Nearby Friend Alert:
Location Anonymity in Mobile Geosocial Networks

Realizing the potential of "nearby friend alerts" in mobile geosocial networks requires addressing privacy issues. The authors enhance the grid-and-hashing paradigm by increasing detection accuracy while preserving wireless bandwidth. Furthermore, their multilevel grid scheme offers continuous proximity detection.

Hong Ping Li, Haibo Hu, and Jianliang Xu
Hong Kong Baptist University

Mobile geosocial networking lets mobile users interact with each other based on their locations. For example, the Google Latitude service lets you visualize the locations of nearby friends on a map and launch social activities, such as chatting or collaborative filtering. So, you might use the service to ask nearby friends, "Where can I find good pizza around here?" Facebook, Foursquare, Gowalla, Loopt, and others offer similar services.

Although mobile geosocial networking presents various business and marketing opportunities, privacy concerns have hindered its wider acceptance. In particular, such services currently collect location information from mobile users (although users can opt out of disclosing their locations to untrusted users or to a service provider when the service isn't needed). Recent research into preserving location anonymity in accessing location-based services (LBS) has mainly focused on queries over public points of interests (such as gas stations and hotels) and protecting only the requesting user's location privacy.[1–9]

This article addresses proximity detection, an emerging category of mobile geosocial networking services. As a typical example, the "nearby friend alert" notifies a user when a friend is nearby. With such services, users not only issue proximity queries but also serve as the query results, so we need privacy protection schemes on mutual parties—those querying and those being queried, and both are constantly moving. A few works have tackled this problem (see the "Related Work in Privacy-Aware Proximity Detection" sidebar), but there's still a lack of support for true proximity and no quantitative studies of detection accuracy with true proximity. Here, based on granule and hashing techniques similar to those described in the sidebar, we propose a proximity detection scheme for "nearby friend alerts" that can address service inaccuracy in a quantitative manner. Using this scheme, we also devise a client-side-location update scheme and server-side update handling procedure for continuous proximity detection.

Nearby Friend Alert: A New Quantitative Solution

Figure 1 demonstrates the nearby friend alert, which notifies a user when a friend (someone in the same social group, such as a family member, classmate, or colleague) is in close proximity—that is, within a (user-specified) distance. In this case, the "proximity" has been set as within 500 meters of the Manhattan distance (the sum of distances in each dimension), and all members in the group have agreed on this distance. Alert messages go on or off the screens as two friends

Published by the IEEE CS ■ 1536-1268/13/$31.00 © 2013 IEEE

Related Work in Privacy-Aware Proximity Detection

The new service paradigm calls for privacy protection schemes on mutual parities—those querying and those being queried, and both are constantly moving. In recent literature, a few works tackle this problem under different system models and proximity definitions.

Serigio Mascetti and his colleagues assumed a trusted service provider and proposed a solution called "Hide&Crypt"—a filter-and-refine two-phase procedure.[1] In the first phase, to protect location privacy, all users cloak their locations in a set of granules (grid cells) and send them to the service provider, who then filters out those pairs that are confirmed as nonresult pairs. In the second phase, the service provider notifies users of uncertain pairs to refine the proximity result. Specifically, each pair of users employs the finer granularity of both users and start a secure two-party set-inclusion protocol on the resulting sets of granules to detect whether they're actually in proximity.

In a later work, Mascetti and his colleagues further relaxed the requirement of a trusted service provider and proposed two solutions for the same proximity detection problem.[2] The first one, Hide&Seek, treats the service provider as an untrusted hub where users store their encrypted granules for the querying user to retrieve and detect proximity. Specifically, the querying user retrieves the encrypted granules, decrypts them, and computes his or her minimum distance to these granules to detect proximity. As such, in Hide&Seek the querying user knows the exact granule of the other users. The second solution, Hide&Hash, further protects this exact granule from the querying user by hashing it instead of using encryption. Then, as in Hide&Crypt, the querying user starts a secure set-inclusion protocol with the service provider to detect whether any other user is in proximity.

Assuming a nontrusted service provider, Laurynas Šikšnys and his colleagues proposed the FriendLocator system,[3] which is the first work that applies the "grid-and-hashing" techniques. The major difference from Hide&Hash is that FriendLocator employs a global (instead of user-specific) grid partitioning, which helps eliminate the two-party protocol. Furthermore, a multilevel granule system is designed to reduce location updates when users are far from their friends. They further proposed VicinityLocator, which differs from the other systems by enabling user-defined proximity criterion.[4] Specifically, it lets the user specify his or her area of interest as the "vicinity region"; a friend is in proximity only if the friend is located in this region.

However, due to the complexity and high cost of the detection process, these works tend to adapt the original definition of proximity to align with the grid settings. In all of the "Hide" approaches,[1,2] the minimum possible distance between two users is used for proximity detection. As such, if two users are in the same granule, they're considered in proximity because their minimum possible distance is 0. Similarly, in FriendLocator and VicinityLocator,[3,4] the authors allow proximity to be defined fuzzily in uncertain cases. In FriendLocator, there are essentially two distance thresholds, d and $d + \lambda$—users are strictly in proximity within d and strictly not in proximity beyond $d + \lambda$. However, the relation for distances between these two values is decided arbitrarily; or, as quoted from the authors, "on a proximity notification, the actual distance between the two users can be in the range from d to $d + \lambda$," where $\lambda = d(2\sqrt{2} - 1)$ is a nonuser-configurable system constant. Likewise, in VicinityLocator, if the minimum distance between the user's location and the vicinity region is shorter than λ (but larger than 0), the proximity relation will be decided arbitrarily.

