I Seek for Knowledge: Exploiting Social Properties in Mobile Ad Hoc Networks

Sebastian Kay Belle* Muhammad Arshad Islam* Marcel Waldvogel*
* University of Konstanz
Distributed Systems Laboratory
Konstanz – Germany
firstname.lastname@uni-konstanz.de

***Abstract*—New social networks are born each day, at a formal conference, at informal social gathering, at family reunions etc. Internet has already been playing an important role to let people socialize through online social websites. For many users, its still not the optimal way of interaction as one has to be very active updating their activities on the online profiles. With the easy access to mobile devices, modern technologies have now started to adopt to more human of socializing. As these mobile devices accompany their users almost all the time, they can record and observe their users behavior as well as gather information about their social circle. Therefore, they can help users to get information from contacts, that they potentially not even know. In this paper we put our efforts towards the initial design of such an architecture, we call *Mergenet*, that will sniff for information around the user's surrounding, leveraging useful answers on their demand.**

I. Introduction

Nowadays, we face a novel kind of social networking, *virtual social networks* as enforced by well known platforms like *Facebook*, *MySpace* or *Friendster* – just to name a few – that have more than 50 million registered users. This new approach to social networks somehow reflect the old habit of mankind that people tend to arrange themselves in groups with similar interests and share their knowledge.

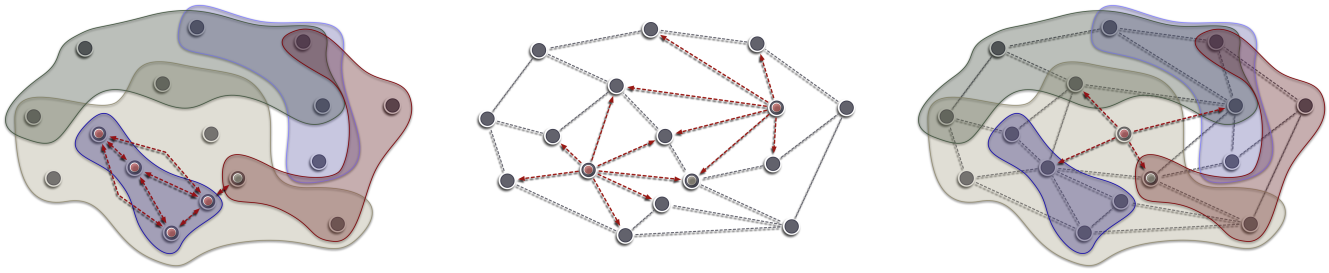
Modern social networking also reflects another aspect of today's' life. Information is no longer solely stored and found in databases or documents in the World Wide Web. Instead information, or to be more precise, *knowledge* is shared by users worldwide. Hence, social networks become even more prominent when searching for information to solve a problem. Simplified, it all comes down to the well known adage:*"It's not what you know, it's who you know"*. Therefore, the question we face is less *how we find the information,* but more how we get in contact with *someone who has the expertise*. Such *group-forming networks*, as discussed in Reed's law [**?**], have a utility which grows exponentially with the number of participants, unlike traditional networks whose utility merely grows linearly or quadratically. Anyway, note that the term "friend" gets somewhat altered in the context of this new type of social network. Relationships become *binary*, either you are connected or you are not connected. These social networks alter the type of relationship between people, personality becomes somewhat irrelevant, only the mere interest for a specific topic connects one person to another. Further, modern social networking bypasses the geographic context in which people reside. Technology simplifies it to stay in contact with others, even in remote locations of our planet, for instance by e-mail or chat. However, the most obvious device we use to stay in contact with others – and probably nowadays the most *personal* device – are mobile phones.

Mobile phones, or mobile devices in general, can help us to bring social networking to the next step, to mobile social networking. Therefore we first define why it is important to rethink what we like to achieve with modern social networks, second, what difficulties we face when we like to integrate these aspects in the next type of social networks.

