, and scales to large numbers of mobile users.

In future work, we plan to address the issue of anonymizing user trajectories, as opposed to user locations. Furthermore, we plan to investigate efficient methods to anonymize queries for infrastructure-less environments, such as ad-hoc wireless networks (Wi-Fi, Bluetooth), where point-to-point communication

channels do not exist between any pair of users, and only users within a limited physical range can be contacted.

References

1. *p2psim:* The Peer-to-Peer Network Simulator. *http://pdos.csail.mit.edu/p2psim.*
2. Tor: Anonymity Online. http://tor.eff.org/.
3. T. Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, 2002.
4. R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *Proc. of Privacy Enhancing Technology Workshop*, 2006.
5. C.-Y. Chow, M. F. Mokbel, and X. Liu. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services. In *ACM International Symposium on Advances in Geographic Information Systems*, 2006.
6. P. Ganesan, K. Gummadi, and H. Garcia-Molina. Canon in G Major: Designing DHTs with Hierarchical Structure. In *Proc. of ICDCS*, pages 263–272, 2004.
7. B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proc. of ICDCS*, pages 620–629, 2005.
8. G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems. In *Proc of WWW*, 2007.
9. M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of USENIX MobiSys*, 2003.
10. B. Hoh and M. Gruteser. Protecting Location Privacy through Path Confusion. In *Proc. of SecureComm*, 2005.
11. H. Hu and D. L. Lee. Range Nearest-Neighbor Query. *IEEE TKDE*, 18(1):78–91, 2006.
12. P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving Anonymity in Location Based Services. Technical Report TRB6/06, National University of Singapore, 2006.
13. P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing Source-Location Privacy in Sensor Network Routing. In *Proc. of ICDCS*, 2005.
14. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *Proc. of ICDE*, 2006.
15. M. F. Mokbel, C. Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proc. of VLDB*, 2006.
16. B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. *IEEE TKDE*, 13(1):124–141, 2001.
17. P. Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE TKDE*, 13(6):1010–1027, 2001.
18. H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.
19. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.
20. L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.