E-SmallTalker: A Distributed Mobile System for Social Networking in Physical Proximity

Zhimin Yang*, Boying Zhang*, Jiangpeng Dai*, Adam C. Champion*, Dong Xuan* and Du Li†

*Dept. of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210 USA
{yangz,zhangboy,jpdai,champion,xuan}@cse.ohio-state.edu

†Nokia Research Center
955 Page Mill Road
Palo Alto, CA 94304 USA
lidu008@gmail.com

Abstract—**Small talk is an important social lubricant that helps people, especially strangers, initiate conversations and make friends with each other in physical proximity. However, due to difficulties in quickly identifying significant topics of common interest, real-world small talk tends to be superficial. The mass popularity of mobile phones can help improve the effectiveness of small talk. In this paper, we present *E-SmallTalker*, a distributed mobile communications system that facilitates social networking in physical proximity. It automatically discovers and suggests topics such as common interests for more significant conversations. We build on Bluetooth Service Discovery Protocol (SDP) to exchange potential topics by customizing service attributes to publish non-service-related information without establishing a connection. We propose a novel iterative Bloom filter (IBF) protocol that encodes topics to fit in SDP attributes and achieves a low false positive rate. We have implemented the system in Java ME for ease of deployment. Our experiments on real-world phones show that it is efficient enough at the system level to facilitate social interactions among strangers in physical proximity. To the best of our knowledge, E-SmallTalker is the first distributed mobile system to achieve the same purpose.**

I. Introduction

A. Motivation

Face-to-face interaction plays an irreplaceable role in our daily lives, especially for social networking purposes. Compared to other forms of social interaction that are separated by time and space boundaries, face-to-face interaction in physical proximity facilitates non-verbal communication. In a face-to-face meeting, for example, people can easily make eye contact and discern others' moods, personalities, and surroundings. These non-verbal cues provide immediate and valuable feedback that helps people adjust their topics of conversation, body language, and communication manners accordingly.

Apparently, not all people are equally skillful in harnessing what physical proximity can offer to its fullest extent. A well-known barrier is the so-called *social gap*. When people interact with strangers or unfamiliar parties, they tend to feel self-conscious and reluctant to communicate. *Small talk* is a widely-used everyday technique for shortening the social gap by initiating conversations about readily observable topics such as the weather. However, the effectiveness of small talk is limited if it only covers superficial weather-like topics. Indeed, without assistance, it is generally difficult for ordinary people to identify more significant common topics with strangers in face-to-face social settings.

The mass popularity of mobile phones could potentially help improve the practice of small talk. Today, the number of mobile phone subscriptions has reached nearly five billion worldwide. These phones always go with their owners and are exposed to much information that could be leveraged for more meaningful social interactions, such as mutual friends, common interests, the same schools attended or places visited, and even the fact that two users have already met. We could leverage these mobile phones to enhance the effectiveness of small talk for social networking in physical proximity among strangers.

B. The Key Challenge

The key challenge for such a mobile social networking system is that it must reach a critical mass of users to be useful. That is, the system would provide little value to a user unless a large percentage of other people with whom he would like to interact were also using it.

Clearly, the system should be able to automatically suggest common topics between users who intend to initiate small talk. One straightforward approach is to use a server-based infrastructure, in which a central server stores all users' information and provides common topics based on matching results. A user's client application on his mobile phone needs to report his geo-location to or send IDs of nearby phones to the server via data services (e.g., Internet or SMS) so that the server can discover commonalities among users. Then the matching results are retrieved by or pushed to the client application.

The centralized approach is problematic for two reasons: (1) Not all mobile phones have data services everywhere. For example, mobile data services may be unavailable or too expensive in many developing regions. (2) Not all users are willing to report their sensitive personal information such as geo-location to a central server. Even though one may be willing to report his own information, he may not have others'

This work was supported in part by the US National Science Foundation (NSF) under grants No. CNS-0916584, CAREER Award CCF-0546668, and the Army Research Office (ARO) under grant No. AMSRD-ACC-R 50521-CI. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

permission to report their information, i.e., their phone IDs. Moreover, a central server can be a performance bottleneck and a single point of failure and has the risk of being compromised. Consequently, a centralized system for small talk would be less likely to reach a critical mass of users than systems without the above roadblocks.

C. Our Contributions

This paper presents *E-SmallTalker*, a novel distributed mobile communication system that aims to facilitate more effective social networking among strangers in physical proximity. Our system makes no assumptions about data services such as Internet access. It exchanges user information between two phones and performs matching locally. It uses Bluetooth for communication and is implemented using Java ME. Hence it can be deployed on most commercial off-the-shelf (COTS) mobile phones that are shipped with Bluetooth and Java. These features make it in a better position to be used by more users.

Beyond addressing the key challenge of reaching a critical mass of users, this work makes novel intellectual contributions mainly in the following two areas:

– We build on the Bluetooth Service Discovery Protocol (SDP) to search for nearby E-SmallTalker users. We extend the SDP service attribute values for a new purpose, i.e., to publish encoded user data for discovering potential small talk topics. Our approach does not require user interference to establish a Bluetooth connection.
– We propose a novel iterative Bloom filter (IBF) protocol to encode user information. Data is encoded in a bit string to address the size limit of Bluetooth SDP attributes. The initial Bloom filters are refined in a few rounds until the desired low false positive rate is achieved.

In the following, we explain the rationales behind our approach and contributions:

First, Bluetooth is the most suitable communication technology for our purposes. There are mainly four types of communication technologies available on mobile phones: cellular networks, IrDA, Wi-Fi, and Bluetooth. Communicating data via cellular networks is costly and often unreliable in typical social settings, e.g., inside a building. Infrared Data Association (IrDA) is limited to line-of-sight communication within a very short distance (e.g., 1 meter), which may be considered intrusive among strangers. Wi-Fi is only available on relatively high-end mobile phones. In contrast, Bluetooth is available on nearly all mobile phones and its communication range is 10 meters on class II devices. Hence we choose Bluetooth as our communication technology.

However, in order to develop a Bluetooth application on mobile phones, we need to overcome several obstacles. For security reasons, a mobile phone will ask for user permission to initiate or accept a Bluetooth connection as well as a passcode for pairing. Hence, an application that relies on Bluetooth connections to transmit data requires explicit user interactions. This requirement not only requires users to "babysit" the system but is also too intrusive for strangers. Therefore, we need to find a way for two phones to exchange information without establishing a Bluetooth connection.