REFERENCES

1. S. Mascetti et al., "Privacy-Aware Proximity Based Services," *Proc. 10th Int'l Conf. Mobile Data Management: Systems, Services and Middleware* (MDM 09), IEEE CS, 2009, pp. 31–40.
2. S. Mascetti et al., "Privacy in Geo-Social Networks: Proximity Notification with Untrusted Service Providers and Curious Buddies," *VLDB J.*, vol. 20, no. 4, 2011, pp. 541–566.
3. L. Šikšnys et al., "A Location Privacy Aware Friend Locator," *Advances in Spatial and Temporal Databases*, LNCS 5644, Springer, 2009, pp. 405–410.
4. L. Šikšnys et al., "Private and Flexible Proximity Detection in Mobile Social Networks," *Proc. 11th Int'l Conf. Mobile Data Management* (MDM 10), IEEE CS, 2010, pp. 75–84.

move toward or away from each other, so the service is running continuously.

Our proposed scheme eliminates false-positive cases by setting an appropriate grid size. To minimize false-negative detection, we propose grid overlay—a set of independent grids—and study the optimal placement of these grids. The key advantage of this scheme over existing works is that users can dynamically trade accuracy for communication cost (or vice versa) in a quantitative manner by adding (or removing) grids. This advantage is particularly critical in mobile environments, where bandwidth and battery conditions can change drastically.

To enhance the real-time performance of this scheme for large-scale systems, we also developed a multilevel grid scheme, inspired by the FriendLocator system,[10] that can significantly reduce the user location update frequency and thus the server load. Based on this design, we devised client-side location update

Figure 1. The "nearby friend alert" service. Alert messages go on or off the screens as two friends move toward or away from each other, so the service is running continuously.

and server-side update handling procedures.

Static Proximity Detection

First, we ignore user mobility and study how the service provider can detect that two stationary users, u and v, are near each other. In this process, the coordinates of user u shouldn't be disclosed to user v or the service provider—the only information provided is that "u is in proximity." For ease of presentation, we assume all parties (the service provider and mobile users) are noncolluding and follow a semihonest model (in which all parties follow the designated protocol properly, except that they can record intermediate results and try to deduce location information about other users).

Let's assume Manhattan distance and δ denote the distance threshold for proximity. (Throughout the article, we use the popular Manhattan distance metric because it's more favorable than the Euclidean distance in urban areas, where this service is needed most.) Let (u_x, u_y) denote u's coordinates, so the distance between u and v is defined as

$$Dist((u_x, u_y), (v_x, v_y)) = |u_x - v_x| + |u_y - v_y|.$$

The binary relation "in proximity" of two users, u and v, is true if and only if they're within δ meters. Formally, $in\_proximity(u, v) := Dist((u_x, u_y), (v_x, v_y)) \le \delta$.

Location Anonymity

The basic idea of proximity detection without disclosing user locations is to apply a one-way space transformation that preserves proximity, and the service provider then detects proximity in the transformed space. Because the transformation is one way, the service provider can't infer the original user locations. We adopt the same "grid-and-hashing" transformation used elsewhere,[10–12] which imposes a uniform grid G that partitions the space into cells. Note that the placement of G is a secret shared by users in the same group and is unknown to the service provider.

For each user u of (u_x, u_y), his or her cell is indexed by $(G^x(u_x), G^y(u_y))$. The user then encodes this index using a one-way hashing function Γ and sends the hash value $\Gamma((G^x(u_x), G^y(u_y)))$ to the service provider. We call this hash value the *signature of u* with regard to grid G, denoted by $G(u)$. To detect the proximity of two users, the service provider simply tests the equality of their uploaded signatures. To prevent reverse engineering of the cell and false positive results, Γ should be implemented by a keyed cryptographic (collision-free) hash function, such as SHA-256. The key is distributed among users in the same group and is dynamically changing (by rehashing) against a service provider's inference.

Unfortunately, an arbitrary grid G doesn't necessarily preserve the proximity relation—the choice of cell size is critical. On the one hand, if the cell's half diagonal length, l_c, is less than $\delta/2$, the maximum Manhattan distance between two points in the same cell exceeds δ, creating "false-positive" cases in which two users are incorrectly detected as "in proximity." On the other hand, even if l_c is arbitrarily small, two in-proximity users might still be separated in an adjacent cell and thus could have different signatures. These are the "false-negative" cases. For example, in Figure 2a, u and v are within δ meters, but they're not in the same cell. In fact, to be in proximity with u, v can be any point in the shaded region—called the *proximity region of u*. The region is a square centered at u with its half diagonal length equal to δ.

