# FRoDO: Friendly Routing over Dunbar-based Overlays

Tobias Amft*, Barbara Guidi†, Kalman Graffi* and Laura Ricci†
* Technology of Social Networks Group, Heinrich-Heine University Düsseldorf, Germany
Email: [amft, graffi]@cs.uni-duesseldorf.de
† Department of Computer Science, University of Pisa, Italy
Email: [guidi, ricci]@di.unipi.it

***Abstract*—Centralized Online Social Networks (OSNs) have become the main communication channel in both the personal and the business domain. A current trend for developing OSN services is towards the distribution of the social network infrastructure by using P2P architectures as basis for Distributed Online Social Networks (DOSNs). One of the main challenges of DOSNs comes from guaranteeing privacy and protection of private data. In previous work [18], we proposed a Dunbar-based approach to preserve data availability in DOSNs. Using Dunbar's circles of intimacy a certain level of trust is ensured which bases on the users confidence in their friends. Now, to achieve privacy and anonymity, we focus on the incorporation of social contacts into existing Peer-to-Peer Overlays and show that a naive integration of social links into existing Overlays like Chord and Pastry is not satisfactory. In order to address drawbacks of the naive approach we introduce *goLLuM*, a general solution which can be used on top of existing structured and unstructured P2P networks. Our protocol enables to route messages via friendly nodes only, even if only few friends per node exist. By using synthetic models and real-data traces for the representation of friendship relationships we highlight the drawbacks of the naive solution and show the functionality of *goLLuM*.**

## I. Introduction

In the last decade, Centralized Online Social Networks (OSNs) have become the common communication channel between people. One of the main problems in these systems is the privacy and protection management. Once users have published their personal data within a OSN, they lose the control over their data with a high risk of privacy breaches and malicious exploitation. In the last few years, the distribution of the social network infrastructure, often through a peer-to-peer (P2P) approach, has been widely studied. A Distributed Online Social Network (DOSN) [11] offers a real cost-effective alternative and, without a central authority which can claim the right to exploit the data, it ensures a better control of data. However, decentralization introduces new several critical problems that do not occur in centralized implementations (i.e. in a P2P DOSN data can be potentially accessed by everyone). Most of the proposed P2P DOSNs provide mechanisms to guarantee privacy and protection over data and also they define a structured overlay in which nodes are connected to nodes that should be friends in real life. This kind of networks are called Friend-to-Friend [4] (F2F) networks and they are a subset of the most general social P2P networks. An interesting problem which has to be faced is that, in most social networks, the number of relationships each user can have is very high (about 5000 in Facebook). Studies on the most common OSNs have shown that only a subset of the friend's set is really active [1], [2]. In a distributed environment, the management of inactive connections takes too much time and effort. In our previous works [17], [18] we have developed a Dunbar-based Distributed Online Social Network based on Pastry [25] which provides trust without any privacy policies or mechanisms because of the Dunbar approach and it provides trusted services for data availability [8] and for information diffusion [9]. The Dunbar approach permits to have an efficient system in which private data are stored only on friend nodes. The most critical point of systems like the one proposed in [8] is the usage of a DHT. By using a DHT, data can be accessible to everyone and more powerful mechanisms are required to ensure trust and anonymity among participants. To build a more trusted system, we want to improve the system by guaranteeing a friendly routing functionality which is completely embedded into the structured overlay we are using. Our intuition is to modify the routing of our Pastry DHT to guarantee a certain level of trust. By integrating Dunbar's approach (social links) into Pastry (*SocialPastry*), we are able to direct the lookup through trusted nodes. On the other hand, some basic properties of Pastry routing are not longer guaranteed, because of the naive modifications. We identify incompatibilities of combining Pastry with social Dunbar links and demonstrate drawbacks which occur in *SocialPastry* to sensitize the reader that social contacts based on Dunbar can not be integrated into existing DHTs effortlessly. To address these drawbacks we provide a more general approach which can be used in structured and unstructured overlay networks because it does not modify routing in the DHT, but can be used over the routing service. In this work, we evaluate how far lookups are possible in Pastry if messages are only forwarded to trusted nodes and we show the drawbacks of this approach in terms of how the number of failed lookups increases if the number of possible contacts in the overlay's routing table decreases. We show an increasing number of visited nodes one message has to traverse if the number of friends is reduced. Furthermore, we show the evaluation of (*SocialPastry*) with a comparison of *goLLuM*, our proposed solution, by using real Facebook traces and two well-known models: Scale-free Graphs [3] and Small World graphs [29]

### A. Outline

The remainder of the paper is structured as follows: Section II discusses the related work regarding DOSNs, Dunbar's properties, structured P2P overlays and the integration of social Dunbar links into DHT-based overlays. In Section III we briefly describe our system and we propose the naive way to apply the Dunbar scheme to a structured P2P overlay, called *SocialPastry*. In Section IV we explain goLLuM, an algorithm which ensures successful message delivery over trusted nodes. Section V describes how friendship relationships are assigned to nodes during simulation. We compare SocialPastry to the goLLuM algorithm in Section VI, and conclude with SectionVII.

## II. Related Work

This section discusses the related work and presents an overview of existing DOSNs, the Dunbar approach, and structured P2P overlays. Furthermore, we discuss studies similar to ours.

### A. Distributed Online Social Networks

In the last years, several P2P DOSNs have been proposed. LifeSocial.KOM [15] is a P2P platform for OSNs which provides a security infrastructure that enables secure communications and access control on the stored data. Users employ public-private key pairs to encrypt profile data that are stored in a FreePastry-based DHT. PeerSon [5] is a distributed two-tiers infrastructure for social networks in which the lowest tier consists of users and their contents and the upper tier provides lookup services. Safebook [10] defines a DOSN based on two design principles: decentralization and exploiting real-life trust. The system integrates several privacy and security mechanisms in order to provide data storage and data management functions that preserve data integrity, privacy and availability. SuperNova [26] provides all the social services by using a super-peer architecture. Cachet [23] is an architecture that provides security and privacy and it guarantees the confidentiality, the integrity and the availability of the user content. It manages the data availability problem through the collaboration of users, but while it ensures availability, it also increases the communication overhead.