We achieve this by using Bluetooth Service Discovery Protocol (SDP) to publish/exchange information. In SDP, each service is represented by a service record that is identified by a 128-bit Universally Unique Identifier. All information about a service that an SDP server maintains (on a phone) is contained within a single service record, which consists of a list of service attributes. Each service attribute describes a single characteristic of a service (e.g., its name, type, parameters, protocols used) and consists of an attribute ID and the corresponding attribute value. The attribute value is a variable length field, which our system utilizes to publish encoded user information. To our knowledge, this is the first work that utilizes Bluetooth SDP by customizing service attributes to exchange non-service-related information.

However, SDP can only publish limited information, the size of which varies depending on brands and models of mobile phones. For example, in our experiments, one phone can publish up to 10 attributes, each of which has a maximum of 128 bytes of data. We use Bloom filters to encode and "compress" user information in order to fit it into SDP's attribute values. To further reduce the size of exchanged information, we propose a novel Bloom filter technique that iteratively refines Bloom filters in several rounds to achieve a desired low false positive rate given SDP's constraint. The Bloom filters are published via SDP to discover common topics. A device determines common topics by testing if its topics are in another device's Bloom filter. As a result, the system incurs limited transmission and computation. As a one-way hashing technique, Bloom filters also provide a reasonable level of privacy against eavesdroppers. It is generally difficult, if not impossible, to reconstruct the information in a Bloom filter without performing an exhaustive search of the input space.

We implemented the proposed system using Java ME, which is supported on a wide range of mobile phones including smartphones as well as low-end phones. In addition, we build our system on the Bluetooth protocol but we do not modify the protocol stack. As a result, the system easily runs on most Bluetooth-enabled COTS mobile phones. We performed experiments on real-world mobile phones. The results show that E-SmallTalker achieves our design goals.

D. A Typical Usage Scenario

Suppose that two strangers, Alice and Bob, encounter each other at an airport. They are both interested in several movies. As they are strangers, they only make small talk about the weather, which is clearly superficial. If they run E-SmallTalker on their mobile phones, the system encodes their movie interests into Bloom filters, which are published as service attribute values in SDP's service record. The system automatically exchanges Bloom filters, performs information matching locally on the phones, and then informs them of their common interest in movies. They can then easily initiate a meaningful conversation about movies. The entire procedure requires neither a central server nor Internet access and it is transparent to both users until each of their phones finds their common interests.

To the best of our knowledge, E-SmallTalker is the first distributed mobile system for social networking between strangers, with two supporting techniques: a new way of utilizing Bluetooth SDP and a novel iterative Bloom filter protocol.

The remainder of this paper is organized as follows. Section II discusses related work on social networking applications. Section III presents our system design. Section IV presents our implementation and performance evaluation. Section V discusses further issues and Section VI concludes.

II. Related Work

In this section, we review related work focusing on social networking applications (SNAs) on mobile phones. A considerable body of work has contributed to this area. Some SNAs are centralized, whereas other SNAs are distributed. We will review background material and related work for Bluetooth SDP and Bloom filters in Section III when discussing system design.

– *Centralized Social Networking Applications:* The SNAs in this category are primarily Internet-based. They store user data including social networks on a (conceptually) central server and allow users to find friends and share data via SNA clients running on mobile phones.

Social Serendipity [1] is a typical example of the centralized SNA. Specifically, It maps Bluetooth MAC addresses to user profiles on other social networking websites. To facilitate face-to-face interactions between nearby strangers, it retrieves their mobile devices' Bluetooth MAC addresses and uses them to retrieve the strangers' profiles on the server for similarity matching. It uses SMS for device-server communication.

A number of other centralized SNAs aim to enhance awareness and interaction between friends when they are in physical proximity. In general, these SNAs obtain a user's current geographical location and notify his nearby friends. This gives friends knowledge of each other's whereabouts, which facilitates opportunistic interactions. Representative applications in this category include PeopleTones [4], Hummingbird [9], Just-for-Us [5], MobiLuck [8], Micro-Blog [10], and Loopt [3].

In general, centralized SNAs have the following limitations: (1) the server may not always be reachable; (2) communications between the server and devices (via SMS, Wi-Fi, or 2G/3G) may be costly, unreliable and even unavailable; and (3) user privacy may be compromised, e.g., by saving location and other personal data on a third-party server. By comparison, our system uses short-range communication technologies such as Bluetooth to provide reliable service operation without suffering from these limitations.

– *Distributed Social Networking Applications:* The SNAs in this category enable mobile devices to directly communicate with each other without requiring a third party. For example, Social Net [6] logs nearby users' Bluetooth addresses to infer users' interaction patterns over time. Nokia Sensor [2] allows users to detect others in the vicinity via Bluetooth; once a connection is established, two devices can exchange information. PeopleNet [11] focuses on multicasting messages or queries to a selected group of devices connected by a mobile ad hoc network based on Bluetooth or Wi-Fi. Nokia Sensor and PeopleNet require establishment of a Bluetooth connection, which requires user intervention and only occurs between trusted parties. These applications focus on providing users service when they are already connected via Bluetooth.

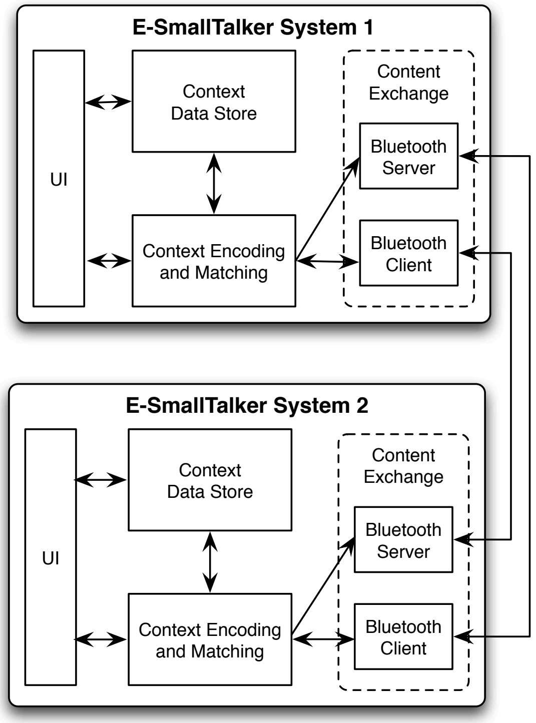

Fig. 1: E-SmallTalker System Architecture.

