Bloom-Filter-Based Profile Matching for Proximity-based Mobile Social Networking

Kun Xie [1], Xin Wang[2], Wei Li[1], Zhe Zheng[1], Gaogang Xie[3], Jigang Wen[3]
[1] College of Computer Science and Electronics Engineering, Hunan University, Changsha, 410082, China
[2] Department of Electrical and Computer Engineering, State University of New York at Stony Brook
[3] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
xiekun@hnu.edu.cn, x.wang@stonybrook.edu, rj_wli@hnu.edu.cn, zhezheng@hnu.edu.cn, xie@ict.ac.cn, wenjigang@ict.ac.cn

***Abstract*—The popularity of smart phones fosters the growth of Proximity-based Mobile Social Networking (PMSN). Although some profile matching approaches have been proposed to facilitate a user to find another user that shares his/her interest in the proximity, these approaches usually model the matching problem as a Private Set Intersection problem or a Private Set Intersection Cardinality problem and require high complexity of computation. Different from current studies, to facilitate more effective building of PMSNs, we propose a novel similarity metric to evaluate the common interests of mobile users by considering the time-dependent features of their interests. To calculate the metric in a low cost and privacy-protection way, we propose a novel time-dependent bloom filter to encode the time-dependent interest and a novel probabilistic algorithm to estimate the time-dependent similarity metric based on the bloom filter. We prove the high accuracy of the probabilistic algorithm both theoretically and through simulations. Based on the proposed BF-based profile matching approach, we further propose InterestMatch, a novel distributed mobile communication system to facilitate more efficient social networking among strangers in the physical proximity. We have done extensive experiments on real-world phones, our experiment results demonstrate that our approach is promising for facilitating mobile social interactions in the physical proximity due to its low complexity and consequently low power consumption.**

I. Introduction

Proximity-based mobile social networking (PMSN) refers to the interactions between mobile users in the close proximity through the Bluetooth/WiFi interfaces of their mobile devices. The popularity of smart phones (such as iphone, ipad) fosters the growth of PMSN. In contrast to traditional web-based online social networking, PMSN can enable more tangible face-to-face social interactions in public places such as cafeterias and airports. For example, E-SmallTalker [24] is a typical PMSN application that connects a person with the nearby person before they start a small-talk based on common interests.

As an important step toward effective PMSN, profile matching is often needed to facilitate a user to find the other users whose profiles best match his/hers. In many applications, the user personal profiles may contain sensitive information that they do not want to make public. Thus, a major challenge for profile matching is to protect the privacy of personal profiles. To address the issue, several private profile matching approaches have been proposed recently [14]–[16], [20], [26]. Relying on a third party or depending on complex encryption and decryption algorithms, these approaches are susceptible to the single point of failure or large computation cost, thus are not suitable to run by energy-limited wireless devices.

As an important characteristic, in many applications, the data of user profiles are randomly generated from a large information space, which are impractical to enumerate. Instead of using traditional private profile matching approaches, we propose to apply bloom filter to realize the profile matching function while at the same time protecting the user privacy.

A bloom filter [1], [23] (BF) is an excellent compact data structure that can succinctly represent a data set in order to support membership queries, and filter out effectively any element that does not belong to the set. With the features of simplicity, ease of use, hardware support, and excellent performance, bloom filters have broad applications. Particularly, bloom filters can work efficiently in a distributed system where information and data need to be exchanged among different entities. To insert an element into the bloom filter, a set of functions are applied to hash this element to set the corresponding bits in the bloom filter. Since the hash is one-directional, the source element cannot be discovered from the bloom filter and the privacy of the element is protected.

Despite their usefulness, bloom filters are generally designed for testing the set membership, rather than for the similarity matching. The matching of the similarity generally requires the calculation of the cardinality of the set intersection. To complete this function, the bloom filter can be straightforwardly applied with the following three steps: The profile data are first encoded in a bloom filter, then the bloom filter is broadcast to other users, and finally a user can calculate the set intersection between his own profile and the one encoded in the incoming bloom filter [19], [24] by checking each element in the profile to determine if it is contained in the bloom filter.

The straight-forward BF approach, however, cannot work well in a practical social network. The social networks are often dynamic, and user interests can change with time. The recent profile data are thus often more important than those of the past when evaluating the mobile user interests. However, in almost

all the existing profile matching approaches, the similarity levels of user interests are determined by treating all the profile data equally. To facilitate more effective building of proximity-based mobile social networks, we propose to design a novel similarity metric to evaluate the common interests of mobile users by considering the time-dependency of their interests. Different from the straightforward BF approaches, we propose a novel bloom filter structure to support not only the time-dependent encoding of the profile data but also the calculation of the time-dependent similarity metric.

Developing such a bloom filter, however, is particularly challenging for the following three reasons. First, the original bloom filter uses bit arrays to keep track of the membership of elements. To record the time-dependent profile data rather than simple membership information, a static bit array is no longer sufficient. Second, to identify the common profile data with the traditional bit-array-based bloom filter, a user needs to query all the other users for their membership to find the set-intersection first. This introduces unnecessarily high computation cost when only the cardinality of the intersection is of interest. To reduce the power consumption, it needs an efficient solution to estimate the cardinality directly. Third, different from the traditional set-intersection calculation, an advanced time-dependent similarity estimation may require calculating the cardinality of both the set-intersection and the set-union of the time-dependent profile data, which is very difficult.

To address the challenges above, unlike the standard bloom filter and its various extensions which usually utilize bit arrays to keep track of the membership of elements, we propose a novel time-dependent bloom filter (TDBF) which utilizes dynamic cell instead of a static bit to store the time related information. Based on TDBF, we propose a novel probabilistic algorithm to estimate the cardinality of the set intersection and set union, which is further applied to the efficient matching of the time-dependent profiles.

Based on the BF-based profile matching approach, we propose InterestMatch, a novel distributed mobile communication system to facilitate more efficient social networking among strangers in the physical proximity. Users can simply exchange their TDBFs, and based on the estimated similarity results, a user can find the one whose interest matches his the most in the proximity. The nice features of bloom filter allow for light-weight information exchange among users and quick matching of user interests to find the best matched peers. The main contributions can be summarized as follows.

- We propose a novel time-dependent similarity metric to measure the similarity of profiles from different mobile users taking into account the time-delaying effect on user interests.
- To facilitate the low-cost exchange of profile information among users, we propose a novel time-dependent bloom filter (TDBF) which can record the weight of the time-dependent user interest.
- Based on TDBF, we propose a novel probabilistic algorithm, i.e., R-Recent Time-dependent similarity matching algorithm, to estimate the time-dependent similarity between mobile users.