1) Why do we care?: (1) Connecting to experts that have some specific knowledge could facilitate problem solving as these experts can provide quick access to answers we could not solve on our own, or at least, only with a high effort. (2) Modern social networks work on a *pull* type approach, where users explore the available connections in the network. However, a *push* type approach, as we know it from *real-world social networks* when we get introduced to others by colleagues of friends would ease the use of the system. This *push* approach can be seen as some kind of recommendation to quickly dig out experts. (3) Related to the previous point is the third important aspect that is specifically inherent to virtual social networks, *transitivity*. Transitivity is a powerful way to explore social connections by following links to n^{th} degree contacts of our direct contacts. However, due to the *pull* approach, utilising transitivity is somewhat limited as the user is responsible to dig out experts and does not get any recommendations.

2) What problems to solve?: (1) Modelling an automatic recommendation system based on the *push* approach. (2) Modeling the transitive relationship in a system that facilitates the desired *push* approach. (3) Including social information of the users in the social network.

We propose a system based on mobile devices, thus, building an implicit infrastructure for mobile social networking. The proposed system integrates social information of its users, stored in Bloom filters [2], to facilitate the look-up of other users that could be of potential interest. We define a two-way message protocol that exploits the social network's inherent transitive connections, enhanced with the social information network's users, to route messages to a suitable receiver.



(a) (b) (c)

Fig. 1: Simplified illustration of the concept of semantic group models fig.(a), centrality measurements fig.(b) and the enhancement of connections in a social network by exploiting social information about the nodes fig.(c).

Further, this message protocol enables a receiver to reply to the initial sender, using a pointed multicasting approach, while maintaining the replier's privacy as well as the privacy of the initial sender.

II. Related Work

Following, we give a brief overview of related work in the area of *Mobile Ad Hoc Networks* (MANETs) and *Delay Tolerant Networks* (DTNs), especially on research related to exploiting social information in these networks. Further we define some basics on Bloom filters we will use later on in section III.

3) Exploiting Social & Semantic Information in Networking: Researchers in social sciences have been debating about several parameters like No. of contacts, closeness to contacts, location in the network etc. that help classify the status of someone in his social environment, but, the concept of *Betweenness centrality* [4] has gotten the most attention. Daly and Haahr [3] propose a metric to identify characteristic nodes in a delay tolerant MANET that can serve as bridge nodes to pass messages to the intended receiver. To identify these nodes, Daly and Haahr exploit social analysis techniques as a measurement of centrality in a MANET to enhance routing. Their idea was derived from the characteristics of the *small world phenomenon* which states that individuals are often linked by a short acquaintances [**?**]. Miklas et al. [8] exploited the use of social information in mobile systems, conducting a simulation on the "Reality Mining" dataset of the MIT Media Lab [**?**]. In their simulation they classified the users into "strangers" and "friends" due to their number of encounters during the experiment. Their simulation has shown that this simple classification already yields a gain in the message delivery performance in DTNs in terms of the time a message takes to arrive at its destination. Zhao et al. [12] introduce semantic models for efficient routing in DTNs by defining group membership models (Temporal Membership, Temporal Delivery and Current Membership) for routing messages through the network. These models define the membership group, that may be interested in receiving a message from the network depending on the location and/or time A similar semantic model was proposed by Gong et al. [5].

In contrast to the approaches of [5], [12] we follow the idea of [8] to utilise social information. However, we not only classify users into "friends" and "strangers" but add a third class to the system, "virtual friends". We define this classification in more detail in section III.

4) Bloom Filters in Networking: Bloom filters [2] are simple space-efficient randomized data structures that represent a set of elements and that support membership queries while allowing false positives. A Bloom filter is a bit array of length m, with k hash functions, that can hold up to n, $n < m$ elements. To add an element into the Bloom filter, the element is hashed to k different indices of the Bloom filters bit array by applying the hash functions. Each of the k bits designated by the hash functions is then set to 1. If a bit would be set by different elements the bit stays set. The probability p_{err} of a false positive depends on the three parameters m, n, k of the Bloom filter and is calculated as::

$$p_{err} = (1 - e^{-kn/m})^k \tag{1}$$

Given a fixed size m for a Bloom filter as well as the intended (maximal) number of elements n, the optimal number of hash functions to uses can be obtained as[1]::