Point&Connect [12] facilitates device pairing between two users in physical proximity by having one user point his device to another user's device. When multiple devices are nearby, acoustic cues are used by all devices to determine the device towards which the initiating device is pointing. Our work focuses on initiating meaningful small talk by identifying common topics between two strangers. While Point&Connect has a entirely different focus, its results can be leveraged in our work, e.g., when one person intends to initiate small talk with another person in a crowd.

III. System Design

In this section, we present the system design of E-SmallTalker. We first overview the system architecture. Then we discuss two key components in this architecture, namely, Context Encoding and Matching, and Context Exchange.

A. System Architecture

Figure 1 shows the E-SmallTalker system architecture, which includes the following four software components:

– *Context Data Store*: This component stores user data that contributes to small talk and metadata that controls system operation. Small talk between two people is often highly *situated:* topics may cover mutual friends and hobbies, schools both attended, places both visited, and historical facts such as their meeting last year in the same conference. System metadata includes the Bloom filter parameters and user preferences such as what data may and may not be published (in Bloom filters to strangers). For the scope of this paper, we simplify this part and assume that friends' information is imported from the phone contact database and that the user's interests are represented in a limited vocabulary of keywords.

– *Context Encoding and Matching*: This component encodes the context data to be published by the Context Exchange component. The data is encoded using Bloom filters with user-configured parameters. To save computation, the Bloom filters are computed and cached in the Context Data Store when the parameters are changed or the encoded data are updated. The Bloom filters are retrieved by the Context Exchange to find matching elements when a query is received.

– *Context Exchange*: This component includes a Bluetooth server (BTS) and a Bluetooth client (BTC). The BTS creates the service record with Bloom filters as service attribute values and publishes the service record via the Bluetooth SDP server. The BTC first performs an inquiry over the Bluetooth radio to retrieve the MAC address of any device in the physical proximity at the time of inquiry; then it discovers whether or not the device is running E-SmallTalker. If this is the case, it retrieves the Bloom filters that the device publishes.

– *User Interface (UI)*: This component provides interfaces for a user to configure small talk policies and rules. For example, the user can specify which contacts to encode in his Bloom filter; what personal profile data to publish, e.g., name, age, phone number, gender, and interests; Bloom filter parameters to encode his own information; and conditions by which a received Bloom filter will be ignored. The user can also specify how he is notified of matching information: text display, ringtone, vibration, or speech. For the scope of this paper, we also simplify this part and assume that the contact Bloom filter encodes all friends' names plus mobile phone numbers and that the interest Bloom filter encodes all listed personal interests.

The four components work together to help users initiate small talk. The corresponding workflow is as follows: First, our E-SmallTalker system uses the Context Data Store to record user-customized profile information. Then, the Context Encoding and Matching component compresses and encodes information from the Context Data Store into Bloom filters. When two users are in Bluetooth communication range, their Context Exchange components automatically exchange their Bloom filters to find their similarities via Bluetooth SDP. In different social scenarios, users can easily configure corresponding settings via the User Interface component. This paper focuses on the two most critical components: Context Encoding and Matching, and Context Exchange.

B. Context Encoding and Matching

To discover possible matches based on Bluetooth SDP, we first formulate the problem and then present a new multi-round, iterative discovery protocol for context encoding and matching based on Bloom filters. The goal is to minimize system overhead and error rate under the constraint that Bluetooth SDP can only exchange limited information.

1) Problem Formulation: Small talk topics can include any commonalities, such as common friends, shared interests. Here we abstract all of those as *topics*. The problem can be formulated as follows. Consider a dynamic set $U = \{u_1, u_2, ..., u_N\}$ of N potential communication partners in which each user u_i has a set $SetU_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,n_i}\}$ of n_i data items. Each $a_{i,j}$ $(1 \le i \le N, 1 \le j \le n_i)$ is a topic of interest. Each user u_i wants to discover two things: (1) the identical items in $SetU_i$ and $SetU_k$ $(1 \le i, k \le N, i \ne k)$; and (2) the corresponding user u_k if there are identical items between $SetU_i$ and $SetU_k$.

A naïve way is to establish a Bluetooth connection between two devices u_i and u_k and transfer the two data sets together to compute their intersection, $SetU_i \cap SetU_k$. However, this requires user interaction to set up Bluetooth connection and transfer a large message.

Alternatively, we encode the data sets in Bloom filters and use the Bluetooth SDP to transmit the Bloom filters. Then we compute their intersection. This approach is non-intrusive and more efficient.

2) Basic Bloom Filter: The Bloom filter [7] is a time- and space-efficient probabilistic data structure for testing whether an element is a member of a set. A Bloom filter is a vector of m bits, each of which is initially set to '0'. When adding a new element to the Bloom filter, we compute the element over k independent hash functions to generate k hash values as the indices to the vector. The corresponding k entries are set to '1'. To insert a set of n elements, this procedure is repeated n times until all the elements are encoded in the Bloom filter. During the procedure, if a bit is already set, we leave it as '1'. To query an element against a given Bloom filter, the k hashing indices are computed: The element is a member of the set only if all the k corresponding bits are '1' in the vector.

As a probabilistic data structure, Bloom filters are subject to false positives, i.e. they may mistakenly confirm the membership of a given element in lookup. The false positive rate f is defined by the probability that all the corresponding k bits for any given element are '1' in the Bloom filter although it is not really a member of the represented set. Assuming that a hash function selects each position in a Bloom filter with equal probability, then the quantitative measurement of false positive rate is defined by the following formula:

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

There are two challenges that must be addressed: First, the size of a Bloom filter m grows linearly with the size of a data set n if we want to guarantee a given false positive rate f. Second, the size of a Bloom filter is effectively bounded by the maximum size of custom information that a SDP service record can publish, which varies on different mobile phones.

Several other extensions to the basic Bloom filter exist in the literature [13]–[18] that are orthogonal to ours. However, the results of these works cannot be readily applied to mobile phones because Bluetooth SDP can only exchange limited information; thus a preferred false positive rate cannot be guaranteed when the data set is large. A suitable compromise must be sought. We discuss our solution in the next subsection, which addresses this problem by a novel iterative protocol.

3) An Iterative Discovery Protocol: We aim to achieve the desired false positive rate f with a minimum total amount of transmission given the constraints imposed by the implementation of Bluetooth SDP. We devised a multi-round protocol: in each round, a Bloom filter with some false positive rate is published and a subset of common data items is computed; this smaller subset is then encoded in another new Bloom filter with much lower false positive rate; eventually, the commonalities are reported with the desired rate f.