Different from other BF-based profile matching approaches [19], [24] which involve a lot of hash calculations to determine the common interest, our similarity matching algorithm only needs the counting function operated over the BF string, which has a much lower computation cost. We also prove the high accuracy of the probabilistic algorithm both theoretically and through simulations.

To evaluate the performance of InterestMatch, we perform experiments on user cell phones. Our results show that InterestMatch can facilitate building PMSN with very low power consumption. Moreover, without relying on a trusted third party, a mobile user can carry out the similarity calculation and select the best matched user nearby distributively.

The rest of this paper is organized as follows. We introduce the related work in Section II, and present the system model and our time-dependent similarity metric in Section III. We propose the architecture of the time-dependent bloom filter and our algorithm to estimate the time-dependent similarity in Sections IV and V, respectively. We evaluate the performance in Section VI and conclude the work in Section VII.

II. Related work

In this section, we review related work and identify the differences between our work and existing work.

A. Profile matching in PMSNs

Profile matching is the most important function to support effective PMSN. A major challenge for profile matching is to ensure the privacy of personal profiles which often contain highly sensitive information. This challenge necessitates the private matching, in which two users compare their personal profiles without disclosing the information to each other. Private matching for PMSN has been recently addressed in [14]–[16], [20], [26]. Among which, assuming the existence of a semi-online central authority, Lu et. al. [16] proposed a symptom matching scheme for mobile health social networks. However, these approaches either realize private profile matching by employing a third party or depend on complex encryption and decryption algorithms. The need of centralized processing or complex encryption and decryption operations also makes these schemes prone to the single point of failure or large computation cost.

To reduce the computation and communication cost, recently, E-SmallTalker [24] and [19] design the profile matching approaches based on bloom filter, in which the profile data is encoded in bloom filter which is exchanged between users. By testing his profile data against the bloom filter, a mobile user can identify the common data. However, to find the common interest, the computation cost on hash calculation for membership query is still high and should be reduced.

To measure the similarity of mobile users, current profile matching approaches are constrained to finding the intersection or intersection cardinality of the user profiles. Thus, current private profile matching problem is usually modeled as Private

Set Intersection (PSI) [25] or Private Set Intersection Cardinality (PSI-CA) [7] problem by treating all profile data equally. As discussed in introduction, the interests of users are often time-dependent in practical applications, simply treating all user profile data equally is not suitable. Therefore, it requires a novel time-dependent similarity metric to measure the user matching degree, and further a profile matching algorithm to estimate the time-dependent similarity metric. However, all current profile matching approaches cannot easily be extended to solve the time-dependent profile matching problem, which is the focus of this paper.

Different from the current work, this paper proposes a novel time-dependent similarity metric and a BF-based probabilistic algorithm to estimate the similarity. Moreover, the probabilistic algorithm adopts the counting operations on bloom filter instead of membership testing to estimate the similarity, which can reduce the computation cost largely.

B. Bloom filter

A bloom filter (BF) [1], [23] is a space-efficient probabilistic data structure that supports set membership queries. Several BF variants have been proposed in the literature to suit various applications [1], [23] including compressed bloom filter [18], deletable bloom filter [21], hierarchical bloom filter [22], spectral bloom filter [6], bloomier filter [3], stable bloom filter [8], space code bloom filter [13], adaptive bloom filter [17], dynamic bloom filter [11], retouched bloom filter [9], and distance-sensitive bloom filter [12].

However, to support profile matching in this paper, we need the bloom filter to record the user interest, which requires not only the support of membership query but also have the ability of recording the time-dependent weight of user interest. Certainly, the bit array adopted in the standard bloom filter and its various extensions is not suitable for our problem. Although counting bloom filter [10] and [4] replace bits of a standard Bloom filter with counters to keep record of the appearance frequency of an element, it cannot be easily extended to record user's time-dependent interest weight.

Moreover, although bloom filters are widely used in various networking systems, such as Web proxies and caches, database servers, and routers, these applications only need the bloom filter to support the quick membership query. Different from current studies, to the best of our knowledge, this is the first work that proposes a BF-based probabilistic algorithm to estimate the similarity at the low cost in social networks.

III. System model and time-dependent similarity metric

Mobile devices can be equipped with the PMSN applications. Generally, a PMSN session consists of three phases. First, in the neighbor-discovery phase, users need to discover each other nearby. Second, in the matching phase, nearby users compare their personal profiles. Last, two matching users enter the interaction phase for real information exchange. Among all the three phases, profile matching is most critical and challenging for the wide use of mobile social networks.

To evaluate the similarity degree between two mobile users u_i and u_j, we first define our basic similarity metric based on the popular similarity criterion of Jaccard metric [2] as

$$\zeta_b(u_i, u_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \tag{1}$$

where S_i and S_j are the profile data sets of users u_i and u_j. The larger the intersection set, the higher the similarity level between two users. Values close to 1 suggest that two sets are very similar, whereas, those closer to 0 indicate that the interests of users u_i and u_j are almost disjoint.

The similarity metric in (1) treats all user interests equally, while in reality, user interests are often time-dependent with the interest weight reducing over time. To quantify and consider the interest decaying, we define a new time-dependent similarity metric by extending the metric in (1):

$$\zeta(u_i, u_j) = \sum_t (\zeta_b(u_i, u_j, t) \cdot (MAX \cdot g(\bar{t} - t))). \tag{2}$$

$\zeta_b(u_i, u_j, t)$ is the basic similarity metric calculated using the interest sets at time t as follows

$$\zeta_b(u_i, u_j, t) = \frac{|S_i(t) \cap S_j(t)|}{|S_i(t) \cup S_j(t)|}, \tag{3}$$

where $S_i(t)$ and $S_j(t)$ denote the interest sets at time t of user u_i and u_j.

In (2), the similarity at time t is weighted by $MAX \cdot g(\bar{t} - t)$, where $\bar{t}$ is the current time, MAX is the original weight of an interest item, and $g(\bar{t} - t)$ is a time-decayed function. We will discuss MAX and $g(\bar{t} - t)$ in more details in Section IV-B. In the similarity defined in (2), the recent interest profile has a higher impact on the similarity evaluation.

To design a profile matching approach based on bloom filter, we should solve the following two challenging problems: 1) How to design a novel bloom filter to encode the time-dependent interest of a user? 2) How to further design an algorithm to calculate the time-dependent similarity based on TDBF? We will present our key techniques next.

IV. Time-dependent bloom filter

In this section, we first review the standard bloom filter, and then introduce our new design of the time-dependent bloom filter to encode user's time-dependent interest.