To study how frequently the false-negative cases occur, we now derive $P(FN|in\_proximity)$ and $P(TP|in\_proximity)$, the conditional probabilities of false-negative and true-positive detection, respectively, given that the actual relation is "in proximity." Let's assume users are distributed uniformly in the space. Obviously, the latter probability equals the proportion of one

cell to the entire proximity region (see Figure 2a). Formally,

$$P(TP \mid in\_proximity) = 1 - P(FN \mid in\_proximity) = \max\{1, (l_c/\delta)^2\}. \quad (1)$$

Consequently, the larger the cell (l_c), the fewer the false-negatives. However, as noted earlier, when $l_c > \delta/2$, increasing l_c will lead to more false-positive cases. We thus decided to make l_c equal to $\delta/2$, eliminating false-positives while keeping the minimum number of false-negative cases.

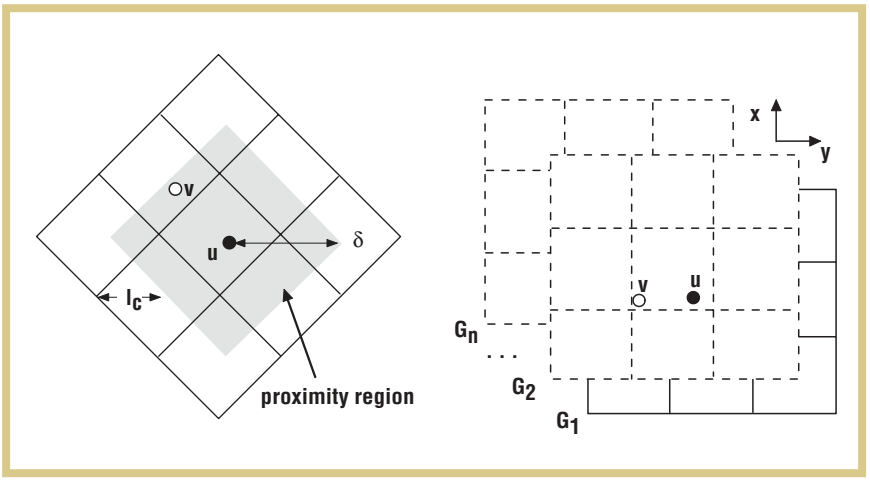


Figure 2. Static proximity detection: (a) the proximity region, where users u and v are within δ meters but aren't in the same cell; (b) the grid overlay imposes more than one grid partition $G_1, \ldots, G_n$. Each grid is created from the previous one by shifting both the x-axis and the y-axis, in the hope that a false-negative case in one grid can be successfully detected in another grid.

Grid Overlay

According to Equation 1, the false-negative probability of the standard "grid-and-hashing" approach is 75 percent when $l_c = \delta/2$. To further reduce this, we propose the grid overlay scheme, in which we impose more than one grid partition $G_1, \ldots, G_n$, as Figure 2b shows. For ease of presentation, we rotate the coordinate system by 45 degrees, so all grid lines are either horizontal or vertical. Each grid is created from the previous one by shifting both the x-axis and the y-axis, in the hope that a false-negative case in one grid (for example, u and v in G_1) can be successfully detected in another grid (such as G_2). As such, the proximity detection of u and v should test the equality of their signatures with regard to all grids—that is, $G_1(u)$ with $G_1(v)$, $G_2(u)$ with $G_2(v), \ldots$, and $G_n(u)$ with $G_n(v)$. Users u and v are in proximity as long as one of these pairs is equal.

We have yet to determine the number of overlay grids (n) and how they're placed. A straightforward approach is to generate these grids by random shifting. Figure 3a illustrates user u's proximity region (the light grey area). The solid boxes are the cells that cover u in grids $G_1, G_2, \ldots, G_n$. Note that their top left corners must reside in the top left quadrant (the dark grey area) of u's proximity region. The probability of false-negatives $P(FN|in\_proximity)$ is the proportion of the proximity region that's not covered by any of the n covering cells.

To calculate this area, let's focus on the top left quadrant (the dark grey area). At point $v = (x, y)$, the probability of v not being covered is $(1 - (2xy/\delta^2))^n$. Integrating both x and y and multiplying by four quadrants, we obtain the expected probability of false-negatives for n random-shifting grids:

$$P(FN \mid in\_proximity) = \frac{1}{n+1}\sum_{i=1}^{n+1}\frac{(-1)^n}{i}\binom{n+1}{i}. \quad (2)$$

The second column in Table 1 shows the change in this (random placement) value as the number of grids (n) increases. Unfortunately, the value doesn't drop significantly, even when n grows exponentially.

To reduce this value, we study the optimal placement of a new grid G' on top of n existing grids. We start with $n = 1$. Figure 3b illustrates the placement of G' and the existing G. Because each dimension is equal, we assume G' deviates t from G in each dimension. Without loss of generality, we assume u is in the upper right cell of G. The area of true-positive cases for v is the union of the two covering cells of u in G and G'. Therefore, minimizing $P(FN \mid in\_proximity)$ is equivalent to minimizing their overlapping area. In this figure, where $u_x > t$ and $u_y > t$, the overlapping area is the light grey area; if $u_x < t$ and $u_y < t$ (such as u'), this area is the dark grey area, and so on.