We specify the protocol as follows. For simplicity, it assumes that there are only two parties, A and B. Let the interesting data sets of A and B be $SetU_A$ and $SetU_B$, respectively. It is not difficult to extend the protocol specification to allow for multiple parties.

Step (1). Initially (round 0), both devices encode their own data sets, $SetU_A^0 = SetU_A$ and $SetU_B^0 = SetU_B$, in two static Bloom filters, BF_A^0 and BF_B^0, respectively, and publish them using a special static attribute ID in the SDP service record.

Step (2). In the $(r+1)$-th round, A first retrieves BF_B^r via Bluetooth SDP and then checks the membership of each data item in $SetU_A^r$ against BF_B^r to obtain a matching set $SetU_A^{r+1} \subseteq SetU_A^r$. Next, A encodes $SetU_A^{r+1}$ into a new dynamic Bloom filter BF_A^{r+1}. Finally, A publishes BF_A^{r+1} and the current step number r using a new dynamic attribute ID calculated from B's Bluetooth ID, (e.g. the last 4 bytes of SHA1 hash of B's Bluetooth ID) making it specific to B. Symmetrically B takes the same action.

Step (3). In the following $(r+2)$-th round, A first retrieves BF_B^{r+1} that is specially generated for A. Step (2) is repeated similarly to generate a new matching set $SetU_A^{r+2}$ and a new Bloom filter BF_A^{r+2} specially for B published with the same attribute ID as in step (2). This process is repeated until the new matching set is empty or the same as that of the last round or the desired false positive rate is reached. A dynamic attribute is removed from SDP service record when a predefined lifetime is reached after the end of the above process.

In each round, we replace an old hash function with a new independent hash function generated by the technique described in [13]. In this way, we can iteratively eliminate the case in which two items have the same set of hash values. Thus, for any two honest parties A and B, the resulting set $SetU_A^r$ in round r ($r \geq 1$) between A and B converges to $SetU_A \cap SetU_B$ as $r \to \infty$.

Between two strangers A and B, it is reasonable to assume that the intersection of their data sets is a proper subset of either original set and the intersection is much smaller. As the data set size n decreases, so does the Bloom filter size m when the false positive rate f and number of hash functions k are fixed. In each round, when a new Bloom filter is constructed, we can either dynamically decrease the filter size m according to the current n and resulting f, or decrease f dramatically by keeping the filter size m.

Algorithm 1 The 2-Round Discovery Protocol for User u_0

```
1:  [Round 1:]
2:  Initialize Bloom filter $BF_0$;
3:  for each item $a_{0,j}$ in $SetU_0$
4:      calculate $k$ hash indices of item $a_{0,j}$;
5:      set the corresponding bit of $BF_0$ to 1;
6:  end for
7:  publish $BF_0$ with a static attribute ID;
8:
9:  [Round 2:]
10: acquire the dynamic device(user) set $U$;
11: for each user $i$ in set $U \setminus \{u_0\}$
12:     retrieve the Bloom filter $BF_i$ from $u_i$;
13:     for each item $a_{0,j}$ in $SetU_0$
14:         if $BF_i.contains(a_{0,j})$ then
15:             add $a_{0,j}$ to common set $CS_i$;
16:         end if
17:     end for
18:     if $CS_i.size > 0$ then
19:         initiate a new Bloom filter $BF_0'$;
20:         encode $CS_i$ into $BF_0'$ with different set of $k$ hash indices;
21:         publish $BF_0'$ with a dynamic attribute $ID_i$;
22:     end if
23: end for
24: for each user $i$ with $CS_i.size > 0$
25:         retrieve the new Bloom filter $BF_i'$ from $u_i$;
26:         for each item $a_{0,j}$ in $CS_i$
27:             if $BF_i'.contains(a_{0,j})$ then
28:                 add $a_{0,j}$ to common set $CS_i'$;
29:             end if
30:         report common set $CS_i'$ associated with user $u_i$ to $u_0$;
31: end for
```

In our implementation, the length of a Bloom filter in each round should be less than or equal to 128 bytes, which is the maximum effective size of any attribute value in the Bluetooth SDP service record, and the number of rounds should be less than 10, which is the maximum number of attribute IDs that can be used to publish Bloom filters. We found that the size of the initial data set (e.g., contacts) $SetU_A^0$ for any user A is a few hundred, the protocol converges after a few rounds, and two rounds are normally enough for strangers. Hence it is effectively a 2-round protocol.

Algorithm 1 specifies our 2-round protocol for publishing user data and discovering matches. The same algorithm is executed on every device in question. In the first round, we initiate Bloom filter BF_0, insert all the user data items into BF_0, and then publish the resulting BF_0. In the second round, Bloom filters are retrieved from all nearby devices. If matches are found with the Bloom filter BF_i from any user u_i, we store the matches in the corresponding set CS_i. We build a new Bloom filter BF_0' out of CS_i and then publish BF_0'. Finally, in steps 25–30, we collect the set of common interests CS_i' with target user u_i and report it to the local user.

The above multi-round scheme is essentially an *iterative Bloom filter* (IBF) protocol that can significantly reduce the size requirements of Bloom filters and hence improve system efficiency without sacrificing precision of the false positive rate. We show the performance benefits in the next subsection.

TABLE I: Precomputed Bloom filter size (in bytes) with regard to the number of items (N) and desired false positive rate (F) with the number of hash functions fixed at 7.

Table for Calculating Bloom Filter Size								
N / F	5%	2%	1.50%	1%	0.10%	0.01%	0.001%	0.0001%
5	5	6	6	7	10	15	21	30
10	9	11	12	13	19	29	41	59
25	21	26	28	31	47	71	103	147
50	42	52	56	61	94	141	205	293
100	83	104	110	120	188	281	408	585
150	125	155	165	180	282	421	612	878
200	166	207	220	240	376	561	816	1170
250	208	258	275	300	470	701	1020	1463
300	249	310	330	360	563	841	1224	1755
350	291	362	385	420	657	981	1428	2048
400	332	413	440	480	751	1121	1632	2340
450	374	465	495	540	845	1261	1836	2633
500	415	516	550	600	939	1401	2040	2925
550	457	568	605	660	1032	1541	2244	3217
600	498	619	660	720	1126	1681	2448	3510
650	540	671	715	780	1220	1821	2652	3802
700	581	723	770	840	1314	1962	2856	4095
750	623	774	825	900	1408	2102	3060	4387
800	664	826	880	960	1501	2242	3264	4680
850	705	877	935	1020	1595	2382	3468	4972
900	747	929	990	1080	1689	2522	3672	5265
950	788	980	1045	1140	1783	2662	3876	5557
1000	830	1032	1100	1200	1877	2802	4079	5849

4) The Benefits of Iterative Bloom Filters: Now we illustrate the performance benefits of our IBF protocol with the following example. Suppose that a user has 150 contacts, a phone number is expressed as 10 digits, and the average length of names is 10 characters. (The actual length of names may be much longer.) In a naïve approach in which we publish a list of phone numbers and names, we need $150 \cdot (10 + 10) = 3000$ bytes. Table I is for calculating the Bloom filter size with different numbers of contacts and false positive rates. To guarantee a 0.001% false positive rate, we need a Bloom filter with 612 bytes, which saves about 80% space over the naïve approach.