A. Review of standard bloom filter

A bloom filter is used to represent a set $S = \{s_1, s_2, \cdots, s_n\}$ of n elements from a universe U. It can be represented as an m bit vector with elements $BF[0], BF[1], \cdots, BF[m-1]$, initially all set to 0. In many applications of bloom filters, the summary message is captured with m-bit vector BF and transmitted in the system. The filter uses k independent hash functions $h_1, h_2, \cdots, h_k$, with each independently mapping an element in the universe to a random number uniformly distributed within the range $0, 1, \cdots, m-1$. For each element $x \in$ S, the bits $BF[h_i(x)]$ are set to 1 for $1 \leq i \leq k$. To check the membership of an item y, we can examine the BF whether

its bits at positions $h_1(y), h_2(y), \cdots, h_k(y)$ are set to 1. If not, then y is definitely not a member of S. If all $h_i(y)$ $(1 \leq i \leq k)$ are set to 1, we assume that y is in S, although it can be wrong with some probability. Fig. 1 shows the operations of the standard bloom filter.

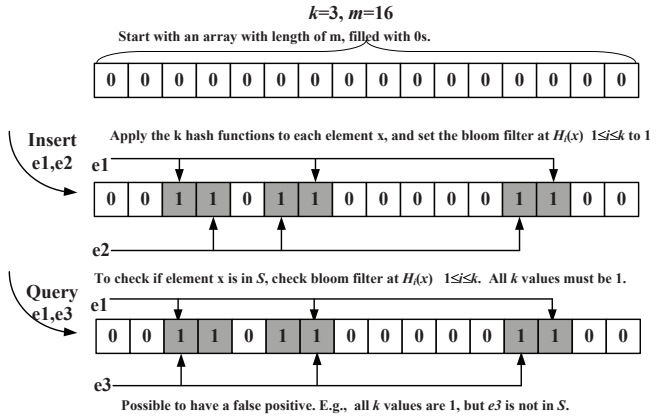


Fig. 1. Operations of the standard bloom filter.

As a probabilistic data structure, bloom filters are subject to false positives. That is, a BF can suggest that an element y is included in S even though it is not (e.g., $e3$ in Fig. 1). The false positive rate f is defined by the following formula:

$$f = \left(1 - (1 - 1/m)^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k \tag{4}$$

B. Time-dependent bloom filter

1) Problem

The interest profile of a user u_i can be recorded with time stamps, i.e,. $S_i = \{(a_{i,1}, t_{i,1}), (a_{i,2}, t_{i,2}), \cdots, (a_{i,j}, t_{i,j}), \cdots, (a_{i,n_i}, t_{i,n_i})\}$, where each $(a_{i,j}, t_{i,j})$ records the interest item $a_{i,j}$ of user u_i at the time $t_{i,j}$. Initially, the weight of each interest item is set to a large value (denoted by MAX). The item may have different meanings in different application contexts. For example, in a movie theater, we can utilize the film name as the item name.

As user interest often reduces as time goes by, we use a time decay function, a special non-increasing and non-negative function $g(\bar{t} - t)$, to represent the relative interest weight over the time. For an item $a_{i,j}$ with the weight $f(a_{i,j}, t)$ at time t, the decayed weight of the interest item $a_{i,j}$ at the current time $F(a_{i,j})$ can be calculated as $F(a_{i,j}) = f(a_{i,j}, t)\, g(\bar{t} - t)$.

Given the interest profile and the time decaying function, our goal is to design a time-dependent bloom filter structure to encode the time-dependent interest profile such that for a given interest item $a_{i,j}$, its time-decayed weight $F(a_{i,j})$ can be found.

2) Design

Different from the standard bloom filter and its various extensions which usually utilize bit arrays to keep track of the membership of elements, to record the time-dependent interest weight of an interest item, TDBF uses the counter cell instead of a bit with the cell value decaying over time.

Algorithm 1 describes the basic operations of our TDBF. Initially, all counters are set to 0 as shown on lines 1-3. To insert an item $a_{i,j}$ into TDBF, k hash functions are applied to this item and set each of the counters $TDBF[h_1(a_{i,j})]$, $TDBF[h_2(a_{i,j})]$, $\cdots$, $TDBF[h_k(a_{i,j})]$ to MAX (i.e., the initial interest weight), as shown on lines 4-6.

In [5], there is a variety of decaying functions, e.g., exponential decay, sliding window decay, polynomial decay, poly-exponential decay, and chordal and polygonal decay. Among which the exponentially time-decaying function is widely used in practice. Therefore, we design our time-decaying function as

$$g(\bar{t} - t) = \lambda^{\lceil \frac{\bar{t}}{T} \rceil - \lceil \frac{t}{T} \rceil}, \tag{5}$$

where T and λ are parameters that control the decaying speed of the weight. A time period of T time units is referred to as an epoch of the decaying function, which controls the granularity of time-dependence. The parameter $\lambda \in [0, 1]$ is the exponential decaying factor to control the speed of decaying, $\lceil * \rceil$ is a ceiling function and $\lceil x \rceil$ is the smallest integer not less than x.

On lines 7-9, to record the time-dependent interest weight, when a new epoch starts, all counter values in the bloom filter are decayed by applying $TDBF[i] = \lambda \times TDBF[i]$, where λ is the exponentially decaying factor in the function (5).

When querying the interest weight of item q, we can apply the k hash functions to the item. Among the k counters $TDBF[h_1(q)]$, $TDBF[h_2(q)]$, $\cdots$, $TDBF[h_k(q)]$, the minimum value is returned as the interest weight, as shown on line 10. As a cell may be set to the MAX value by a latter arriving item which is mapped to the same position, so we use the minimum value as the query response to avoid this impact.

Algorithm 1 Basic Operations of Time-dependent Bloom Filter

Input: Interest profile $S_i = \{(a_{i,1}, t_{i,1}), (a_{i,2}, t_{i,2}), \cdots, (a_{i,j}, t_{i,j}), \cdots, (a_{i,n_i}, t_{i,n_i})\}$.
Output: A TDBF to encode the interest profile.

Initialization Operation
1: **for** $0 \leq p \leq m - 1$ **do**
2: $\quad TDBF[p] = 0$
3: **end for**
Set Operation (on arrival of a new item $a_{i,j}$)
4: **for** $1 \leq p \leq k$ **do**
5: $\quad TDBF[h_p(a_{i,j})] = MAX$
6: **end for**
Decay Operation (on start of a new epoch)
7: **for** $0 \leq p \leq m - 1$ **do**
8: $\quad TDBF[p] = \lambda \times TDBF[p]$
9: **end for**
Query Operation (query q's time-dependent weight)
10: return $Min\{TDBF[h_1(q)], TDBF[h_2(q)], \cdots, TDBF[h_k(q)]\}$

Although TDBF is designed based on an exponentially time-decaying function, it can be easily extended to support other types of time-decaying functions. In our simulation part, we evaluate the performance of our proposed TDBF by utilizing both exponentially time-decaying function and linearly time-decaying function.