### B. Ego Network and Dunbar's concept

Collecting data about entire social networks is often infeasible, and researchers are focused on personal social networks of single individuals, called ego networks [13]. An ego network consists of a single actor (ego) together with the actors that are connected to it (alters) and all the links among those alters. The most important result found on ego networks is that the cognitive constraints of the human brain and the limited time that a person can use for socializing bounds the number of social relationships that it can actively maintain in its network. As shown by Dunbar [12], this limit is about 150 and is called the *Dunbar number*. Some studies aimed at analyzing offline social networks depicted these networks as being composed by several concentric circles around the ego [12]: these circles reflect different social ties, and the tie strength fades as the distance from the ego increase. Each of these circles has typical size and frequency of contact between the ego and the alters contained in it. This inclusive concentric sequence of circles is called *Dunbar's circles*, with an increasing level of intimacy. The inner circle, called *support clique* is characterized by increasing level of tie strength while outer circles, called *sympathy group*, *affinity group* and *active network* are characterized by a low level of tie strength. Several studies have evaluated the presence of Dunbar's circles in different online social networks contexts and showed that it is possible to identify their average size as 5 (*support clique*), 15 (*sympathy group*), 50 (*affinity group*) and 150 (*active network*) with a scaling factor approximately equal to three [28], [1], [14], [2]. The design of a measure of tie strength is still an open issue. Some studies such as [1], [2] use frequency of the online interactions to estimate tie strength, since the two concepts appear to be tightly correlated.

### C. Structured Overlays

Structured P2P overlay networks like CAN, Chord, Pastry offer an interesting platform for a variety of distributed applications. Pastry is a P2P overlay network with good performance and high reliability [25]. It is completely decentralized, scalable, and self-organizing and the overlay graph design of Pastry also tries to target the issue of locality. Like Pastry, Chord [27] uses a circular id space, but it forwards messages only in clockwise direction in the circular id space. CAN [24] is a distributed content addressable P2P infrastructure designed around a virtual multidimensional coordinate space. The coordinate space is randomly partitioned into zones, and each node is the responsible of a zone. Kademlia [21] takes the basic approach of assigning each peer an identifier in the 160-bit key space. It defines a novel routing metric in which the distance between two nodes is the numeric value of the XOR of their identifiers.

### D. Social Links in DHT-based Overlays

The idea of integrating social links into P2P overlays is not completely new but the assumptions and implementations differ. Freenet [7] is an unstructured overlay which provides DHT-like storing and retrieval of document sets while sender and receiver of messages are promised to stay anonymous. Requests for specific documents are forwarded among participants by selecting neighbors as next hop which are numerically closest to the target identifier. Nodes feel responsible for a request if they are closest to the given identifier among their neighbors. As nodes in Freenet are exposed to their neighbors, a darknet version [6] has been released which allows participants to connect to their friends in a F2F-manner only. Our research is different from Freenet in the way that we do not consider social links that consist of mutual friendship between nodes. Links in Dunbar's circles are associated with a tie strength which represents the frequency a node is contacted by others. Friendship is therefore not necessarily mutual in our approach, as an example one node

could be contacted frequently while it answers rarely. The authors of SROUT [19] are investigating the effect of social links in Chord. Their aim is to improve routing performance in Chord in terms of reduced latency and routing failures. Nodes maintain friendship connections to 8-9 nodes which are preferably used during forwarding process which is the following: messages are forwarded to friends whose identifiers are closest to the target but not greater than the target node. This procedure is repeated as long as such a friend exists. Regular Chord is used if no further friend exists whose identifier is closer to the given target identifier. In order to improve routing, sprout uses a lookahead of one or two nodes and a minimum hop distance to regulate how far a distance should be traveled at least with every hop, i.e. to regulate how fast the target is approached. Sprout assumes that nodes far away might drop packages as they are friends of friends and not trustful anymore. We assume on the contrary that nodes do not drop packages as they do not know if a friend was asking for a request or a friend of a friend. Also, we do not allow a lookahead to friends of friends as it might reveal the structure of the local network. Since regular DHTs comprise more links than only those to trusted nodes, they are insufficient for anonymity purpose and therefore a modification of standard DHTs is desired.

The novelty in our work compared to existing studies lies in the combination of Dunbar's circles with DHT-like routing. While existing approaches assume social links to be based on mutual friendship we investigate Dunbar-based relationships which can be formed by the contact frequency (tie strength) between two nodes and allow therefore directed, one-sided relationships to be established.

## III. Social Dunbar-based DHTs: SocialPastry

Our previous system has been composed by a P2P social overlay, where the local view of each node is obtained by applying Dunbar's approach [17], [18]. As explained in [8], the size of an ego network is limited: each user is connected only to a reduced number of friends arranged in a hierarchical inclusive sequence ordered by increasing level of intimacy. Our system requires a lookup level to provide all the social network services. Currently, the look up level exploits the Pastry DHT. The main goal of our system is to provide a certain level of trust and anonymity which can not be assured using Pastry because information about storage nodes are kept inside the global accessible DHT. This is the principal motivation for the definition of a social Dunbar-based DHT like *SocialPastry* which allows routing over friendly nodes.

Out of many popular structured P2P overlays (CAN, Kademlia, Chord, Pastry) we have arbitrarily chosen Pastry as representative structured P2P overlay for our system to present drawbacks of a naive approach. Pastry has a one dimensional identifier space and routing is possible in $O(logN)$. Each node is uniquely responsible for a well-defined part of the identifier space. Instead, in Kademlia, for example, depending from which node in the network a request starts, the request does not lead deterministically to the same node (because of $\alpha$ parallel lookups). To define a social Dunbar-based Pastry DHT, the overlay has to be extended by a friend-routing table, which comprises friends of one node (friends are also sorted by level). We call this protocol *SocialPastry*.