Using our 2-round IBF protocol, we can achieve the same false positive rate with much less bytes. Given the same contact list and the 128 bytes length constraint, we can use 125 bytes for the first Bloom filter in the first round. This allows us to achieve a false positive rate of 5%. We obtain an initial common set with $150 \cdot 5\% = 8$ false positives. In the second round, we need only build a Bloom filter to address this reduced set while keeping the false positive rate at $0.001\%/5\% = 0.02\%$. Regardless of whether there are true common elements, the second Bloom filter's overhead that accounts for the 8 false positives is less than 29 bytes. Therefore the worst-case overhead is about 154 bytes to achieve the same false position rate of 0.001%, which is almost 75% savings compared to the 612 bytes required by the basic Bloom filter. More savings are expected when the number of contacts or number of rounds grows larger. This example speaks up for the merits of our approach over the basic Bloom filter.

C. Context Exchange

A common way of using Bluetooth to exchange information involves several steps: (1) search for nearby devices; (2) discover the services they provide; and (3) pair two devices and establish a connection to use a discovered service. In step (3), a mobile phone will ask for user permission to accept a Bluetooth connection as well as a passcode for pairing. For target users of our system who are strangers, this is too intrusive and irritating.

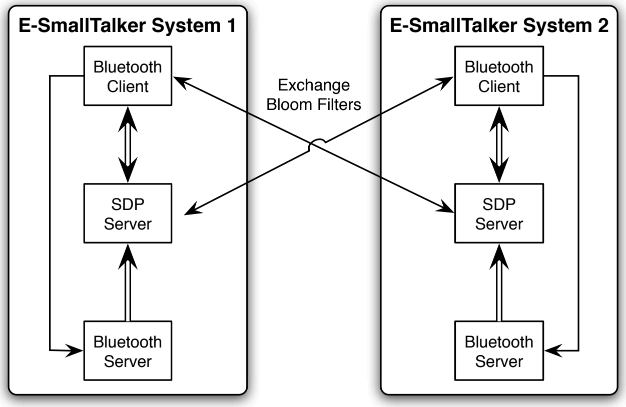

Fig. 2: Information Exchange without Bluetooth Connection.

We need to find a way for two phones to communicate with each other without establishing a Bluetooth connection. We achieve this by using Bluetooth's Service Discovery Protocol (SDP) to exchange encoded user information.

The trick is to exploit Bluetooth service attribute values to publish Bloom filters. SDP provides a means for applications to discover which services are available and to determine the attributes of those services. In SDP, each service is represented by a service record that is identified by a 128-bit Universally Unique Identifier (UUID). All information about a service that an SDP server maintains is contained within a single service record, which consists of a list of service attributes. Each service attribute describes a single characteristic of a service (e.g., its name, type, parameters, protocols used) and consists of an attribute ID and the corresponding attribute value. The attribute value is a variable length field, which is explored by our system to publish custom data. A client may issue an SDP request to retrieve information from a service record maintained by the SDP server on another device.

Our system takes advantage of this structure. We create a service record by starting a *virtual service* with a known UUID and a list of attributes. We use attribute values to publish information encoded in Bloom filters. We call it a "virtual service" because it provides no service in the traditional sense that a client can consume or to which a client can connect. It exists only to publish attributes. By updating the service record to the SDP servers, two Bluetooth devices can exchange information *without* setting up a Bluetooth connection.

Unlike Wi-Fi, there is no broadcast channel or beacon signals in Bluetooth due to the fact that it uses a frequency-hopping spread spectrum radio technology. In order to perform communications, a Bluetooth slave device must follow the master's hop pattern which cannot be generated without knowing the master's address/clock values. Therefore, our system adopts a *pull* model, not a *push* model. A device is not *broadcasting* its service; rather, it is *publishing* its service, waiting for another device to discover it and retrieve the information.

An alternative approach to exchange information without setting up a Bluetooth connection is to use the Bluetooth device name, which consists of a maximum of 248 bytes of text data. We instead use Bluetooth SDP because it can publish more information than a single device name. The size of custom data (the number of attributes times the length of attribute values) in a service record varies on different mobile devices. For example, in our experiment a successful communication between a Sony-Ericsson W810i phone and a Nokia N82 can exchange up to 10 attributes, each of which has a maximum of 128 bytes of data. Two Nokia N82s can exchange more attributes and more bytes of data in each attribute. In addition, SDP provides more flexible and fine-grained control as we are able to change an attribute value individually.

The Context Exchange component in Figure 2 shows an overview of Bluetooth SDP-based communication in our system. The Bluetooth server publishes Bloom filters through the phone's SDP server. The Bluetooth client acquires the other phone's Bloom filters by sending a SDP request to the SDP server on the target phone.

IV. Implementation and Evaluation

We will first overview the E-SmallTalker system implementation and then present the performance evaluation results.

A. System Implementation

We choose Java ME as our prototype development environment because it is supported on most mobile phones on the market. We implement the E-SmallTalker system and test it on several brands of mobile phones, including Sony Ericsson (W810i) and Nokia (5610xm, 6650, N70, N75, N82). It is important for a social networking system like E-SmallTalker to run across a wide variety of phones: it is useful in practice only when a critical mass of users are using it.

We implement the system with the Eclipse SDK and the Sun Java Wireless Toolkit for Connected Limited Device Configuration (CLDC) based on the Mobile Information Device Profile (MIDP) specification. We use the Java APIs for Bluetooth described in the JSR-82 interface for developing service discovery applications. We import some Java code from the XSiena (eXtended Scalable Internet Event Notification Architecture) project for the hash functions used in our Bloom filters. The size of the deployed executable is about 127KB. When a user starts the application for the first time, it asks him to configure his personal profile (e.g., name, telephone number, interests, past experience) and the system settings (e.g. automatic or manual discovery, time period between each discovery). The system imports the contacts from the address book and lets the user choose which ones will be published in the Bloom filter. Then the system performs the following major operations: (1) generating and publishing Bloom filters through Bluetooth SDP according to the user's configurations, (2) retrieving Bloom filters from nearby mobile phones, (3) matching common interests or contacts, and (4) prompting the user when there are matches.