Fig. 2 shows an example to encode the interest profile of a mobile user $\{\{e_1,1\}\{e_2,2\},\{e_3,4\},\{e_4,5\},\{e_5,7\}\}$ using a TDBF ($k=3$, $MAX=5$, $\lambda=0.8$, $m=16$, $T=3$). Obviously, interest items $\{e_1,1\}$, $\{e_2,2\}$ arrive in the first time epoch, items $\{e_3,4\}$, $\{e_4,5\}$ arrive in the second time epoch, and the item $\{e_5,7\}$ arrives in the third time epoch. At the start of the second and third time epoches, all the counters in the filter decay. When we query the decayed-weight of item e_1 at $time=8$ against the bloom filter, we can obtain the value 3.2, which is equal to $\lambda^{\lceil \frac{\bar{t}}{T} \rceil - \lceil \frac{t_{e_1}}{T} \rceil} \times MAX = 0.8^{\lceil \frac{8}{3} \rceil - \lceil \frac{1}{3} \rceil} \times 5 = 3.2$. This result demonstrates that our TDBF can effectively track the time-dependent interest weight.

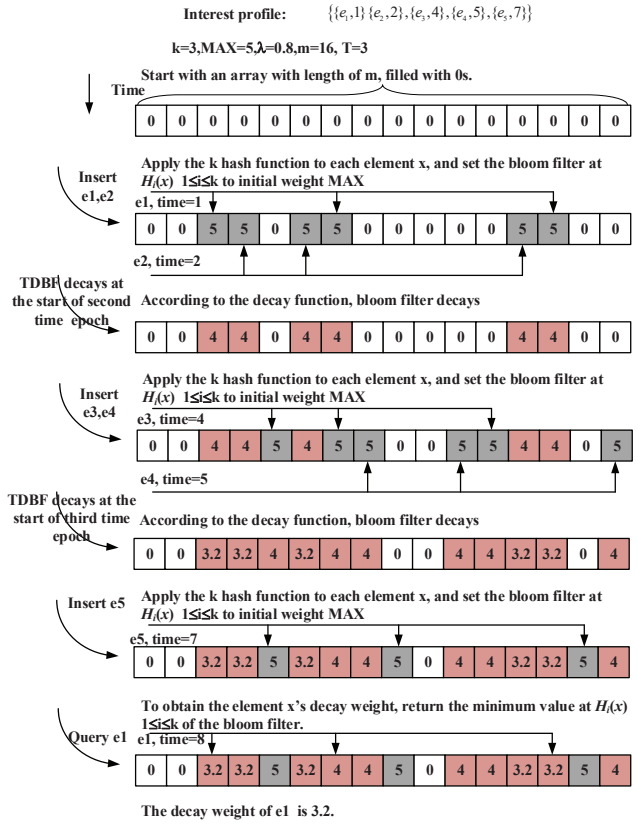


Fig. 2. Operations of TDBF with $k=3$, $MAX=5$, $\lambda=0.8$, $m=16$, and $T=3$.

C. Property analysis

In this subsection, we present some properties of our proposed TDBF, which will be applied in our similarity matching algorithm in the next section.

***Theorem** 1:* The TDBF has a low time complexity $O(1)$ for inserting an interest item to the TDBF, and responding to a query for the decayed interest weight of an item.

Proof: To insert an interest item q to the TDBF, k hash functions are applied to the item q and the corresponding k locations in the bloom filter string $TDBF[h_p(q)]$ for $1 \le p \le k$ are set to MAX. The time complexity is $O(k)=O(1)$. To query the time-decaying weight of the item q, we should check the k counters of $TDBF[h_p(q)]$ for $1 \le p \le k$ and return the minimum value of these counters. The time complexity is also $O(k)=O(1)$. ■

Therefore, our TDBF can support the time-dependent interest representation and query. The time-decaying weight $F(q)$ for the interest item q can be answered in time $O(1)$, and the query cost is independent of the length of the interest profile $|S_i|$.

Let $a_{i,j}$ be the interest item of a mobile user u_i at time $t_{i,j}$, $\hat{F}(a_{i,j})$ be the estimated interest weight of this item by querying against the bloom filter. If $\hat{F}(a_{i,j}) < \lambda^{\lceil \frac{\bar{t}}{T} \rceil - \lceil \frac{t_{i,j}}{T} \rceil} \cdot MAX$, where $\bar{t}$ is the current time, then there exists an *under-estimate error*.

***Theorem** 2:* The TDBF has *zero* under-estimate error.

Proof: When inserting an interest item $a_{i,j}$ to the bloom filter at time $t_{i,j}$, all counters of $TDBF[h_1(a_{i,j})]$, $TDBF[h_2(a_{i,j})]$, $\cdots$, $TDBF[h_k(a_{i,j})]$ are set to MAX. At the start of the next time epoch after $t_{i,j}$, all counters decay with their values updated with $TDBF[h_p(a_{i,j})] = MAX \times \lambda$ for $1 \le p \le k$. However, some of the corresponding counters $TDBF[h_p(a_{i,j})]$ for $1 \le p \le k$ may be set to MAX by other items in the next time epoch after $t_{i,j}$. Therefore, after one time epoch decaying, item $a_{i,j}{}'s$ corresponding counters $TDBF[h_p(a_{i,j})]$ (for $1 \le p \le k$) have their value range $\{MAX \times \lambda, MAX\}$.

Similarly, at the current time, the corresponding counters $TDBF[h_p(a_{i,j})]$ (for $1 \le p \le k$) have their values within the set $\{\lambda^{\lceil \frac{\bar{t}}{T} \rceil - \lceil \frac{t_{i,j}}{T} \rceil} \cdot MAX, \lambda^{\lceil \frac{\bar{t}}{T} \rceil - \lceil \frac{t_{i,j}}{T} \rceil - 1} \cdot MAX, \cdots, MAX\}$. Therefore, all counters of $TDBF[h_p(a_{i,j})] \ge \lambda^{\lceil \frac{\bar{t}}{T} \rceil - \lceil \frac{t_{i,j}}{T} \rceil} \cdot MAX$ for $1 \le p \le k$, and we have $\hat{F}(a_{i,j}) \ge \lambda^{\lceil \frac{\bar{t}}{T} \rceil - \lceil \frac{t_{i,j}}{T} \rceil} \cdot MAX$ according to the operations of the bloom filter. The proof completes. ■

***Theorem** 3:* Although an arbitrary number of the same interest items are allowed to be put in the user interest profile, the estimated decaying weight of an item reflects the latest operation of the item.