In any case, this area achieves the minimum when $t = \delta/4$. This same rationale applies to $n > 2$. Suppose $n = 2k$ ($k > 1$). The optimal grid overlay is created by shifting the first grid G in both diagonals for every $1/k$ of the cell length. Figure 3c shows the case of $k = 4$. Then the false-negative probability is

$$P(FN \mid in\_proximity) = \frac{2n-1}{n^2}. \quad (3)$$

Note that the two diagonals share one duplicate grid (for example, $G1$ and $G5$); if k is odd, they share one more duplicate grid in the midpoint (for example, $G3$ and $G7$). As such, the actual number of grids needed is either $n - 1$ or $n - 2$. For simplicity, we omit this saving of grids throughput this article.

In practice, the service provider should have a service-level agreement with the subscribers in this nearby

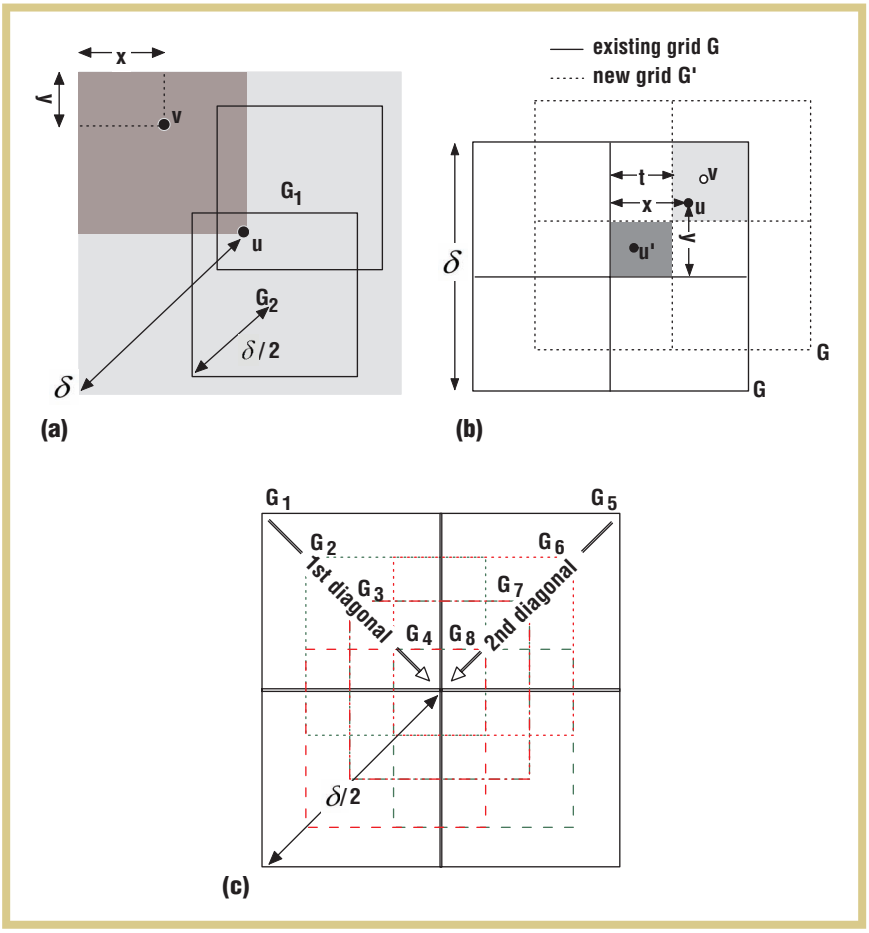


Figure 3. Placement of grids: (a) user *u*'s proximity region (the light grey area) and two randomly placed grids (G_1 and G_2); (b) the optimal placement of a new grid G' (number of grids (n) = 2); and (c) the optimal grid overlay is created by shifting the first grid G in both diagonals for every $1/k$ of the cell length ($n = 2k - 2$, where k is less than 1).

TABLE 1
The conditional probabilities of false-negative detection of random and optimal grid placements.

No. of grids (n)	Random placement (%)	Optimal placement (%)
2	61.1	43.8
8	31.4	23.4
16	20.2	12.1
32	12.4	6.15
64	7.32	3.10
128	4.22	1.26
256	2.39	0.78

friend-alert service, which declares a threshold value τ for the expected false-negative probability $P(FN|in\_proximity)$. The number of grids needed can then be derived from Equation 3. Table 1 shows the numerical values side by side with those of the random placement. We observe that the optimal placement achieves even lower false-negative probability with only half the number of random grids.

Dynamic Proximity Monitoring with Location Anonymity

We now incorporate user mobility and study how the service provider continuously monitors the in-proximity result, alerting two users when they're moving closer than the proximity threshold δ and canceling the alert when they move apart. The continuous monitoring involves user signature update and service provider query reevaluation.

A naïve approach is that the user updates any signature of the grid overlay whenever it changes. Once the service provider receives the updated signature, it executes the proximity detection procedure outlined earlier—that is, it performs a signature equality test against existing signatures from other users in the same group. Obviously, this approach is extremely costly, because the overlay might contain dozens of grids and thus lead to frequent signature updates. However, updates for nonproximity pairs can be less frequent than updates for in-proximity pairs.