B. Performance Evaluation

1) Experimental Setup: There are two classes of metrics to evaluate E-SmallTalker's performance: *system performance* and *social performance*. System performance includes discovery performance and power consumption. Social performance mainly includes the likelihood that strangers in physical proximity share common interests. In order to evaluate social performance, we need to run a massive field test, which is very costly. As a first step of performance evaluation, we focus on system performance. System performance has a critical impact on whether people accept our system, which is the key challenge of our design. We plan to run a massive field test to measure social performance in our future work.

One key system performance is discovery performance. To evaluate the discovery performance in E-SmallTalker, we focus on two detailed evaluation metrics: (1) *Successful discovery rate.* This is defined as the percentage of successful discoveries among all attempts to search among nearby users. Given an E-SmallTalker user A, assuming that he shares common interests with n nearby users, a *successful discovery* is one that reports all the matches among user A and the n nearby users. (2) *Commonality discovery time.* This is the period from the time of starting a search to the time of finding someone with common interests. This can be considered as an end-to-end delay. Power consumption is key to system performance. Since Bluetooth communication dominates E-SmallTalker's power consumption, we measure power consumption in the two major parts of communication: Bluetooth device discovery and Bloom filter retrieval (i.e., Bluetooth service discovery). We also measure power consumption when E-SmallTalker is always on.

In our experiments, we use 6 mobile phones (a Sony Ericsson W810i, a Nokia 5610xm, a Nokia 6650, and 3 Nokia N82s). Each experiment is repeated 10 times to average the results. We conduct a survey of 35 college students that reveals that they have, on average, 143 contacts in their mobile phones. From this result, we set the number of contacts per phone to $n = 150$ in the experiments. In addition, we use seven hash functions ($k = 7$) for all Bloom filter insertion and lookup operations. Since Bluetooth's technical details influence our experiments, we conduct experiments considering the following three factors: (1) *the Bluetooth search interval*, i.e., the time between two consecutive searches for Bluetooth devices and services in the application, (2) *the number of nearby devices*, and (3) *the distance between two devices*, which is 4 meters by default.

2) Successful Discovery Rate and Discovery Time: We first conduct experiments to measure how the Bluetooth search interval and the number of users (devices) impact discovery performance.

Intuitively, if the Bluetooth search interval is too long, the user may miss potential opportunities for social interactions. On the other hand, if the interval is too short, the chance of encountering collisions increases significantly and causes discovery failures, especially when multiple mobile phones are in physical proximity. This is caused by current limitations of the Bluetooth protocol as will be discussed later. For similar reasons, the more devices in physical proximity, the higher the chance of encountering collisions and discovery failures.

In this group of experiments, we put the phones in close

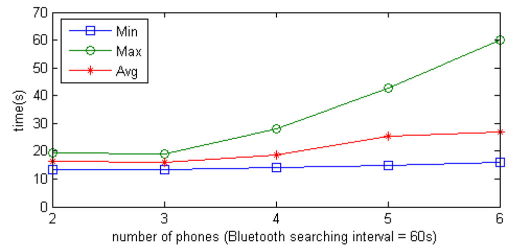

Fig. 3: Discovery Time with 60 s Bluetooth Search Interval.

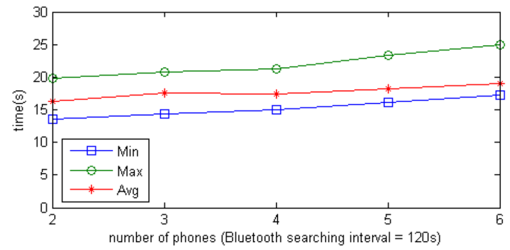

Fig. 4: Discovery Time with 120 s Bluetooth Search Interval.

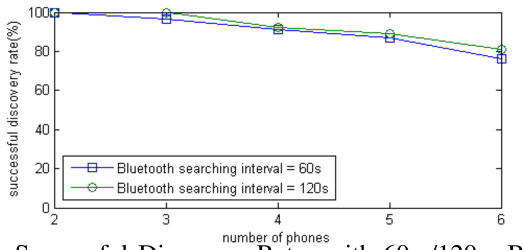

Fig. 5: Successful Discovery Rates with 60 s/120 s Bluetooth Search Intervals.

proximity (about 4 meters apart if possible). The Bluetooth search intervals were set to 60 or 120 seconds and the number of phones was iteratively increased from 2 to 6.

Regarding the number of phones with different search intervals, Figures 3 and 4 show the minimum, average and maximum discovery time and Figure 5 shows the successful discovery rate. From the experimental data, by fixing the number of devices, the Bluetooth search interval does not show any significant impact on discovery time. The average discovery time when the interval is 60 s is only slightly longer than the case of 120 s. This can be explained by the fact that the discovery time is actually dominated by the Bluetooth device discovery time, which is on the order of magnitude of 10 s [21]. On the other hand, it is clear from the data that as the number of devices increases, the discovery performance decreases. Furthermore, both the number of devices and the search intervals show a clear impact on the successful discovery rate. The success rate drops as the interval decreases and the number of devices increases.

In the next group of experiments, we seek to understand how the distance between devices affects the common interest discovery performance. We place two Nokia N82 phones on the floor 1 meter apart, run the experiment 10 times, and measure the average discovery time and successful discovery rate. Then we repeat the experiments by increasing the distance 1 meter at a time until the two phones are 10 meters apart.

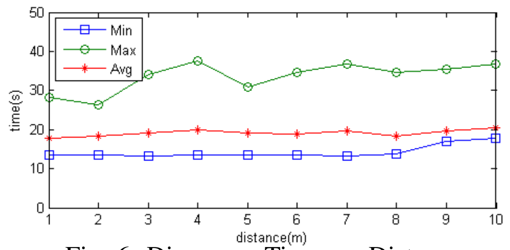

Fig. 6: Discovery Time vs. Distance.

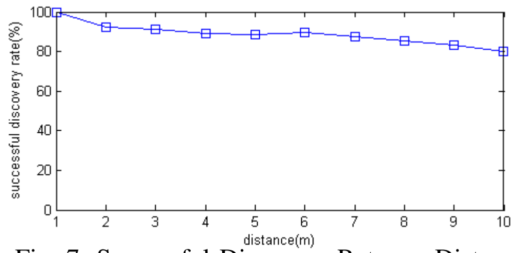

Fig. 7: Successful Discovery Rate vs. Distance.