Proof: Assume the interest profile of a user is $S = \{(e_1,t_1),(e_2,t_2),\ (e_3,t_3),\cdots,(e_n,t_n)\}$ in which the item e_i appears p times with their time stamps being t_{i1}, t_{i2}, $\cdots$, t_{ip}, respectively. According to the operations of time-dependent bloom filter, all counters at the locations of $TDBF[h_1(e_i)]$, $TDBF[h_2(e_i)]$, $\cdots$, $TDBF[h_k(e_i)]$ are set to MAX at time t_{i1}, t_{i2}, $\cdots$, t_{ip}. These counters can be set to MAX multiple times due to the multiple arrivals of the same interest item, but only the last operation of the item takes effect. The proof completes. ■

Theorem 2 and Theorem 3 are two important and interesting properties of our TDBF. Theorem 2 allows us to estimate the time-dependent similarity iteratively. Theorem 3 guarantees that our similarity matching algorithm based on TDBF does not over-estimate the similarity resulted from multiple arrivals of the same interest item.

V. Time-dependent similarity matching

In this section, we present our probabilistic algorithm to estimate the time-dependent similarity metric defined in Eq(2).

To estimate the similarity (defined in Eq(2)) of a pair of mobile users, we should first estimate the basic similarity value of these users in each time epoch t according to Eq(3), that is, $\zeta_b(u_i, u_j, t) = \frac{|S_i(t) \cap S_j(t)|}{|S_i(t) \cup S_j(t)|}$, and then sum up the weighted basic similarity values of different time epochs using Eq(2).

To estimate the basic similarity value, furthermore, the cardinality of both the set intersection $|S_i(t) \cap S_j(t)|$ and the set union $|S_i(t) \cup S_j(t)|$ should be first calculated. Different from the traditional method which calculates the common interest though the computation-intensive membership testing [19], [24], we propose a probabilistic algorithm to estimate the cardinality of both the set intersection and the set union, and further the time-dependent similarity.

From Algorithm 1, TDBF represents the interest items invoked by utilizing k hash functions to set the corresponding counters to MAX. Even though some interest items may share the counters, the number of counters set to MAX in the bloom filter increases as the number of interest items encoded in the bloom filter becomes larger. Even though the time-decaying function is applied to the TDBF, there still exists a strong relationship between the number of items and the counter's values. In Theorem 4, we will utilize this relationship to estimate the number of items probabilistically.

If we can estimate $|S_i(t)|$, $|S_j(t)|$, and $|S_i(t) \cup S_j(t)|$ from $TDBF_i$ and $TDBF_j$, the $|S_i(t) \cap S_j(t)|$ can be further calculated through $|S_i(t) \cap S_j(t)| = |S_i(t)| + |S_j(t)| - |S_i(t) \cup S_j(t)|$. In this section, we first present our scheme for estimating the number of interest items in a given time epoch by a user u_i, e.g.,$|S_i(t)|$ and the number of items of interest to either user u_i or u_j, e.g., $|S_i(t) \cup S_j(t)|$, and then present our proposed similarity matching algorithm.

A. Estimation of $|S_i(t)|$

Estimating the number of items in a given time epoch directly is difficult because the items arriving after this time epoch may reset the corresponding counters to MAX. Instead, in Theorem 4, we propose a probabilistic algorithm to estimate the number of items arriving in the last p time epochs by utilizing the counter information of the bloom filter, based on which we will present our solution to estimating the number of items in a given time epoch.

Theorem 4: Suppose a $TDBF(m, k, T, \lambda)$ represent an interest profile of a mobile user. The number of counters not smaller than $\lambda^{p-1}MAX$ in the $TDBF$ is s. Then the number of items of interest to the mobile user which are inserted into the filter in the last p time epochs with the interest weight not smaller than $\lambda^{p-1}MAX$ is:

$$n = \log\left(1 - \frac{s}{m}\right) / \left(k \times \log\left(1 - \frac{1}{m}\right)\right) \tag{6}$$

Proof: We assume until the current time, totally M time epochs have passed. The number of items in the last time epoch is n_M, in the previous one time epoch is n_{M-1}, and so on. There are $n = n_M + n_{M-1} + \cdots + n_{M-p+1}$ interest items in the last p time epochs. After all items in an interest profile are hashed into TDBF and the time decay function is applied to TDBF, the probability that a specific counter is smaller than $\lambda^{p-1} \cdot MAX$ can be expressed as:

$$p(n) = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}. \tag{7}$$

The probability that a specific counter is not smaller than $\lambda^{p-1} \cdot MAX$ can be expressed as $1 - p(n)$. Therefore, the total expected number of counters that are not smaller than $\lambda^{p-1} \cdot MAX$, $\hat{S}(n)$, can be expressed as

$$\hat{S}(n) = (1-p) \times m = \left(1 - (1 - 1/m)^{kn}\right) \times m \tag{8}$$

We thus have

$$n = \log\left(1 - \hat{S}(n)/m\right) / \left(k \log\left((1 - 1/m)\right)\right) \tag{9}$$

Let $\hat{S}(n) = s$, the proof completes. ■

Furthermore, as proved in Theorem 2, our TDBF has a good property of zero under-estimate error. Therefore, to estimate the number of items in the pth time epoch from the last, we propose a novel ***inverse estimation principle***. We can first count the number of items in the last p time epoches based on Theorem 4, denoted as D_p, and then count the number of items in the last $p-1$ time epoches, denoted as D_{p-1}. The number of items in the pth time epoch from the last can be calculated as $D_p - D_{p-1}$. Therefore, the number of items in a given time epoch can be estimated.

Following the example in Fig. 2 (in Fig. 2, time 8 is in the third time epoch, the second time epoch is from time 4 to 6), when the current time is 8, we can estimate based on this proposed principle the number of items in the previous one time epoch (from time 4 to 6) in TDBF. This is shown as Steps 1 to 3 in Fig. 3.

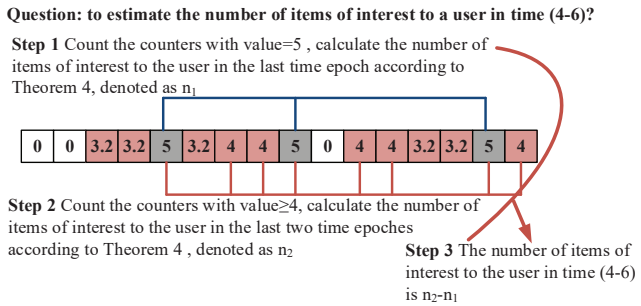


Fig. 3. Example of inverse estimation principle.