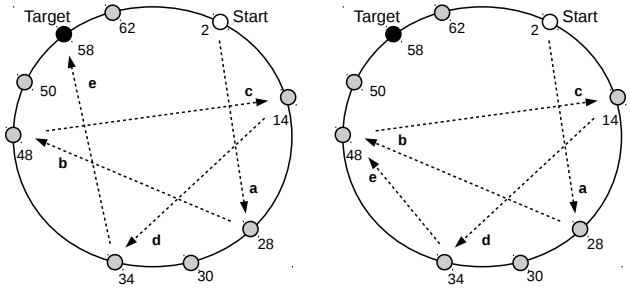
To obtain *SocialPastry*, the routing table is modified in such a way that original routing behavior is conserved while at the same time contact information to friendly nodes is stored. Instead of storing one contact node per entry in the table we extend the entries so that a set of $k$ nodes can be stored inside. Additionally, a level of closeness is associated to each node in the table which refers to the strength of the personal relation of a node $n$ to that node. The closeness level corresponds to the Dunbar's circle where the friend is located.

Given a target identifier $ID_t$ for a lookup request, the SocialPastry protocol acts as follows:

- Node $n$, which has to find the next hop of the lookup request examines if the given target identifier $ID_t$ is in the range of its leaf set. That is to decide whether $ID_t \in [L_{ccw}, L_{cw}]$ or not, where $L_{ccw}$ is the node in the leaf set farthest away from $n$ in counter-clockwise direction in the id space and $L_{cw}$ is farthest away from $n$ in clockwise direction. If so, node $n$ has to determine the next hop of the lookup request, which is the numerically closest node to $ID_t$ in the leaf set, whose closeness level is below or equals a given threshold $\lambda$ at the same time.
- If $ID_t$ is not in range of $n$'s leaf set or all closeness levels are greater than $\lambda$, the next hop is considered to be found in Pastry's routing table. Similar to original Pastry, the contact node with the longest common prefix to $n$ will be the next hop of the lookup request. Instead of storing only one Pastry contact per routing table entry, we store for each entry the original contact node as well as those friendly nodes which share the same common prefix with the original node. The lookup request will be forwarded to one node in the entry set whose closeness level is not greater than $\lambda$.
- In case that no contact node has been selected as next hop so far (e.g. if closeness levels did not satisfy $n$'s decision), $n$ picks the first fitting node of the next numerically closest nodes which fulfills the requirements of the closeness threshold.
- If no contact node is found (e.g. if node $n$ does not have any friend stored in the routing table), the lookup request is stopped.

SocialPastry fails if any node $n$, which is responsible to forward a certain lookup request, has no friends in its routing sets or the number of maximal hops for a lookup request message is exceeded. Therefore, the number of failed lookups should increase with decreasing threshold value $\lambda$, since more hops have to be traversed. Due to these drawbacks, we have implemented a general protocol which is independent from the P2P overlay to overcome the routing issue and, on the other hand, to apply friendly (social) routing on top of any structured and unstructured P2P overlay.

Another pathologic case in which SocialPastry or other naive approaches fail is shown in the following example.




(a) Successful path of a lookup (b) Lookup fails due to routing loop

Fig. 1. Routing loops are possible in naive approaches for Social DHTs

Consider the topology shown in Figure 1(a), node 2 starts a lookup to target node 58. Node 2 considers node 28 to be next hop. 28 selects node 48 as next hop, etc. The message is forwarded until the target node is found. In Figure 1(b) node 34 does not consider node 58 as friend. Instead it forwards the lookup message to node 48 which still considers node 14 to be best suited as next hop. As a result the message is endlessly forwarded in a routing loop among nodes 14, 34 and 48.

## IV. Friendly Routing by using goLLuM

In this section we introduce *goLLuM* (*going over Local Links using Dunbar's Method*), a distributed depth-first search in combination with routing preferences, which is used to find existing paths between two nodes efficiently, or to inform the sender if no such path exists. *goLLuM* allows us to forward messages via friends and friends of friends only, which fulfill a certain closeness level. The main goal of this algorithm is to forward messages from any starting point to any given target node (note, that both nodes do not necessarily have to be friends) whenever a friend-to-friend path between those two nodes exists. A message should be sent back to the starting point if no such route exists in order to inform the initial sender about the inaccessibility of the desired receiver.

Normally, the principle of structured overlays is to carry a message closer to a given receiver with every selected hop. If we limit routing by forwarding messages only to friendly nodes with certain trust level, it might happen that the direct path from starting point to target node is blocked due to an incompatible closeness level of the next, closer hop. Our proposed algorithm will detect obstacles in the direct routing path and circumvent them. As can be seen in Figure 2, node $B$ would be best choice of node $A$ for being next hop on the way to target $T$. Nevertheless, $A$ is able to forward the message to $C$ which is second closest node to $T$ in $A$'s routing table and is considered as friend of $A$. $C$ in turn might decide to forward the message to $D$ which has no further friends than $C$. $D$ will thereupon send the message back to $C$ which has enough alternatives to forward the message.

### A. Friendly Routing

In order to realize the distributed search, messages have to be extended by a list of already visited nodes $L_V$ and by a list of nodes $L_P$ which represents the current traversed path from the start point towards the target node. Additionally, a flag within the message indicates whether it is forwarded or sent back as we saw in Figure 2 when $D$ had no further friends than $C$.

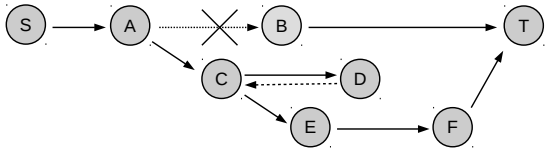



Fig. 2. Basic idea of the routing algorithm is to route around blocked links.

```
1  // Receive LookupMsg which contains:
2  // L_P: list in which previously traversed path is stored
3  // L_V: list in which previously visited nodes are stored
4  // ID_t: given target identifier
5  // back: specifies if LookupMsg has been sent back (true) or forward (false)
6  receipt of LookupMsg(L_P, L_V, ID_t, back) from m at n {
7      if ( n.isResponsibleFor(ID_t) ) {
8          deliver(LookupMsg);
9      } else {
10         // get friends from routing table with level ≤ λ
11         L_F = getFriends(λ);
12         L_F.sort(ID_t);
13         L_V.add(n);
14
15         if ( back == true ) {
16             pos = L_F.getPos(m) + 1;
17         } else {
18             pos = 0;
19         }
20         while ( pos < L_F.size() ) {
21             nextHop = L_F.get(pos);
22             if ( ! L_V.contains(nextHop) ) {
23                 L_P.push(n);
24                 sendto nextHop : LookupMsg(L_P, L_V, ID_t, false);
25                 return; // quit
26             } else {
27                 pos = pos + 1;
28             }
29         }
30         // all nodes have been visited before
31         if ( L_P.size() > 0 ) {
32             nextHop = L_P.pop();
33             sendto nextHop : LookupMsg(L_P, L_V, ID_t, true);
34         } else {
35             stop(); // msg is at initiator again (fail)
36         }
37     }
38 }
```