To capture this, we designed a multilevel grid overlay hierarchy,[10] as shown in Figure 4. The lowest level (level 0) grid overlay is the original grid overlay for the static case. Recall that the cell size (in terms of the half-diagonal length l_c) of any grid in this overlay is $\delta/2$. This level of grid overlay is used for in-proximity pairs, while all upper levels are used for nonproximity pairs. In any upper level (level 1, level 2, ...), the cell size is m times that of its immediate lower level (level 0, level 1, ...) until, in the uppermost level (level L), a single cell covers the entire space. The scaling factor m is a system parameter that should be based on the user's mobility pattern. In general, the farther users can travel in a given time period, the larger m should be. In Figure 4, $m = 3$.

Note that for clarity, in Figure 4 we only plot one grid in every level. The number of grids needed in the level 0 grid overlay should be calculated from the false-negative probability threshold τ. The lower τ is, the more grids needed. Similarly, each overlay in the upper levels should have a sufficient number of grids to guard all nonproximity pairs with the same false-negative probability threshold τ. In the ith level, the

expected false negative probability can be generalized as

$$P(FN \mid in\_proximity) = \frac{2n-1}{m^{2i}n^2}.$$

Signature Update

Given this hierarchy, the signature update scheme for user u is as follows. If user v is already in proximity, then u and v must have a common signature set (CSS) at level 0, denoted $CSS_0(u, v)$. As such, u should update when any signature in $CSS_0(u, v)$ changes. (Here, we require "any," not "all," because v could face the same update decision without knowing u's decision.)

On the other hand, if v isn't in proximity, then u and v don't have a CSS at level 0, level 1, ..., until—at some level $i (i > 0)$—their CSS is no longer empty. The level $i - 1$ is called the *uppermost noncommon level* (UNL) of u and v. For example, in Figure 4, $UNL(u, v) = 1$, because they have a CSS in level 2. Although any level below UNL also guarantees the nonproximity relation, UNL is the highest such level and thus has the largest cell size to reduce signature updates. Therefore, user u only needs to store the whole signature set (SS) in level $UNL(u, v)$ and update when any such signature is changed.

User u's update decision should be based on all v users in the same social group. To summarize, u should update when there's any signature change in any in-proximity v's CSS_0 or any nonproximity v's $SS_{UNL}(u, v)$. As such, throughout the service running time, the user u must maintain either $CSS_0(u, v)$ or $UNL(u, v)$ of all v users. To reduce the set of signatures to be checked, for nonproximity v users, we apply a unified UNL instead of treating each v independently. $UNL(u)$ is defined as the lowest $UNL(u, v)$ of all nonproximity v users.

Query Reevaluation

Upon receiving an update of signatures, the service provider attempts to reevaluate the in-proximity relation of u and every v in the same group. If u and v were in proximity, the service provider checks if there's any update in $CSS_0(u, v)$. If there's no such update, or if the updated $CSS_0(u, v)$ is still nonempty, then u and v are still in proximity. Otherwise, they're no longer in proximity.

To calculate $UNL(u, v)$, the service provider probes v for updated signatures in all levels and replies to both with the $UNL(u, v)$. If u and v weren't in proximity, the service provider checks if there's any update in $SS_{UNL}(u,v)$. If there's no such update, or if the updated $CSS_{UNL}(u, v)$ is still empty, then u and v are still nonproximity, and nothing else needs to be done. Otherwise, the service provider probes v for updated signatures in all levels to calculate the new relation of u and v. If they're now in proximity, the service provider replies to both with $CSS_0(u, v)$; otherwise, the service provider replies to both with $UNL(u, v)$. Note that throughout the service running time, the service provider must maintain the same data structure as the users—that is, $CSS_0(u, v)$ or $UNL(u, v)$ of all (u, v) pairs.