B. Estimation of $|S_i(t) \cup S_j(t)|$

Theorem 5 provides a way to estimate the number of items of interest to either user u_i or u_j in the last p time epochs.

Theorem 5: Suppose $TDBF_i(m, k, T, \lambda)$ and $TDBF_j(m, k, T, \lambda)$ represent the interest profiles of users u_i and u_j. The total number of counter locations at either $TDBF_i$ or $TDBF_j$ with the counter value not smaller than $\lambda^{p-1}MAX$ is s_{ij}. Then in the last p time epochs the number of items of interest to either user u_i or u_j is:

$$n_{ij} = \log\left(1 - \frac{s_{ij}}{m}\right) / \left(k \times \log\left(1 - \frac{1}{m}\right)\right) \tag{10}$$

Proof: The standard bloom filter has an important union property, that is $BF(S_i \cup S_j) = BF(S_i) \cup BF(S_j)$. To prove the theorem, we can create two standard bloom filters, BF_i for $TDBF_i$ and BF_j for $TDBF_j$, respectively, following the rule expressed as

$$BF_k[q] = \begin{cases} 1 & TDBF_k[q] \geq \lambda^{p-1} MAX \\ 0 & otherwise \end{cases} \quad (11)$$

where $k = i, j$ and $0 \leq q \leq m-1$. By using the Theorem 4 and the union property of the standard bloom filter, we can easily prove this theorem. ■

Based on Theorem 5, the *inverse estimation principle* in Section V-A can be also applied to estimate the number of items of interest to either user u_i or u_j at a given time epoch.

C. R-Recent time-dependent similarity matching algorithm

To calculate the similarity between mobile users, too old an interest profile is not useful. Therefore, we propose our R-Recent time-dependent similarity matching algorithm, in which only the items in the last R time epochs are considered in the similarity calculation. As shown in Algorithm 2, to calculate R-Recent time-dependent Similarity between mobile users, the algorithm should run R iterations.

On lines 4-6, the number of counters in $TDBF_i$ and $TDBF_j$ whose values not smaller than $\lambda^{p-1} MAX$ can be counted and denoted as $C_1(p)$ and $C_2(p)$, respectively. The total number of counters at either $TDBF_i$ or $TDBF_j$ whose values not smaller than $\lambda^{p-1} MAX$ can be denoted as $C_{ij}(p)$.

On lines 7-8, with Theorem 4, the number of items of interest to user u_i and user u_j in the last p time epochs are $n_i(p) = \log\left(1 - \frac{C_i(p)}{m}\right) / \left(k \log\left(1 - \frac{1}{m}\right)\right)$ and $n_j(p) = \log\left(1 - \frac{C_j(p)}{m}\right) / \left(k \log\left(1 - \frac{1}{m}\right)\right)$, respectively. Moreover, on line 9, with Theorem 5, the number of items of interest to either user u_i or u_j in the last p time epochs is $n_{ij}(p) = \log\left(1 - \frac{C_{ij}(p)}{m}\right) / \left(k \log\left(1 - \frac{1}{m}\right)\right)$.

The similarity is updated in each iterative step, as shown on the line 10. Applying the ***inverse estimation principle***, the number of items of interest to the user u_i and the user u_j in the pth time epoch from the last are $n_i(p) - T_i$ and $n_j(p) - T_j$, respectively, while the number of items of interest to either user u_i or u_j in this time epoch is $n_{ij}(p) - T_{ij}$.

With the design based on simple counting operations on the bloom filters, our proposed probabilistic similarity matching algorithm has a low computation cost. We will evaluate its contribution to energy saving through experiments on cell phones in Section VI.

VI. Performance Evaluations

In this section, we first investigate the accuracy of our R-Recent Time-dependent Similarity Matching algorithm through simulations, and then perform experiments on real-world phones to evaluate the performance of InterestMatch.

A. Accuracy of the similarity matching algorithm

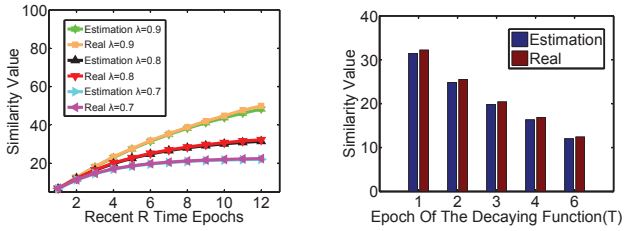
In the simulations, we randomly generate two profiles corresponding to two mobile users, each having 1200 interest items covering 12 months. Each user encodes his profile data into a TDBF according to the set operation and decaying operation in Algorithm 1. After exchanging the TDBFs between the two mobile users, each mobile user estimates the similarity of his profile with the other user following Algorithm 2, denoted as Estimation. For performance comparison, we also directly compute the R-Recent Time-dependent similarity using Eq(2), denoted as Real. In the simulation, the TDBF's parameters are set as follows: length $m = 6000$, the number of hash functions $k = 3$, the time epoch $T = 1$ month, and the initial interest weight $MAX = 128$. We evaluate the performance of the proposed algorithm by utilizing both exponential decay function and linear decay function.

According to Algorithm 2, under the exponential decay function, the parameters R, T and the delaying factor λ directly impact the similarity value. Fig. 4(a) shows how R and λ impact the similarity value by fixing $T = 1$. Obviously, the similarity value increases when R increases, while the increasing speeds under different delaying factors (λ) are different. When the delaying factor λ is a small value, the item's interest weight decays fast with time, which results in smaller similarity value. Fig. 4(b) shows how the time epoch (T) impacts the similarity value using the total 12 month profile data by fixing $\lambda = 0.8$. We observe that when T becomes larger, the similarity value

Algorithm 2 R-Recent Time-dependent Similarity Matching

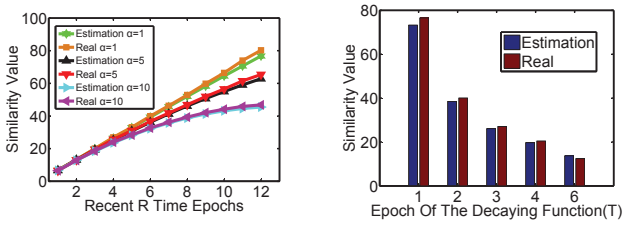
Input: $TDBF_i$ and $TDBF_j$ from two users u_i and u_j
Output: Similarity degree of these two users