$$k = \ln 2 \cdot (m/n) \tag{2}$$

Bloom filters have been used to solve a variety of networking problems and different variations of Bloom filters have been proposed in the last years, most of them suited towards a specific application area. Mitzenmacher [9] introduced the concept of compressed Bloom filters and has shown that, by some modifications to the original concept, the size of a Bloom filter can be reduced without increasing p_{err}. Guo et al. [6] proposed dynamic Bloom filters that can grow as needed by adding static sized Bloom filters to a list of Bloom filters as soon as the actual Bloom filter's false positive rate increases over a given threshold. Bauer et al. [1] propose simple boolean set operations on Bloom filters to combine sub-queries for efficient queries on distributed hash tables. However, Bloom filters are by far not the only way to go. Hurley and Waldvogel [7] have shown that in the case that false negatives are admissible, Bloom filters can be outperformed by other techniques.

[1]Note that in practice k is rounded down to the next integer as this, in general does not increase p_{err} dramatically and yields better performance.

We use Bloom filters to encode the social information of a user as they allow efficient set operations as shown in [1]. As already briefly stated, we enhance our system with this user specific information and further utilise this information in the look-up procedure of potentially interesting contacts as well as in our two-way message protocol. Therefore we can tolerate false positives, whereas false negatives are not preferable.

III. Mergenet Architecture

We assume that users of our system are in possession of mobile devices capable of P2P communication. We aim *Mergenet* to be flexible enough to let a user establish a P2P connection through arbitrary techniques like WAN, LAN, Wifi, or Bluetooth. A user's profile is present on the mobile device, giving the responsibility to the user to keep his profile updated, especially if he uses multiple devices. Details about the profile will be presented in section **??**.

Mergenet takes into account the disparity among the users with respect to their social activities. Some users may like to be very social by introducing themselves to as many people as possible while on the other extreme, some may want to keep a very limited activity by keeping a small number of close contacts. *Mergenet* classifies contacts as "strangers" and "friends" similar to the idea of Miklas et al. [8] depending on the frequency and amount of time the devices of two users see each other. However, in contrast to [8], we let the user also add "trusted friends" manually or, depending on recommendations of the system, let the user choose to add a "virtual friend" in his list of trusted contacts. A contact will be classified as trusted friend if both users agree to identify each other like this and someone who happens to pass by user very seldom can be called a stranger. The set of virtual friends C_{VF} is an intersection of both, trusted friends C_{TF} and strangers C_S depending on a threshold α as::

$$C_{VF} = C_{TF} \cap C_S \ \exists T_\alpha \geq \beta, \tag{3}$$

where $T_\alpha = (\sum Wt)/Mc$. Here Wt is the waiting time, users had to wait for each other and Mc is the number of meeting count. We will use this threshold when $Mc > \alpha$ to have realistic classification.

Following, we present two definitions we will use in the next sections.

- A *client* is a user that is interested either to introduce himself or to inquire some specific information.
- A *responder* is a user that may be interested in getting to know a new client or he likes to respond to a query from a client, either by himself or by knowing someone who could give and appropriate response.

A. Profile Structure

We propose a profile structure based on Bloom filters [2] where every profile will contain a set of Bloom filters in a stack $BS_i = \{B_1^i, B_2^i, \ldots, B_j^i\},\ i, j \in \mathbb{N}_1$. BS_1 denotes the Bloom filter stack for the current user while $i > 1$ denote the Bloom filter stacks for the users that came in direct contact with the current user. The first Bloom filter in each stack denoted by B_1^i is considered mandatory and contains the minimum information for the profile to exist and will always be populated with at least identification information of the users, while rest of the Bloom filters correspond to hobbies, activities, music, movies and other interests. For every Bloom filter in the stack we compute the MD5 check sum so that other users in the networks see that the a profile of a contact got updated. Formally, let $CS_j^{B^i} = F^B(B_j^i)$ be the check sum for the Bloom filter B_j^i computed by $F^B()$, then the check sum of BS_i would be computed as $CS_i^{BS_i} = F^{BS}(CS_k^{B^i}),\ k = 1, \ldots j$ This helps to shortlist the number of Bloom filters in the case of profile exchange. Moreover, this also helps to control the extent to which users may want to socialize with each other.