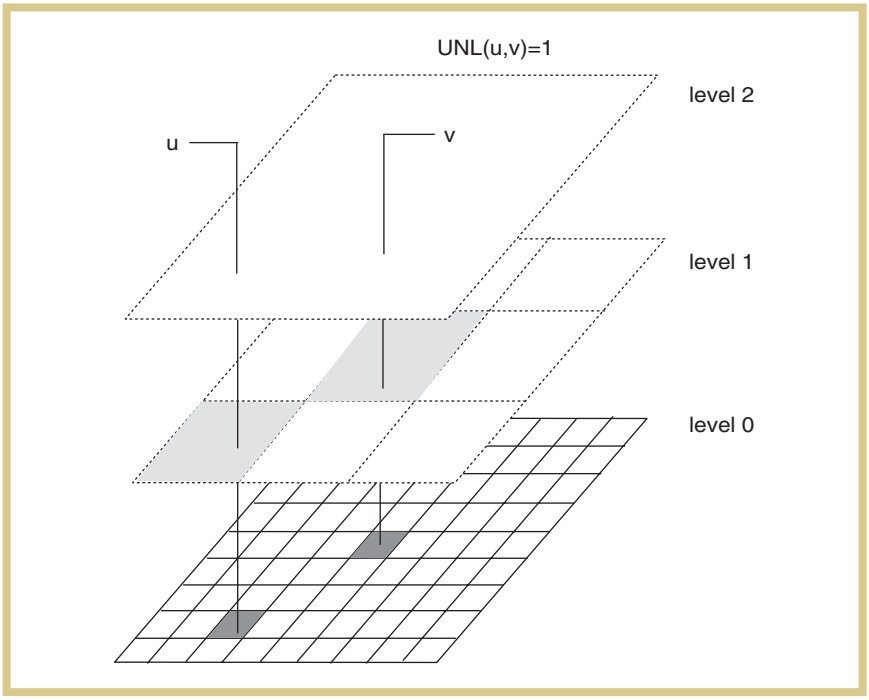


Figure 4. The grid hierarchy and uppermost noncommon level (UNL). The lowest level (level 0) grid overlay is the original grid overlay for the static case. In any upper level (level 1, level 2, ...), the cell size is *m* times that of its immediate lower level (level 0, level 1, ...) until, in the uppermost level (level L), a single cell covers the entire space.

Figure 5 illustrates an example with users a, b, and c. For simplicity, we assume only one grid in each level and $m = 2$. Initially, none of the users are in proximity with each other, and their mutual UNLs are listed in the top table. When a moves to a', because only the signature of level 0 changes, and $UNL(a) = 1$, a doesn't need to update. When b moves to b', because the signature of level 1 is changed, and $UNL(b) = 1$, b must update its signatures.

To reevaluate the relation of a and b, a is probed. Because $CSS_0(a, b) = \{sig(x)\} = \emptyset$, a and b are in proximity and both will receive $\{sig(x)\}$ as the reply. When c moves to c', because the signature of level 2 changes, and $UNL(c) = 2$, c must update its signatures. Both a and b are probed to reevaluate their relations with c, both of

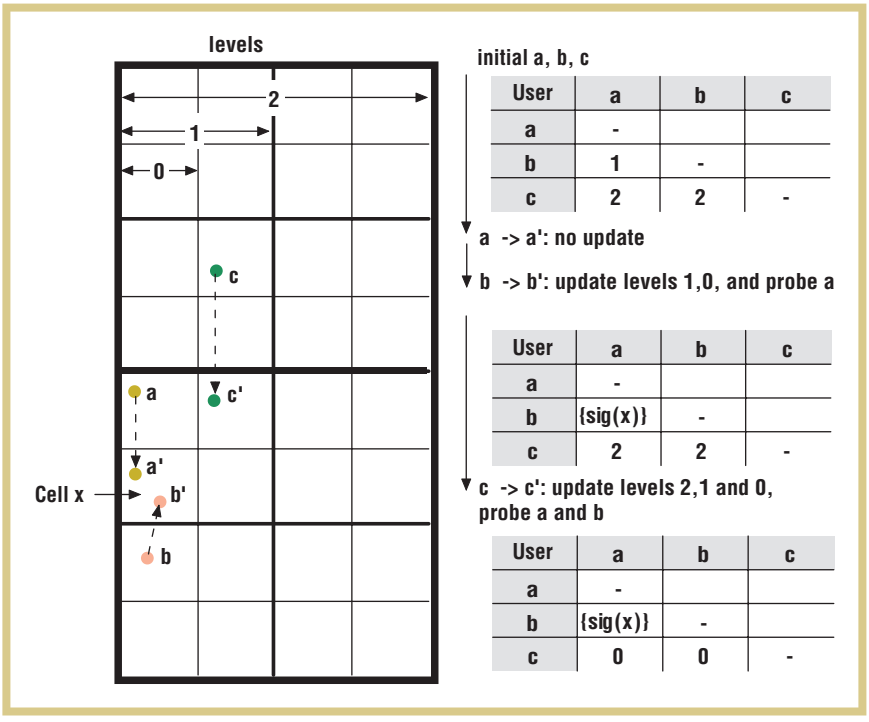


Figure 5. Signature update and query reevaluation for users *a*, *b*, and *c*. For simplicity, we assume only one grid in each of the three levels, and the multigrid scaling factor is two. Initially, none of the users are in proximity, and their mutual UNLs are listed in the top table. When *b* moves to *b*′, because the signature of level 1 changes, and UNL(*b*) = 1, *b* must update its signatures (middle table). When *c* moves to *c*′, because the signature of level 2 changes, and UNL(*c*) = 2, *c* must update its signatures (bottom table).

TABLE 2
Parameter settings for experiments.

Parameter	Symbol	Value
Number of users	N	50,000
Average group size	g_size	50
Average moving speed	v	10 meters per second
Proximity threshold	δ	100 meters
Multigrid scaling factor	m	3
Number of grids	n	32

which are nonproximity. They both receive the updated UNL = 0 as the reply.

Empirical Results

We now evaluate the performance of our proposed solution. The data is generated from the moving object generator[13] based on the German city Oldenburg, which has been widely used in related studies.[10–12] The dataset covers an area of 14 × 12.26 km^2 and contains 50,000 users. The generator creates the location of each user in every timestamp. The users are divided into social network groups with an average size of 50. Table 2 summarizes default system parameter settings.