1: Initialize $Sim = 0$.
2: Initialize $T_i = 0$, $T_j = 0$, $T_{ij} = 0$.
3: **for** $p = 1; p \leq R; p++$ **do**
4: $C_i(p) = count\left(l \middle| TDBF_i[l] \geq \lambda^{p-1} MAX, 0 \leq l \leq m-1\right)$.
5: $C_j(p) = count\left(l \middle| TDBF_i[l] \geq \lambda^{p-1} MAX, 0 \leq l \leq m-1\right)$.
6: $C_{ij}(p) = count\left(l \middle| \begin{array}{l} TDBF_i[l] \geq \lambda^{p-1} MAX \\ \text{or} \quad TDBF_j[l] \geq \lambda^{p-1} MAX, \\ 0 \leq l \leq m-1 \end{array}\right)$
7: $n_i(p) = \log\left(1 - \frac{C_i(p)}{m}\right) / \left(k \log\left(1 - \frac{1}{m}\right)\right)$
8: $n_j(p) = \log\left(1 - \frac{C_j(p)}{m}\right) / \left(k \log\left(1 - \frac{1}{m}\right)\right)$
9: $n_{ij}(p) = \log\left(1 - \frac{C_{ij}(p)}{m}\right) / \left(k \log\left(1 - \frac{1}{m}\right)\right)$
10: $Sim = Sim + \frac{(n_i(p) - T_i) + (n_j(p) - T_j) - (n_{ij}(p) - T_{ij})}{(n_{ij}(p) - T_{ij})} \lambda^{p-1} MAX$.
11: $T_i = n_i(p)$, $T_j = n_j(p)$, $T_{ij} = n_{ij}(p)$.
12: **end for**
13: Return Sim



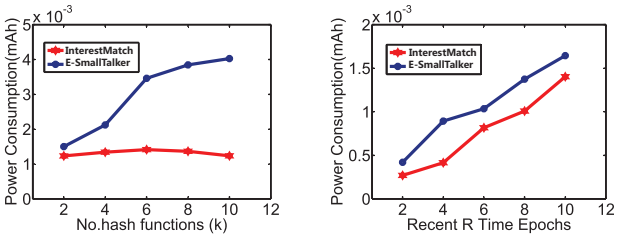
(a) Impact of R and λ (b) Impact of T

Fig. 4. Exponential decay function



(a) Impact of R and α (b) Impact of T

Fig. 5. Linear decay function



(a) Impact of hash number (b) Impact of R

Fig. 7. Power consumption

decreases as the weights of more interest items decay in each epoch. In all scenarios, our TDBF-based similarity matching algorithm can achieve a very high accuracy.

To implement the linear decay function, we change the decay operation in Algorithm 1 to $TDBF[p] = TDBF[p] - \alpha$ (for all $0 \leq p \leq m-1$). To investigate how α and R impact the performance, we vary α and R by fixing $T = 1$. In Fig. 5(a), the similarity value decreases with the increase of α. We also investigate how T impacts the similarity value under the linear decay function by fixing $\alpha = 1$, as shown in Fig. 5(b). Similar to Fig. 4(b), the similarity value decreases with the increase of T. Both Fig. 5(a) and Fig. 5(b) demonstrate that our TDBF-based similarity matching algorithm can achieve a very high accuracy in all scenarios under linear decay function.

Although we utilize an exponential decay function to illustrate our TDBF design, our similarity matching algorithm is a general time-dependent similarity matching algorithm which does not depend on the decay function adopted.

B. Evaluation of InterestMatch

We implement the proposed system InterestMatch using Android Studio with WiFi protocol without modifying the protocol stack on mobile phones, which is supported on a wide variety of mobile phones.

After a user installs our InterestMatch software, when the user wants to find nearby people with common interests to talk, the user will broadcast a request message. Upon receiving the request, a user may decide whether to send back its own bloom filter based on his own decision. After the requester receives the reply, it will estimate the similarity value according to Algorithm 2.

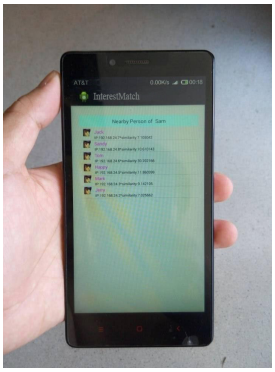


Fig. 6. System running GUI.

We set up a small network testbed with 7 mobile users each holding a smart phone, and the system running GUI at mobile user "Sam" can be shown in Fig. 6. Obviously, because Tom's profile has the highest similarity with that of Sam, Sam can select Tom to begin a short interaction. Because InterestMatch is designed based on TDBF to effectively track the time-dependent interest weight, our InterestMatch can further recommend the topic (of the most interest by both users recently) to begin with by exploiting TDBF's query operations.

To evaluate the energy consumption and demonstrate the effectiveness of our InterestMatch, we implement an adapted E-SmallTalker [24] for performance comparison. As existing profile matching algorithms usually try to find the Set Intersection (PSI) or Private Set Intersection Cardinality (PSI-CA) without considering the decaying weight of user interest, for fair comparison, 1) we implement E-SmallTalker by using multiple bloom filters with each encoding user interest items in one time epoch, then calculate the time-dependent similarity by testing each time-epoch bloom filter, 2) as E-SmallTalker cannot calculate the cardinality of the set union (which is required in basic Jaccard metric and thus our proposed metric) through the membership query, we simplify our Algorithm 2 to calculate only the time-dependent set intersection. To evaluate the power consumption, we first fully charge the Smartphone. As the key part of PMSN application is the profile matching module, we run the module 1000000 rounds after receiving the bloom filter from other users. After that, we read the battery to calculate the average power consumption for one round.

1) Power comparison

The average power consumption of one round is shown in Fig7. The power consumption in our algorithm is not sensitive to the hash function number. Therefore, the power consumption under our algorithm is parallel to x-axis. E-SmallTalker calculates the similarity value through the membership testing based on hash calculations, thus the power consumption increases with the increase of the number of hash functions, as shown in Fig. 7(a). Fig. 7(b) shows the power consumption under different R. As expected, the consumption increases with the increase of R. Since our profile matching approach estimates the similarity through the counting operations, while the matching approach in E-SmallTalker depends on the hash function calculations, the computation complexity thus the power consumption under our approach is much lower than that in E-SmallTalker in all the scenarios

2) Space consumption

To encode the time-dependent user profile following E-SmallTalker, multiple standard bloom filters are needed with each corresponding to one time epoch. TDBF only utilizes a single filter vector which consists of cells instead of bits to store the time related information of the profile, thus the space

of TDBF is determined by the size of the cell. The size of the cell is corresponding to the number of weight values of the time-dependent profile. For a cell with 8 bit, the cell can represent up to $2^8 = 128$ different values. In practice, we can map the values stored in cells to the weight values needed in the system, and set the size of cell according to the number of weight values.