Fig. 3. goLLuM Routing Algorithm

Figure 3 describes the *goLLuM* routing algorithm in detail. Lookups are started by sending a *LookupMsg* to a friendly node which is closest to a given target identifier. First, upon receiving a *LookupMsg* a node decides if it is responsible for the given target identifier by itself (line 7). If so, the message has reached the desired receiver. Otherwise, the current node is added to the list of visited nodes $L_V$ (line 13). In order to determine the node with the closest identifier to the target, all friends with matching closeness level are stored in a new list $L_F$ (line 11) and sorted according to their distances to the target (line 12 in Figure 3). In the next step, the current node has to decide if the previous node has forwarded the message to the current node through the normal way (line 15: the previous node is closer to the target id and was therefore selected as next hop), or if the previous node has sent the message back because it has no further friends (this means

the current node has forwarded the message previously to a node which seemed to be a good next hop, but has no other friends). In the first case, if a message is received a second time (it has been sent back), the position in list $L_F$ of the last sender (who sent the message back) is determined and increased by one to select a possible next hop out of all friends (line 16). In the second case, the first closest friend is simply selected as possible next hop (line 18). Starting at the calculated position, the list of sorted friends $L_F$ is now iterated to find the next closest node which has not been visited previously (lines 20-22). If such a node is found, it is added to the temporary calculated path represented by list $L_P$ and the message is forwarded to this node (lines 23-25). Only if all friends of one node have been visited before, the message is sent back to the last recently visited node on list $L_P$ (line 33), i.e. the node which previously sent the message to the current node. In addition, the last sender will be removed from list $L_P$ which is used to remember the sequence of all the previous senders of the message (line 32). The depth-first search will be executed until the message reaches its receiver or the initial sender again (line 35).

### B. Privacy

We are aware that both lists, $L_V$ and $L_P$, infringe privacy issues as information about all participants which build the path between sender and receiver are exposed to non-friendly and not-trusted nodes. Therefore we describe next, how to overcome the lack of privacy while the functionality is retained. First, the routing path represented by $L_P$ is not included within the message. Instead, each node caches the last senders of all forwarded messages for a limited amount of time. The cached information is released early if the node sends the message back to the last sender. The list of previously visited nodes $L_V$ is privatized by storing encrypted information about the visited nodes. Each node only needs to know if its own friends appear on the list (line 22 in Figure 3), regardless of whether the information has been put into list $L_V$ by the current node itself or by another participant that shares the same friend. In order to encrypt the nodes stored in $L_V$, each node creates four individual symmetric keys, one for each level of closeness. For each friend in a node's routing table those keys are stored, whose priority is greater or equal to the level of the respective friend. As an example, for level-1-friends keys for level 1, 2, 3, and 4 are stored, whereas for level-3-friends only keys 3 and 4 are stored. Doing this, every node maintains contact information, the associated closeness levels, and the affiliated symmetric keys for each friend. Previously visited nodes, added to list $L_V$, are encrypted by using the key with the lowest priority that is still above the defined maximum level. Therefore, information about visited nodes is only readable to participants that share a key which fulfills the desired closeness level which means those nodes are allowed to be seen.

Using cryptography-based approaches alone for anonymity purpose is expensive and not sufficient as keys for every connection or group-based encryption had to be maintained. Anonymity should be based on trust in networks build on friendship relationships. Encryption alone does not hide sender and receiver nodes from being traced by other nodes, cryptography can only be used to hide content. Connections to friends should rather be used to hide sender and receiver of messages. The problem in existing DHTs is that nodes are exposed to other members in the network, therefore we rely on trusted mechanisms represented by routing via Dunbar's circles.

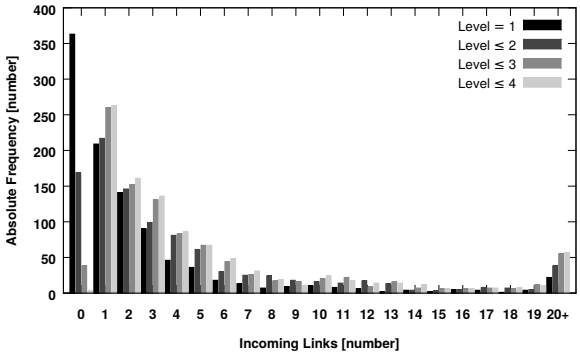
### C. Independence

The *goLLuM* algorithm is absolutely independent from any underlying P2P overlay. As an extension which allows routing over friends only it can be run on every existing approach, since the used distributed depth-first search is designed to find always an existing route between two nodes. If no underlying structure exists, similar to flooding in unstructured overlays, depth-first search will still find an appropriate path to a given target node. As a drawback many nodes have to be visited to find a path. One characteristic of structured overlays is to keep the network scalable by introducing links to far away nodes. Another feature all structured peer-to-peer overlays have in common is the need of a distance metric to describe the difference between participants. In Chord and Pastry for example, distances are derived from positions in an one-dimensional identifier space, which is mapped onto a modulo ring. In Kademlia the XOR operation is used to define distances in the overlay. Furthermore, in every unstructured or structured overlay in which data objects are searched, it has to be clear who is responsible for the data objects. In unstructured overlays like Gnutella[1], participants are simply responsible for content if they hold a given data item. In most structured overlays responsibility for a given data object is distributed equally among all overlay nodes and depends on the position of a node on the identifier space. In Kademlia, lookups are not deterministic, so that many nodes can be responsible for a desired data item. Our routing approach benefits from a distance metric to order links and selects preferred links first. Furthermore, it has to be known where to stop a lookup. Although we build the algorithm over Pastry for evaluation purpose, it can be applied to any other structured or unstructured overlay in principle, which only has to define an interface for the used distance metric and an interface for announcing responsibility for a given data item.