The code of our experiments is implemented in C#. The multithreaded simulation testbed is executed on a desktop computer with Intel Core2 Quad processor Q6600 and 4 Gbytes of RAM, running the Windows XP 32-bit edition. For the cryptographic hashing algorithm, we use SHA-256, which generates digests of 256 bits. For performance evaluation, we measure the communication cost in terms of the number of bytes exchanged per timestamp, per user. To evaluate the system scalability, we also measure the server CPU time per timestamp.

In the first experiment, we verify the proposed optimal grid placement. We use a single-level overlay, varying the number of grids and measuring the false negative rate. Figure 6a plots the values of both random and optimal grid placement, together with their expected values as derived from the settings in Table 2. The experiment values align well with the expected values, which verifies our claim that the optimal placement saves more than 50 percent of the grids compared to random placement. For the following experiments, we consistently use 32 grids, where the false negative rate is approximately 6 percent. This should be low enough for most users.

The second experiment tests the impact of proximity threshold δ. We vary δ from 400 to 25 meters and measure the communication cost and server CPU time. Figure 6b plots the results. We observe that even though δ drops to 1/16 from 400 to 25—that is, more users become in proximity—the communication cost only increases by approximately 65 percent, and the server CPU time is barely affected. This is attributed to the multilevel grid overlay and the efficient signature update scheme, which greatly reduces the update and reevaluation costs.

The third experiment tests the scalability of the proposed solution. We increase the number of users (N) from 1,000 to 100,000 and the average group size from 10 to 90. Figure 6c plots the server CPU time with regard to N, which increases linearly as N increases. Furthermore, because proximity

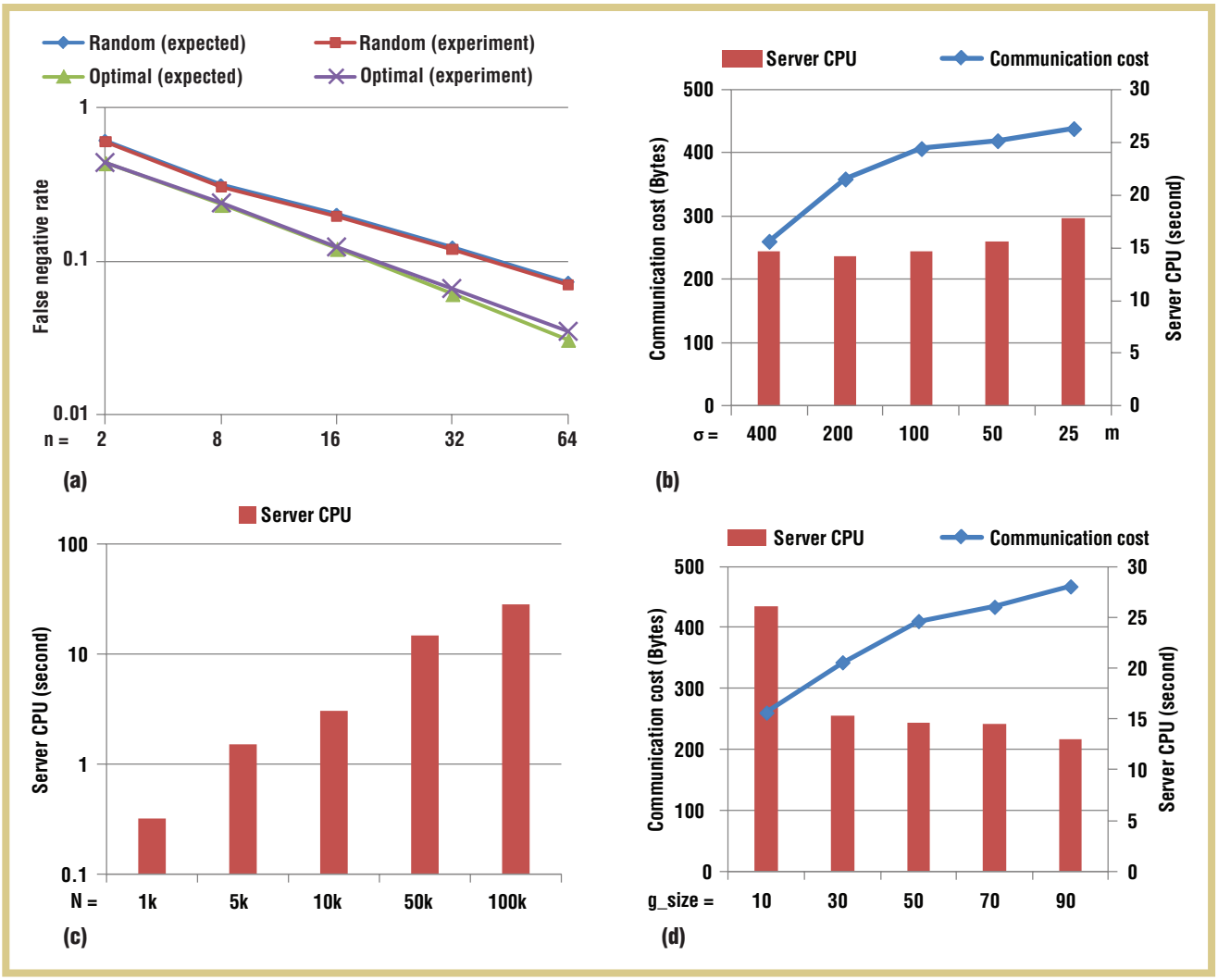


Figure 6. Performance results versus system parameters: (a) the false positive rate versus the number of grids, (b) performance versus the proximity threshold, (c) server CPU versus the number of users, and (d) performance versus the average group size.