Taking our experiment as an example, to encode user profile with 12 time epochs (thus 12 different weight values), 4 bit cell is enough as $2^4 > 12$. Therefore, the total space needed for TDBF is $4m$, where m is the length of the filter. However, to encode the time-dependent profile following E-SmallTalker, 12 time epochs need 12 standard bloom filters, thus the space is $12m$. Obviously, the space occupation in our method is only 33% of that in E-SmallTalker.

Therefore, compared to E-SmallTalker, our TDBF can facilitate the building of an effective PMSN at lower power consumption and communication cost.

VII. Conclusions

This paper proposes a novel time-dependent profile matching algorithm based on bloom filter, based on which the mobile users close by can build a PMSN based on their common interests. By considering a mobile user's time-dependent interest, we propose a novel time-dependent similarity metric. Furthermore, to calculate the metric in a low cost and privacy-protection way, we propose a novel time-dependent bloom filter to encode the time-dependent interest and a novel probabilistic algorithm to estimate the time-dependent similarity metric based on the bloom filter. To evaluate the performance of the proposed probabilistic similarity matching algorithm, we have done extensive simulations, and the results demonstrate that our proposed probabilistic similarity matching algorithm has a high accuracy. Based on the proposed profile matching approach, we implement InterestMatch and evaluate its performance on real-world phones. Our experimental results demonstrate that our approach is promising for facilitating social interactions and building mobile social networks in a physical proximity due to its low power consumption.

Acknowledgment

The work is supported by the National Natural Science Foundation of China under Grant Nos.61572184, 61472131, the Prospective Research Project on Future Networks (Jiangsu Future Networks Innovation Institute) under Grant No.BY2013095-4-06, the National High Technology Research and Development Program of China (863 Program) under Grant No.2015AA010201 and 2015AA016101, the National Basic Research Program (973 Program) under Grant No.2012CB315805, Beijing Natural Science Foundation under Grant No.4162057, U.S. National Science Foundation under Grant Nos. ECCS-1231800 and CNS 1247924.

References

[1] M Mitzenmacher. A Broder. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[2] Jesús Carretero, Florin Isaila, A-M Kermarrec, Francois Taïani, and Juan M Tirado. Geology: Modular georecommendation in gossip-based social networks. In *IEEE ICDCS 2012*.

[3] Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. The bloomier filter: an efficient data structure for static support lookup tables. In *ACM SODA*, 2004.

[4] Kai Cheng, Limin Xiang, Mizuho Iwaihara, Haiyan Xu, and Mukesh M. Mohania. Time-decaying bloom filters for data streams with skewed distributions. In *IEEE RIDE*, 2005.

[5] Edith Cohen and Martin J Strauss. Maintaining time-decaying stream aggregates. *Journal of Algorithms*, 59(1):19–36, 2006.

[6] Saar Cohen and Yossi Matias. Spectral bloom filters. In *ACM SIGMOD 2003*.

[7] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.

[8] Fan Deng and Davood Rafiei. Approximately detecting duplicates for streaming data using stable bloom filters. In *ACM SIGMOD 2006*.

[9] Benoit Donnet, Bruno Baynat, and Timur Friedman. Retouched bloom filters: allowing networked applications to trade off selected false positives against false negatives. In *ACM CoNEXT 2006*.

[10] Li Fan, Pei Cao, J Almeida, and A.Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *Networking, IEEE/ACM Transactions on*, 8(3):281–293, 2000.

[11] Deke Guo, Jie Wu, Honghui Chen, Ye Yuan, and Xueshan Luo. The dynamic bloom filters. *IEEE Transactions on Knowledge and Data Engineering*, 22(1):120–133, 2010.

[12] Adam Kirsch and Michael Mitzenmacher. Distance-sensitive bloom filters. In *ALENEX*, volume 6, pages 41–50. SIAM, 2006.

[13] Abhishek Kumar, Jun Jim Xu, Li Li, and Jia Wang. Space-code bloom filter for efficient traffic flow measurement. In *ACM SIGCOMM 2003*.

[14] Ming Li, Ning Cao, Shucheng Yu, and Wenjing Lou. Findu: Privacy-preserving personal profile matching in mobile social networks. In *IEEE INFOCOM*, 2011.

[15] Muyuan Li, Zhaoyu Gao, Suguo Du, Haojin Zhu, Mianxiong Dong, and Kaoru Ota. Primatch: Fairness-aware secure friend discovery protocol in mobile social network. In *IEEE GLOBECOM*, 2012.

[16] Rongxing Lu, Xiaodong Lin, Xiaohui Liang, and Xuemin Shen. A secure handshake scheme with symptoms-matching for mhealthcare social network. *Mobile Networks and Applications*, 16(6):683–694, 2011.

[17] Yoshihide Matsumoto, Hiroaki Hazeyama, and Youki Kadobayashi. Adaptive bloom filter: A space-efficient counting algorithm for unpredictable network traffic. *IEICE transactions on information and systems*, 91(5):1292–1299, 2008.

[18] Michael Mitzenmacher. Compressed bloom filters. *IEEE/ACM Transactions on Networking (TON)*, 10(5):604–612, 2002.

[19] Marcin Nagy, Emiliano De Cristofaro, Alexandra Dmitrienko, N Asokan, and Ahmad-Reza Sadeghi. Do i know you?: efficient and privacy-preserving common friend-finder protocols and applications. In *ACM ACSAC*, 2013.

[20] Ben Niu, Xiaoyan Zhu, Tanran Zhang, Haotian Chi, and Hui Li. P-match: Priority-aware friend discovery for proximity-based mobile social networks. In *IEEE MASS*, 2013.

[21] Christian Esteve Rothenberg, Carlos AB Macapuna, Fabio L Verdi, and Mauricio F Magalhaes. The deletable bloom filter: a new member of the bloom family. *arXiv preprint arXiv:1005.0352*, 2010.

[22] Kulesh Shanmugasundaram, Hervé Brönnimann, and Nasir Memon. Payload attribution via hierarchical bloom filters. In *ACM CCS*, 2004.

[23] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2012.

[24] Zhimin Yang, Boying Zhang, Jiangpeng Dai, Adam C Champion, Dong Xuan, and Du Li. E-smalltalker: A distributed mobile system for social networking in physical proximity. In *IEEE ICDCS 2010*.

[25] Qingsong Ye, Huaxiong Wang, and Josef Pieprzyk. Distributed private matching and set operations. In *Information Security Practice and Experience*, pages 347–360. Springer, 2008.

[26] Rui Zhang, Rui Zhang, Jinyuan Sun, and Uanhua Yan. Fine-grained private matching for proximity-based mobile social networking. In *IEEE INFOCOM*, 2012.