Whenever the user would like to update his profile, the changes will be hashed in the corresponding Bloom filter and the check sum is recomputed. Whenever two users will come in contact with each other for the first time, they will create a new Bloom filter stack for each other and increment the corresponding values of i. At the time of profile exchange, *Mergenet* will compute the union of all Bloom filters at one level of the stack before transmitting them. Formally, let $BS' = \{B'_j\},\ j \in \mathbb{N}_1$ be the Bloom filter stack to be transmitted, this can be expressed as[2]::

$$BS' = \forall\ BS_i : \cup_j B_j \text{ where } j > 1. \tag{4}$$

This way we can ensure the transitivity property in the network i.e. if user A shares her profile with B and B shares with C, C can also see the effects of A's profile without knowing A directly. If C would like to refer to something in A's profile, she can do it via B, keeping the privacy of A. Moreover, *Mergenet* only permits the replication of a profile, if both clients identify each other as friends, otherwise they will be categorized as strangers and only basic profile information will be exchanged that may be used in routing the information. Fig. 2 illustrates the data maintained per user in the system.

The size of any Bloom filter $B_j \in BS_i$ will be kept flexible by using the strategy introduced in [6]. In this technique, whenever the existing Bloom filter gets heavy to sustain the false positive rate, another Bloom filter of a specific size is created while the old one is sent to storage and treated as legacy Bloom filter. From that point on, whenever the Bloom filter is to be consulted, we consult the most recent one added. This way, the clients with low populated profiles with not be burdened by unfilled Bloom filters.

B. Protocol definition

The transmission protocols consists of two kinds of exchange of data i.e. Profile exchange and Information exchange.

1) Profile Exchange: Mergenet identifies every client $c \in C = \{1, 2, ...\}$ by a unique identifier. Whenever two users are connected to each other by wired/wireless medium, one of them assumes the role of *Client* and the other as a *Responder*. The possible deadlock in this situation can be removed by

[2]Note that we exclude the mandatory Bloom filter B_1 in the aggregation.

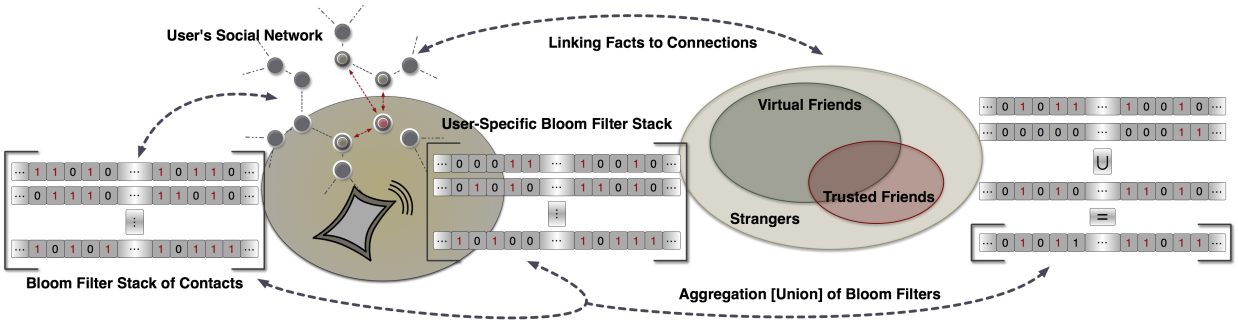


Fig. 2: Illustration of the user profile as stored on the mobile device. The user's social contacts are classified according to the Venn diagram on the right.

attempting random waiting and reattempting to acquire the role. When one device has acquired the role of client,

- c send its identification to every responder r.
- r looks up at its contacts list and adds c to the contact list if r is not already present.
- r responds with the list $BFlist$ of BloomFilter identifiers in which she is interested. if r is not really interested in socialize then she will only request one BloomFilter containing mandatory information about c.
- c sends back the checksums $CS_{j,k}$ where $j \in BFlist, k$ is the index of current BloomFilter.
- r matches the checksum with the exisiting checksums and replies back with the list of BloomFilter identifiers that need to be updated.
- c sends the new BloomFilters and thus updating the profile at r.
- r computes the new $B_{x,j}$ for all the updated BloomFilters as well as the new corresponding checksums.