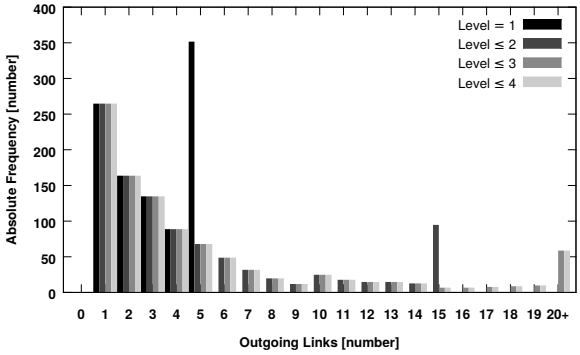
## V. Simulation Methodology

One central part of our evaluation is the assignment of friendship relationships to every node we simulate. In this section we explain how friend contacts are distributed among overlay nodes during simulation and we investigate properties of the modeled friendship topologies we use. Besides theoretical models for relationships in social networks, namely those proposed by Barabási and Albert in [3], or Watts and Strogatz [29], we consider relationships collected from real-world measurements presented by Wilson et al. in [30].

[1] http://rfc-gnutella.sourceforge.net/developer/stable/index.html




(a) Histogram of incoming links




(b) Histogram of outgoing links

Fig. 4. Distribution of incoming and outgoing links for the *EuroSys '09* dataset model

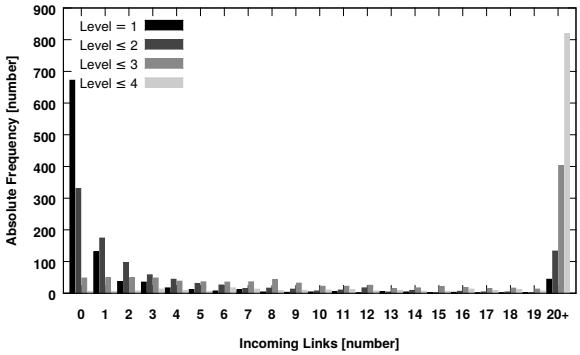



Fig. 5. Distribution of incoming links for the Barabási-Albert model

| **Barabási-Albert Model** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *Level = 1* | | *Level ≤ 2* | | *Level ≤ 3* | | *Level ≤ 4* | |
| | Out | | Out | | Out | | Out | |
| 0 | 0 | | 0 | | 0 | | 0 | |
| 5 | 983 | | 5 | | 5 | | 5 | |
| 15 | 0 | | 938 | | 6 | | 6 | |
| 50 | 0 | | 0 | | 788 | | 3 | |
| 150 | 0 | | 0 | | 0 | | 357 | |
| Others | 17 | | 57 | | 201 | | 629 | |
| **Small World Graph** | | | | | | | | |
| | *Level = 1* | | *Level ≤ 2* | | *Level ≤ 3* | | *Level ≤ 4* | |
| | In | Out | In | Out | In | Out | In | Out |
| 0 | 71 | 0 | 61 | 0 | 26 | 0 | 0 | 0 |
| 5 | 919 | 1000 | 1 | 0 | 1 | 0 | 0 | 0 |
| 15 | 0 | 0 | 909 | 1000 | 1 | 0 | 0 | 0 |
| 50 | 0 | 0 | 0 | 0 | 874 | 1000 | 0 | 0 |
| 150 | 0 | 0 | 0 | 0 | 0 | 0 | 1000 | 1000 |
| Others | 10 | 0 | 29 | 0 | 98 | 0 | 0 | 0 |

TABLE I
LINK DISTRIBUTION FOR THE BARABÁSI-ALBERT MODEL AND THE SMALL WORLD GRAPH

### A. Friendship Assignment in our Simulations

As first step of our simulations, we generate graphs of 1000 nodes which represent the relationships between all participating nodes in the network. We decided to consider scale-free graphs which follow a power-law distribution proposed by Barabási and Albert, Watts' and Strogatz' Small World graphs, and a realistic dataset which describes friendship relationships obtained from Facebook (see Section V-B). Since our goal is to point out the drawbacks of a naive integration of social links into existing P2P overlays it is acceptable to simulate a small number of nodes only. In the first phase of the simulations, the used overlay (we choose Pastry) is established and stabilized. In the second phase, one of the previously created topologies is used to extend the overlay by an additional routing table, namely the friend-routing table. Each node in the friendship topology represents an overlay node during simulation and for each overlay node, all outgoing links of the friendship topology node are considered as friends. According to their level of closeness the additional nodes (friends) are stored inside the friend-routing table in friend-buckets. For each level of closeness one bucket exists. Starting with the first, the buckets are sequentially filled with nodes such that the first bucket contains up to 5 friends. The second bucket has not more that 10 nodes, up to 35 nodes are maintained in bucket 3, 100 nodes are maximal stored in bucket 4. All remaining nodes are added to the friend-routing table as level 5 contacts. Using the Facebook data set, friends are assigned to the different buckets according to their closeness which is given in the data set itself. Simulating the Barabási and Albert model or Watts' and Strogatz' Small World graphs closeness levels are assigned randomly to neighbors in the corresponding graph.

By applying the Dunbar scheme to the routing tables of each node, a directed friend graph is obtained. Undirected friendship graphs are not compatible with the Dunbar scheme. Since the different level buckets are limited in their sizes, and all nodes have different friends, it is not possible to assign bi-directionality to all links in the friendship network. For example, consider nodes that only have one friend in their routing table. If now more than 5 of these nodes maintain level-1 closeness links to a specific node $n$, this node $n$ will not be able to hold all of this nodes in its own level-1 bucket at the same time. Therefore, friendship might not exist mutually inside the friend graph which is induced by the Dunbar scheme. We assume in our simulations that each node has at least one friend he trusts in. The resulting graph might be weakly connected only. This is the case if some nodes have only outgoing links but no incoming links, which means that no other node considers them trustful. In reality this might be the case if those nodes only request data from others but never offer data by their own.