Figure 6 shows the minimum, average and maximum discovery times versus distance between two phones. Figure 7 presents the success rate versus distance. In summary, the minimum, average and maximum discovery times in our experiments are 13.39 s, 20.04 s, and 58.11 s, respectively, among all the E-SmallTalker data we collected. The overall success rate is 90%. From the data, there is no clear trend of how the distance affects the average discovery time or the success rate, although the maximum discovery time becomes flicky when the distance increases from 2 meters to 5 meters. This result demonstrates that E-SmallTalker's performance is quite stable within 10 meters. However, when two phones are placed more than 10 meters apart, which is the nominal communication range for a standard Bluetooth device, they cannot find each other.

According to the Bluetooth specification [21], "the inquiry substate may have to last for 10.24 seconds unless the inquirer collects enough responses and determines to abort the inquiry substate earlier." In a noisy environment, there is no guarantee of successful inquiry. In such situations, the inquiry time may far exceed the default time of 10.24 seconds. In the inquiry substate, a Bluetooth device will not respond to another device. As a consequence, if two mobile phones start searching for devices at approximately the same time, there is a collision and they cannot obtain each other's service information in a timely manner. With an increased number of devices or a reduced Bluetooth search interval, the probability of collision increases, which leads to a lower successful discovery rate and a longer discovery time.

3) Power Consumption: To study trends regarding power consumption, we place two Nokia N82 phones 4 meters apart, both fully charged. A Symbian application, the Nokia Energy Profiler (called NEP or Juice), is used to read the battery voltages, currents, and energy consumptions once per second.

First, we measure one N82's energy cost for Bluetooth device discovery and service discovery (for the purpose of Bloom

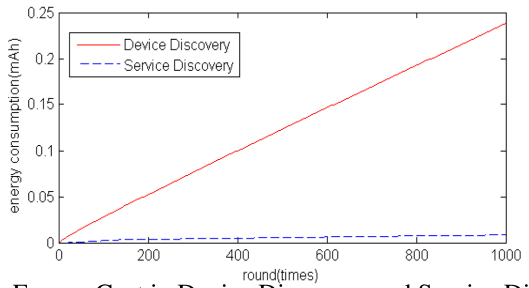

Fig. 8: Energy Cost in Device Discovery and Service Discovery.

filter retrieval), respectively. Figure 8 shows the accumulated energy consumption for 1000 rounds/times of device discovery and service discovery. From the data we can see that device discovery consumes significantly more energy than service discovery. Hence, the service discovery part in our 2-round protocol has little impact on the overall power consumption of the application, which is dominated by the device discovery part. Considering the fact that the device discovery procedure is needed by all Bluetooth applications, even a multi-round iterative Bloom filter protocol contributes little power consumption overhead.

Then we monitor the power consumption in device discovery with nonstop (i.e., continuous) and 60 s Bluetooth search intervals. Figure 9 shows the results of a fully charged Nokia N82 for 8 hours. It shows a clear trend where the energy consumption decreases when the Bluetooth search interval increases.

Finally, we record the run time of E-SmallTalker on a fully charged N82 phone (1050 mAh) until the battery is exhausted. In our experiment, we let the phone run in standby mode while turning on the cellular and Bluetooth radios. When the search interval is set at 60 s, the phone runs slightly longer than 29 hours. Using the same settings on the cellular and Bluetooth radios, we redo the experiment without running E-SmallTalker. In this case, the phone runs 32 hours until the battery is depleted. This result is encouraging since a user will likely only need to run E-SmallTalker when necessary, e.g., in specific social settings.

V. Discussions

A. *Security and Privacy Considerations*

Our design assumes that users are willing to share personal information at some level with strangers without their awareness. Although security and privacy is not the focus of this work, we can still provide privacy against eavesdroppers (passive attacks) due to the fact that Bloom filters use one-way hashing to encode the topics (contacts, interests, etc.), making it very difficult to reconstruct the original list of topics in a filter without performing an exhaustive search of the topic-value space. To provide privacy against active attackers, we can utilize research results in the area of privacy-preserving set intersection, such as the cryptography protocols in [22]. However, their protocols incur extensive computation and communication costs, which do not fit in our situation.

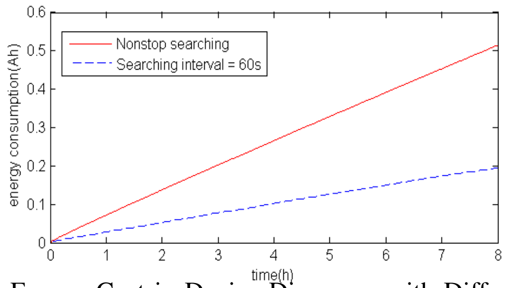

Fig. 9: Energy Cost in Device Discovery with Different Bluetooth Search Intervals.

In general, our design is a trade-off between privacy and performance given the constraints imposed by Bluetooth and mobile phones. We also assume that the communicating parties are in physical proximity, i.e., within 10 meters, which is the nominal communication range for class II Bluetooth devices. Within such a close physical distance, there is only so much attackers can do without being spotted.

Several possible attacks may compromise the users' privacy:

– An attacker can publish a Bloom filter consisting of a high percentage of random `1`'s such that everyone in the vicinity would recognize him as having common friends. Although this attack leads our system to provide false topics for the honest user, it does help the user start a conversation. To defend against this attack, we can modify E-SmallTalker to check that the percentage of `1`'s in any Bloom filter circulated in our system is below a certain threshold.

– An attacker may launch a man-in-the-middle attack by intercepting and forwarding other people's Bloom filters. On one hand, since the purpose of our system is just to provide topics for facilitating a conversation among strangers, the information being transferred is unlikely very sensitive. On the other hand, the middleman has no context information to decode the bits in an intercepted Bloom filter and it is very difficult to infer any useful information.

– An attacker may also copy contact information from a telephone directory and store it in bulk to his mobile phone. As a result, many users might recognize him as having common friends. However, this kind of attack is not very effective given the large number of contacts imported from the phone book. For example, suppose that the false positive rate is 0.1% and the number of contacts in the phone book is 300,000. Then the number of matches is $300,000 \cdot 0.1\% = 300$, which is very high considering that a person normally has only a few hundred contacts. Furthermore, this type of attack can be detected because of its unusually high percentage of `1`'s caused by the size limit of Bloom filters imposed by Bluetooth SDP. As in the first case, we can modify E-SmallTalker to ignore such Bloom filters.