2) Information Exchange: Information exchange includes any kind of queries that different users may ask among each other. Mergenet will try to find the suitable responders to respond to a query by directing the query to only those users who have common tastes in the same category to which query belongs. This means Mergenet should know in advance that to which classes the query relates to. Every query has a lifetime limited by *MaxLifeTime*. A client c may ask several queries with overlapping lifetimes.

Mergenet purposes that c maintains the list of all alive queries Q=$\{0,1....q\}$ sorted by their expiry time with high priority for the queries with less remaining lifetime. Mergenet assumes that every user has already exchanged the latest profile according to procedure described in the previous section.

C. Query Privacy

Mergenet purposes a proxy enveloping technique to ensure the privacy as well as optimised transmission of reply to the source. Fig.4 shows the UML class diagram of our envelope model, fig. **??** depicts how the source envelopes the message for the first time and all the forwarding clients wrap it in their own envelopes, inserting their own identification as well of

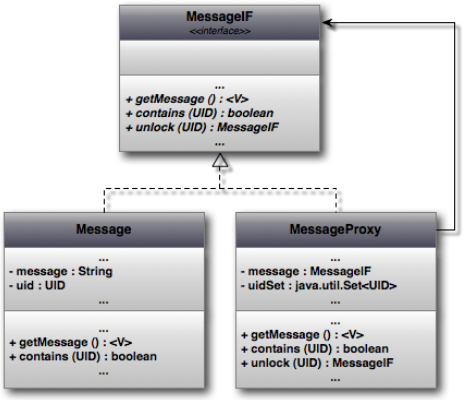


Fig. 3: The UML class structure of the message objects used in our system utilising the *Decorator* design pattern. The message interface exposes three methods. One to retrieve the initial query of a user, the second to check if a UID encountered during back routing is contained in the set of users that are a potential way back to the client, and finally, third a method with which the creator of a message envelope (message proxy) can unlock his own envelope (as well as all envelopes above his own) and discard it.

their trusted friends. Any client, on the way, cannot access the information in the proxy envelopes but rather only check that given a client ID, whether that id exists in the envelopes or not. This way, the responder of the query can respond to the trusted friends as well as to any client that is present in the proxy wraping without knowing that its the sourse or forwarder of the query.

Mergenet follows different strategies for forwarding of queries and responses. Every user maintians a querylist, a responselist and she tries to attain the status of client, if she has either at-least one alive query or response of alive query. If its not successful than it waits for random amount of time and then retries. When one device has acquired the role of client,

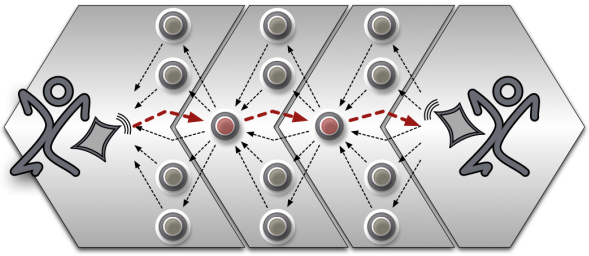


Fig. 4: Illustration of the two-way message protocol. The thick red edges correspond to the query of the client on the left, routed towards a potential expert on the right. During the hops the initial message is embedded in several message envelopes, each containing the information about the forwarding node (red) as well as some additional nodes (beige), included by the forwarding node, that represent potential retrieving candidates. Any of these additional nodes can serve as a proxy for the responding node in the corresponding hop-layer, passing the message to its original forwarding node in this layer. The original forwarding node then unlocks the envelope with his UID and proceeds with sending the message to the nodes in the previous layer.