To get a better feeling for the characteristics of the simulated graphs, in Figures 4, 5, and in Table I it is shown, how incoming and outgoing links are distributed over the nodes in the different topology types with respect to the considered friend levels. While the x-axis presents the number of incoming or outgoing links, the y-axis presents the absolute number of nodes which hold the given number of links respectively. For the real-world data it can be seen in Figure 4(a) that more than 350 out of 1000 nodes have no incoming links. Furthermore, Figure 4(b) reveals that each node has at least one outgoing link. In Figure 5 the power-law distribution of incoming links in the Barabási topology is shown. The distribution of outgoing links for the Barabási model as well as the link distribution for Watts' and Strogatz Small World model is presented in table I. The distribution of nodes for both models is presented for each closeness level and for selected numbers of incoming and outgoing links. All nodes which do not have 0, 5, 15, 50, or 150 links are listed as *Others*.

### B. EuroSys '09 Dataset

To evaluate our approach we have used a Facebook dataset which is publicly available for research[2] referred as *Anonymous regional network A*. The data set consists of a social graph which describes the overall structure of the downloaded network and it consists of more than 3 million nodes and more than 23 million edges (social links). Furthermore, the dataset contains also four "interaction graphs". More details about the dataset are shown in [1]. We have used a small connected component of 1000 nodes obtained from the Interaction graph. These graphs only contain simple edges connecting anonymized node identifiers.

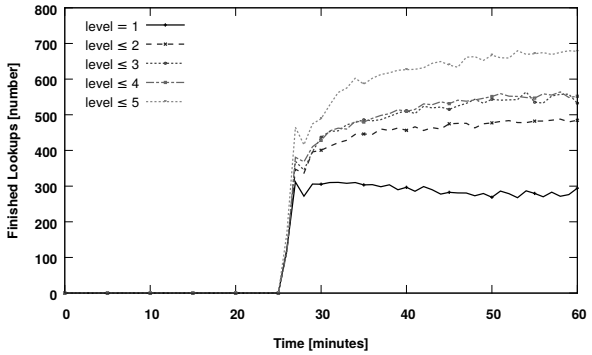
## VI. Evaluation

To obtain realistic results and insights into the social DHT we use the event-based simulator PeerfactSim.KOM [16] for our simulations because it contains already implementations of Pastry [25] and further structured overlays. Each simulation uses GNP coordinates [22] to estimate delays in the fundamental network. Furthermore, measurements from the PingEr project [20] are integrated into the simulator for reasonable approximations of jitter and packet loss. Each of our simulation has been run with 10 different random seeds, therefore the values we obtained represent the average of 10 different values.

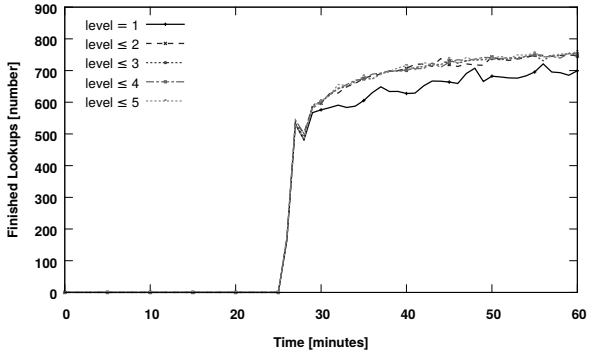
### A. Simulation Setup

Within our simulations we selected Pastry as underlying (representative) DHT and compare SocialPastry (proposed in Section III) to the *goLLuM* routing algorithm introduced in Section IV. In our setup, approximately 800 lookups to random targets are started per simulation minute. In addition, lookups are only forwarded to nodes which satisfy condition $L_c \leq \lambda$, where $\lambda$ is a given maximum level. We test these implementations with 1000 nodes per simulation and three different friendship relationship models, namely we compare

[2]http://current.cs.ucsb.edu/facebook/




(a) Social Pastry



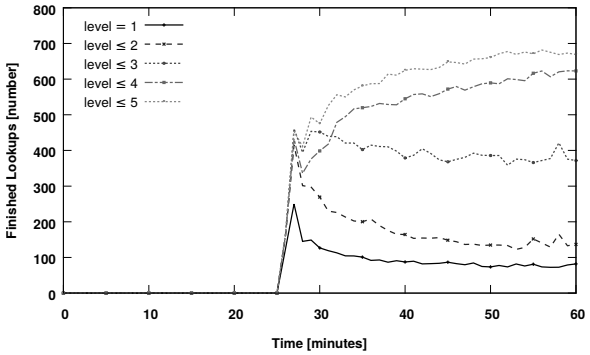

(b) goLLuM Algorithm

Fig. 6. EuroSys '09 Dataset: Finished Lookups

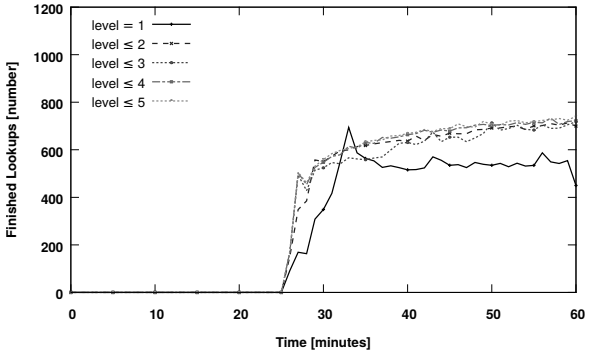
friend topologies according to Barabási and Albert [3], a Small World graph introduced by Watts and Strogatz [29], and a topology observed in real friendship relations by Wilson et al. in [30]. We want to investigate primarily, the number of successful lookups to prove functionality of our routing algorithm. Secondly, we identify the average number of hops which have to be traversed to reach a given target identifier. A small number of nodes (1000) is enough to prove the functionality of the *goLLuM* routing algorithm on the one side and to show that social routing over certain friends in naive approaches such as SocialPastry suffer from routing problems on the other side. The high number of failing lookups indicate that a simple integration of social links into Pastry does not lead to a reliable and proper social DHT.