B. *User Experience and Future Work*

Our current implementation works best when there are only two users in physical proximity. At the system level, the discovery protocol can identify all devices with matching interests. However, partly due to Bluetooth limitations, our system cannot yet tell who owns which devices. To identify this, one possible solution is to exchange users' pictures or descriptions (such as shirt color) after common topics are found and some privacy policies are met. Other complementary techniques such as Point&Connect [12] can be leveraged for one to point his device to the device of an interesting person. We plan to extend the system such that the user can tell specifically with which persons in a crowd he can talk about certain topics.

E-SmallTalker can also be extended to collect a rich variety of context information such as the time and location at which two users meet, who else is around when they meet, and the topics about which they conversed. It is also possible to retrieve public information about the users, e.g., from their personal web sites or other social networking services. Such information can make small talk topics more interesting.

Furthermore, it is important to reduce the total discovery time for a better user experience. By our experiments, the Bluetooth device discovery time accounts for over 90% of the time cost of our application. We plan to extend the system by reducing the Bluetooth device discovery time. We point to recent results that aim to accelerate Bluetooth device discovery, such as Scott et al. [19] and Woodings et al. [20]. These results can be leveraged to further improve performance of our system.

VI. Conclusion

We presented E-SmallTalker, a mobile phone-based distributed system for social networking in physical proximity among strangers. Our system suggested common topics for users to initiate significant conversations. Our system leveraged Bluetooth SDP to exchange these topics without establishing a connection. We customized service attributes to publish non-service-related information. We proposed a novel, iterative Bloom filter protocol that encodes topics to fit in SDP attributes to achieve a low false positive rate. Our approach was efficient in computation and communication. We have implemented the system and evaluated its performance on real-world phones. Our experiments and analyses illustrated our approach was promising for easing social interactions in physical proximity.

References

[1] N. Eagle and A. Pentland. "Social Serendipity: Mobilizing Social Software." In *IEEE Pervasive Computing*, *Special Issue: The Smartphone*, pp. 28–34, April 2005.

[2] Nokia Sensor. [Online]. Available: http://www.nokia-asia.com/A4416020

[3] Loopt. [Online]. Available: http://www.loopt.com

[4] K. Li, T. Sohn, S. Huang, W. Griswold. "PeopleTones: A System for the Detection and Notification of Buddy Proximity on Mobile Phones." In *Proc. 6th Int'l. Conf. on Mobile Systems (MobiSys)*, Breckenridge, CO, Jun. 2008.

[5] J. Kjeldskov and J. Paay. "Just-for-Us: A Context-Aware Mobile Information System Facilitating Sociality." In *Proc. 7th Int'l. Conf. on Human Computer Interaction with Mobile Devices & Services*, Salzburg, Austria, Sep. 2005.

[6] M. Terry, E. D. Mynatt, K. Ryall and D. Leigh. "Social Net: Using Patterns of Physical Proximity over Time to Infer Shared Interests." In *Proc. Human Factors in Computing Systems (CHI)*, ACM Press, pp. 816–817, 2002.

[7] B. Bloom, "Space/time tradeoffs in hash coding with allowable errors," In *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[8] MobiLuck. [Online]. Available: http://www.mobiluck.com

[9] L. E. Holmquist, J. Falk and J. Wigström. "Supporting Group Collaboration with Interpersonal Awareness Devices." In *Journal of Personal Technologies*, vol. 3, nos.1–2, pp. 105–124, 1999.

[10] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox and A. Shmidt. "Micro-Blog: Sharing and Querying Content Through Mobile Phones and Social Participation." In *Proc. 6th Int'l. Conf. on Mobile Systems (MobiSys)*, Breckenridge, CO, Jun. 2008.

[11] M. Motani, V. Srinivasan, and P. S. Nuggehalli. "PeopleNet: Engineering A Wireless Virtual Social Network." In *Proc. of the 11th Annual Int'l. Conf. on Mobile Computing and Networking (MobiCom)*, pp. 243–257, 2005.

[12] C. Peng, G. Shen, Y. Zhang and S. Lu. "Point&Connect: Intention-based Device Pairing for Mobile Phone Users" In *Proc. of the 7th Int'l. Conf. on Mobile Systems (MobiSys)*, Jun. 2009

[13] A. Kirsch and M. Mitzenmacher, "Less hashing, same performance: Building a better Bloom filter," In *Proc. of 14th Annual European Symposium on Algorithms (ESA)*, pp. 456–467, Sept. 2006.

[14] L. Fan, P. Cao, J. Almeida, A. Broder, "Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol". In *Proc. SIGCOMM*, pp. 1361, 1998.

[15] M. Mitzenmacher. "Compressed Bloom filters". In *IEEE/ACM Transactions on Networking*, 10(5):604–612, 2002.

[16] D. Guo, J. Wu, H. Chen, and X. Luo, "Theory and Network Applications of Dynamic Bloom Filters", In *Proc. of the 25th IEEE Int'l. Conf. on Computer Communications (INFOCOM)*, Barcelona, Catalunya, Spain, April 2006.

[17] J. Bruck, J. Gao, and A. Jiang,"Adaptive Bloom Filter". *Technical Reports of California Institute of Technology*, No. 72, 2006.

[18] Y. Matsumoto, H. Hzeyama and Y. Kadobayashi,"Adaptive Bloom Filter: A Space-Efficient Counting Algorithm for Unpredictable Network Traffic". In *IEICE - Trans. on Information and Systems*, v.E91-D n.5, pp. 1292–1299, May 2008.

[19] D. Scott, R. Sharp, A. Madhavapeddy, and E. Upton, "Using Visual Tags to Bypass Bluetooth Device Discovery", In *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 1, pp. 41–53, 2005.

[20] R. Woodings, D. Joos, T. Clifton, and C. D. Knutson, "Rapid Heterogeneous Connection Establishment: Accelerating Bluetooth Inquiry Using IrDA." In *IEEE Wireless Communications and Networking Conf.*, pp. 342–349, vol. 1, 2002.

[21] Bluetooth Specification Version 2.0 + EDR [Online]. Available: http://bluetooth.com/Bluetooth/Technology/Building/Specifications/

[22] M. Freedman, K. Nissim and B. Pinkas, "Efficient Private Matching and Set Intersection", in Proc. of Eurocrypt, LNCS, Springer, vol. 3027, pp. 1–19, 2004.