D. Query Forwarding

- For each of the alive query$q$, Client c, determines, to which interest BloomFilter does the query belong?
- looks into *responselist* to check whether she has already has response from r
- if no, c check the corresponding BloomFilter stack of responder r
- if the interest match, wraps the query into proxy envelope with IDs from her contacts that she sees often(virtual friends) and forwards the query to r and starts waiting for response.
- if interest fails to match, c discards r as potential responder and tries to query the next r.
- r on receiving the query id, checks whether she can respond it, if yes she directly responds the query to any of the client according to procedure in next section.
- if no, r looks for a response in the *responselist* if yes she responds the query
- if no, r tries to forward to her contact that has the common interest with her proxy envelope.
- When she gets the response from her contact, she saves it in her *responselist* with the expiry time of q.

E. Response Forwarding

- For each of the alive response p, Responder r checks whether the client is present in the proxy envelope list.
- if yes, r transfers the response to c.
- c removes the proxy envelope and proceeds with procedure if there still exist another proxy envlope.
- if there is no more proxy envlope, this means c is herself the creator of this query.

Mergenet purpose to use the betweenness criterion whenever the the client is unable to find any responder with the matching interest. As discussed in [4], *Betweenness* associate a value depicting, how central a node is in a network. In other words, a node having good betweenness value is center point between several communities/clusters. Therefore, a client will send the query to everyone who has a better betweenness value than herself. This way, the query may reach to the corresponding user that has the value able response.

IV. Conclusion and Future Work

Mergenet is a system that assists user to socialize in way that online social network cannot do. It helps them to make contacts in real life and lets them use these contacts to share as well as gather information. As data intensive applications like MMS and Mobile television are on their way, we intend to test this architecture in real world by focusing on gathering maximum information by transmitting the least amount of data in minimum possible time. This also involves BloomFilters compression without loosing their efficiency. We have tried to quickly route the information without losing privacy but there is much more to be done to call Mergenet very secure.

References

[1] D. Bauer, P. Hurley, R. Pletka, and M. Waldvogel, "Bringing efficient advanced queries to distributed hash tables," *lcn*, vol. 00, pp. 6–14, 2004.

[2] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[3] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2007, pp. 32–40.

[4] M. Girvan and M. E. Newman, "Community structure in social and biological networks." *Proc Natl Acad Sci U S A*, vol. 99, no. 12, pp. 7821–7826, June 2002. [Online]. Available: http://dx.doi.org/10.1073/pnas.122653799

[5] Y. Gong, Y. Xiong, Q. Zhang, Z. Zhang, W. Wang, and Z. Xu, "Anycast routing in delay tolerant networks," in *GLOBECOM'06: In Proceedings of the Global Telecommunication Conference 2006*, San Francisco, CA, USA, November 2006, pp. 1 – 5.

[6] D. Guo, J. Wu, H. Chen, and X. Luo, "Theory and network applications of dynamic bloom filters," in *INFOCOM*, 2006.

[7] P. Hurley and M. Waldvogel, "Bloom filters: One size fits all?" *lcn*, vol. 0, pp. 183–190, 2007.

[8] A. G. Miklas, K. K. Gollu, K. K. Chan, S. Saroiu, K. P. Gummadi, and E. de Lara, "Exploiting social interactions in mobile systems," Innsbruck, Austria, September 2007.

[9] M. Mitzenmacher, "Compressed bloom filters," in *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 2001, pp. 144–150.

[10] M. Musolesi and C. Mascolo, "Spatio-Temporal Communication Primitives for Delay Tolerant Systems," in *Proceedings of 3th MiNEMA Workshop*, Leuven, Belgium, February 2006.

[11] D. P. Reed, "That sneaky exponential—Beyond Metcalfe's law to the power of community building," `http://www.reed.com/Papers/GFN/reedslaw.html`, 1999.

[12] W. Zhao, M. Ammar, and E. Zegura, "Multicasting in delay tolerant networks: semantic models and routing algorithms," in *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM, 2005, pp. 268–275.