### B. Simulation Results

In the following we focus on routing performance in Pastry in terms of successful lookups per closeness level and number of hops each lookup request has to traverse. We want to investigate both presented metrics here as they give us good insights on basic functionality of SocialPastry and *goLLuM*, further metrics like bandwidth consumption or network traffic will not be discussed here since they only reveal additional information concerning performance. In our simulations we observe the following results: in Figures 6, 7, 8 we present the number of successful lookups per maximum closeness level for the real-world *EuroSys* model, the Barabási-Albert model, and a Small World graph respectively. As expected, it can be seen in Figures 6(a), 7(a), and 8(a) that the more links each node

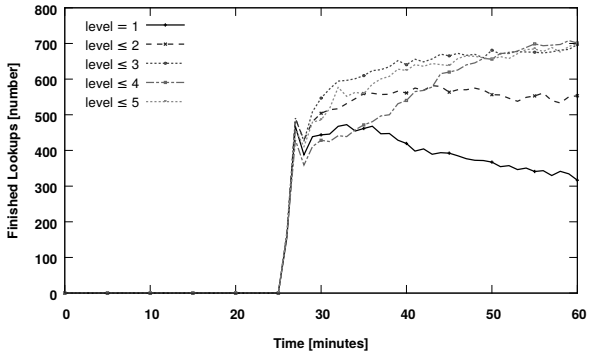



(a) SocialPastry

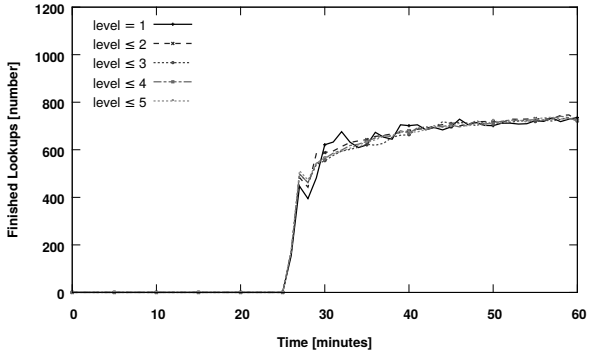



(b) goLLuM Algorithm

Fig. 7. Barabási-Albert Model: Finished Lookups




(a) SocialPastry




(b) goLLuM Algorithm

Fig. 8. Small World Graph: Finished Lookups

uses for routing decisions the more lookups are successful in SocialPastry. The reason for this outcome is easily explained: it is not enough to forward messages to closest friend contacts only, as it is done in SocialPastry. Additional rules have to be defined which control the forwarding mechanism in situations in which no friend fits as next hop or situations in which routing loops might occur. Considering Figures 6(b), 7(b), and 8(b), on the contrary, reveals that using the *goLLuM* algorithm, lookups are performed similarly regardless of which maximum closeness level is chosen. The advantage of *goLLuM* to naive approaches is that already traversed paths are detected and circumvented. If we consider Level-1-friends for routing only, it can be seen in Figure 7(b) that even using depth-first search does not guarantee that every node in a social, Dunbar-based network is found. Still some Lookups might be not successful, especially if the node responsible for a given identifier has no incoming links. In this case there is no participant in the network that considers the mentioned node as friend, those lookups can never be successful. For each simulated friendship model, for both routing approaches, and for each selected level of closeness, Table II presents the number of average hops which have to be traversed to reach a given destination node successfully. Hop counts obtained from failed lookups attempts are not listed in the table. The actual hop count is presented in row *Hops*, whereas $+\sigma$ and $-\sigma$ denote the average standard deviation in positive and negative direction.

During simulations, we observed that by using the *goLLuM* routing algorithm, lookups fail only if no route between the starting point of a lookup and the target node exists. In this case, the lookup message is sent back to the initiator after some time. Using SocialPastry, we observed further that lookups fail only due to routing loops in the friend topology In our simulations we aborted lookup attempts after 50 hops.

With our simulations, we have shown that routing via friends and friend-of-friends is possible if a path between sender and receiver of a message exists. Further it has been shown that the number of visited nodes is reasonable.

## VII. Conclusion

In this work we address the issue of finding trustful routes between sender and receiver in a Dunbar-based social overlay. First, we propose a P2P system, in which friendship relationships are established according to Dunbar's concept. A naive social DHT is built on Pastry (SocialPastry) as first approach to understand the problem of trustful routing in social DHTs. The goal is to route messages via trusted friends and friends of friends where the privacy level can be selected dynamically. Secondly, we describe the *goLLuM* routing algorithm which ensures to find a trusted route between two arbitrary nodes if one exists. For doing this, we use a modified and distributed depth-first search in which links with a short distance to the given target node are preferably investigated. The advantage of our approach is its independence from the underlying overlay, in principle it can be applied to any P2P systems. By using different friendship models for social networks in our simulations, we compare SocialPastry with the proposed routing algorithm. Evaluation reveals that using the distributed

depth-first search it is possible to route via trusted friends. Both approaches, F2F-routing and DHT routing come with dedicated benefits such as trust or privacy on the one side and general applicability on the other side. By combining these two worlds, we enhance reliable DHT routing with the help of trusted nodes.

| **EuroSys '09 Dataset** | | | | | |
|---|---|---|---|---|---|
| | Level = 1 | Level ≤ 2 | Level ≤ 3 | Level ≤ 4 | Level ≤ 5 |
| *SocialPastry* | | | | | |
| $+\sigma$ | 7.47 | 8.88 | 8.92 | 9.04 | 12.77 |
| Hops | 2.22 | 2.24 | 2.26 | 2.28 | 3.59 |
| $-\sigma$ | 0.98 | 0.90 | 0.89 | 0.91 | 2.29 |
| *goLLuM Algorithm* | | | | | |
| $+\sigma$ | 16.60 | 14.00 | 11.30 | 9.21 | 9.41 |
| Hops | 12.28 | 7.06 | 4.22 | 3.52 | 3.63 |
| $-\sigma$ | 8.66 | 4.82 | 2.44 | 1.88 | 1.96 |
| **Barabási-Albert Model** | | | | | |
| | Level = 1 | Level ≤ 2 | Level ≤ 3 | Level ≤ 4 | Level ≤ 5 |
| *SocialPastry* | | | | | |
| $+\sigma$ | 1.27 | 2.35 | 5.51 | 5.39 | 11.89 |
| Hops | 1.26 | 1.70 | 2.39 | 2.09 | 3.17 |
| $-\sigma$ | 0.29 | 0.70 | 1.12 | 0.93 | 1.93 |
| *goLLuM Algorithm* | | | | | |
| $+\sigma$ | 2.25 | 3.65 | 14.73 | 2.36 | 1.40 |
| Hops | 2.65 | 2.99 | 5.44 | 1.07 | 1.04 |
| $-\sigma$ | 1.31 | 1.70 | 4.16 | 0.09 | 0.06 |
| **Small World Graph** | | | | | |
| | Level = 1 | Level ≤ 2 | Level ≤ 3 | Level ≤ 4 | Level ≤ 5 |
| *SocialPastry* | | | | | |
| $+\sigma$ | 3.18 | 3.05 | 2.80 | 3.87 | 12.03 |
| Hops | 3.26 | 2.83 | 2.36 | 1.80 | 3.33 |
| $-\sigma$ | 1.72 | 1.50 | 1.13 | 0.76 | 2.08 |
| *goLLuM Algorithm* | | | | | |
| $+\sigma$ | 5.69 | 3.80 | 9.23 | 11.49 | 11.13 |
| Hops | 3.87 | 1.94 | 1.63 | 1.05 | 1.06 |
| $-\sigma$ | 2.24 | 0.90 | 0.65 | 0.07 | 0.07 |