detection is independent for different social groups, the system can be easily parallelized by delegating the execution to different processing cores. As such, the server CPU time can always be reduced as needed. Figure 6d plots the results with regard to the group size (g). As g increases, the communication cost increases much more slowly, thanks to the multilevel signature update scheme that keeps most nonproximity pairs in the upper levels. As an indicator, even the server CPU time drops as the group size increases. These results show that the system scales well to a large number of users or large group size.

Under the umbrella of the grid-and-hashing paradigm, our optimal grid overlay and multilevel grids techniques significantly increase detection accuracy while saving the wireless bandwidth. Based on these two techniques, we devised the client-side location update scheme and the server-side update handling procedure for continuous proximity detection.

We're working to extend this solution to the general L^p norm distance, where the Manhattan distance is its special case ($p = 1$). This distance metric could facilitate a more versatile set of geosocial services, where contexts of distance are different, such as location-based games, virtual reality, and location-aware advertising.

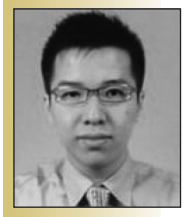
ACKNOWLEDGMENTS

This work was supported by Research Grants Council, Hong Kong SAR, China, under projects HKBU 210811, 210612, 211212, FRG2/11-12/140, and 11-12/074.

REFERENCES

1. A. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003, pp. 46–55.

the AUTHORS



Hong Ping Li is an MPhil student in the Database and Information Management Research Group (www.comp.hkbu.edu.hk/db) at Hong Kong Baptist University. His research interests are mobile data management and location privacy protection. Contact him at hpli@comp.hkbu.edu.hk.

Haibo Hu is a research assistant professor in the Department of Computer Science at Hong Kong Baptist University. His research interests include mobile and wireless data management, location-based services, and privacy-aware computing. Hu received his PhD in computer science from the Hong Kong University of Science and Technology. He's a member of ACM. Contact him at haibo@comp.hkbu.edu.hk.

Jianliang Xu is an associate professor in the Department of Computer Science at Hong Kong Baptist University. His research interests include data management, mobile and pervasive computing, and distributed and networked systems. Xu has a PhD in computer science from Hong Kong University of Science and Technology. He's a senior member of IEEE. Contact him at xujl@comp.hkbu.edu.hk.

2. G. Myles, A. Friday, and N. Davies, "Preserving Privacy in Environments with Location-Based Applications," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003, pp. 56–64.

3. B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," *Proc. 25th IEEE Int'l Conf. Distributed Computing Systems* (ICDCS 05), IEEE, 2005, pp. 620–629.

4. G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Prive: Anonymous Location-Based Queries in Distributed Mobile Systems," *Proc. 16th Int'l Conf. World Wide Web* (WWW 07), ACM, 2007, pp. 371–380.

5. A. Khoshgozaran and C. Shahabi, "Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy," *Advances in Spatial and Temporal Databases*, LNCS 4605, Springer,

6. X. Pan, J. Xu, and X. Meng, "Protecting Location Privacy Against Location-Dependent Attack in Mobile Services," *Proc. 17th ACM Conf. Information and Knowledge Management*, ACM, 2008, pp. 1475–1476.

7. T. Xu and Y. Cai, "Location Anonymity in Continuous Location-Based Services," *Proc. ACM Int'l. Symp. Geographic Information Systems* (GIS 07), ACM, 2007, pp. 300–307.

8. H. Hu and J. Xu, "Non-Exposure Location Anonymity," *Proc. IEEE 25th Int'l Conf. Data Eng.* (ICDE 09), IEEE, 2009, pp. 1120–1131.

9. C.-Y. Chow, M.F. Mokbel, and W.G. Aref, "Casper*: Query Processing for Location Services without Compromising Privacy," *ACM Trans. Database Systems*, vol. 34, no. 4, 2009, article 24.

10. L. Šikšnys et al., "A Location Privacy Aware Friend Locator," *Advances in Spatial and Temporal Databases*, LNCS 5644, Springer, 2009, pp. 405–410.

11. S. Mascetti et al., "Privacy in Geo-Social Networks: Proximity Notification with Untrusted Service Providers and Curious Buddies," *VLDB J.*, vol. 20, no. 4, 2011, pp. 541–566.

12. L. Šikšnys et al., "Private and Flexible Proximity Detection in Mobile Social Networks," *Proc. 11th Int'l Conf. Mobile Data Management* (MDM 10), IEEE CS, 2010, pp. 75–84.

13. T. Brinkhoff and O. Str, "A Framework for Generating Network-Based Moving Objects," *Geoinformatica*, vol. 6, no. 2, 2002, pp. 153–180.