TABLE II
AVERAGE HOP COUNTS FOR DIFFERENT ROUTING METHODS, FRIENDSHIP MODELS AND LEVELS OF TRUST

## REFERENCES


[1] V. Arnaboldi, M. Conti, A. Passarella, and F. Pezzoni, "Analysis of Ego Network Structure in Online Social Networks," in *International Conference on Privacy, Security, Risk and Trust (PASSAT), and International Conference on Social Computing (SocialCom)*. IEEE, 2012, pp. 31–40.

[2] V. Arnaboldi, A. Guazzini, and A. Passarella, "Egocentric Online Social Networks: Analysis of Key Features and Prediction of Tie Strength in Facebook," *Computer Communications*, vol. 36, no. 10-11, pp. 1130–1144, 2013.

[3] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[4] Bricklin, Dan, "Friend-to-Friend Networks," http://www.bricklin.com/f2f.htm, 2000.

[5] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta, "PeerSoN: P2P Social Networking: Early Experiences and Insights," in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems (SNS)*. ACM, 2009, pp. 46–52.

[6] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel, "Private communication through a network of trusted connections: The dark freenet," *Network*, 2010.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 46–66.

[8] M. Conti, A. De Salve, B. Guidi, F. Pitto, and L. Ricci, "Trusted Dynamic Storage for Dunbar-Based P2P Online Social Networks," in *On the Move to Meaningful Internet Systems (OTM)*. Springer, 2014, pp. 400–417.

[9] M. Conti, A. De Salve, B. Guidi, and L. Ricci, "Epidemic Diffusion of Social Updates in Dunbar-Based DOSN," in *Euro-Par 2014: Parallel Processing Workshops*. Springer International Publishing, 2014, pp. 311–322.

[10] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: a Privacy Preserving Online Social Network Leveraging on Real-Life Trust," *Communication Magazine, IEEE*, vol. 47, no. 12, 2009.

[11] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca, "Decentralized Online Social Networks," in *Handbook of Social Network Technologies and Applications*, 2010, pp. 349–378.

[12] R. I. M. Dunbar, "The Social Brain Hypothesis," *Evolutionary Anthropology: Issues, News, and Reviews*, vol. 6, no. 5, pp. 178–190, 1998.

[13] M. Everett and S. P. Borgatti, "Ego Network Betweenness," *Social Networks*, vol. 27, no. 1, pp. 31–38, 2005.

[14] B. Gonalves, N. Perra, and A. Vespignani, "Modeling Users' Activity on Twitter Networks: Validation of Dunbar's Number," *PLoS ONE*, vol. 6, no. 8, 2011.

[15] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz, "LifeSocial.KOM: a Secure and P2P-based Solution for Online Social Networks," in *Consumer Communications and Networking Conference (CCNC)*. IEEE, 2011, pp. 554–558.

[16] K. Graffi, "PeerfactSim.KOM: A P2P System Simulator Experiences and Lessons Learned," in *Proceedings of the International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2011.

[17] B. Guidi, "P2P Architectures For Distributed Online Social Networks," in *International Conference on High Performance Computing and Simulation (HPCS)*. IEEE, 2013, pp. 678–681.

[18] B. Guidi, T. Amft, A. De Salve, K. Graffi, and L. Ricci, "Didusonet: A p2p architecture for distributed dunbar-based social networks," *Peer-to-Peer Networking and Applications*, pp. 1–18, 2015.

[19] S. Marti, P. Ganesan, and H. Garcia-Molina, "Sprout: P2p routing with social networks," in *Current Trends in Database Technology-EDBT 2004 Workshops*. Springer, 2005, pp. 425–435.

[20] W. Matthews and L. Cottrell, "The PingER Project: Active Internet Performance Monitoring For the HENP Community," *Communications Magazine*, vol. 38, no. 5, pp. 130–136, 2000.

[21] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Proceedings of the International Workshop on Peer-To-Peer Systems (IPTPS)*, ser. LNCS, vol. 2429. Springer, 2002.

[22] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-based Approaches," in *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2002, pp. 170–179.

[23] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia, "Cachet: a Decentralized Architecture for Privacy Preserving Social Networking with Caching," in *Proceedings of the International Conference on Emerging Networking Experiments and Technologies*, 2012, pp. 337–348.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, vol. 31. ACM, 2001.

[25] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proceedings of the International Conference on Distributed Systems Platforms (IFIP/ACM Middleware)*. Springer, 2001.

[26] R. Sharma and A. Datta, "SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks," *CoRR*, vol. abs/1105.0074, 2011.

[27] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. ACM, 2001.

[28] A. Sutcliffe, R. Dunbar, J. Binder, and H. Arrow, "Relationships and the Social Brain: Integrating Psychological and Evolutionary Perspectives," *British Journal of Psychology*, vol. 103, no. 2, pp. 149–168, 2012.

[29] D. J. Watts and S. H. Strogatz, "Collective Dynamics of Small-World Networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[30] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User Interactions in Social Networks and Their Implications," in *Proceedings of the 4th ACM European Conference on Computer systems*. ACM, 2009, pp. 205–218.