A Socio-Aware Decentralized Topology Construction Protocol

Stefanos Antaris*, Despina Stasi*, Mikael Högqvist†, George Pallis*, Marios Dikaiakos*
*Department of Computer Science, University of Cyprus †Hive Streaming AB
{antaris.stefanos, despina.stasi, gpallis, mdd}@cs.ucy.ac.cy, mikael@hivestreaming.com

***Abstract*—Decentralized Online Social Networks (DOSNs) offer an alternative to the usual centralized solutions, that promises to preserve more control of a user's data to the user herself. An important aspect in the design of DOSNs is the selection of the Peer-to-Peer overlay network. We propose the use of an augmented Pastry-based overlay network design, so as to produce a topology that incorporates the social network. Our augmentation aims to reduce the number of overlay hops required for the communication between two users in the social network. We experimentally show that our approach reduces the number of overlay hops and the total latency between two socially connected users versus the standard Pastry overlay. Additionally, we compare our approach with Sprout, a Chord alternative, and show that in many cases we achieve similarly low latency, while maintaining a bounded number of links.**

Keywords*—Decentralized Social networks; DOSN; P2P; Overlay Network;

I. Introduction

Online Social Networks (OSNs) evolved into ubiquitous platforms of communication over the past few years and revolutionized interpersonal communication between people. Social users produce and disseminate an unprecedented amount of textual and multimedia content that is stored in a central data storage. This central data storage is controlled by the OSN providers and come with a variety of serious concerns, which are amplified by the very personal nature of social-networking-related data.

These concerns attracted the attention of the research community to decentralized OSNs (DOSNs). DOSN architectures are promising as they inherently offer better privacy and less dependence on a single service provider. However, these architectures bring new challenges regarding core features of OSNs such as data access and dissemination.

Recently, researchers proposed a wide range of DOSN solutions [1]–[3] using P2P overlay networks. These solutions enable data to be distributed in the social users' machines and data retrieval is accomplished using the P2P lookup service. However, current DOSN approaches do not integrate the social dynamics of the OSNs in the design of the P2P topology construction protocol. Specifically, the P2P overlay networks are designed to be generic in the sense that they provide low-level communication abstractions of the Physical IP Network, and are maintained based on the adjustments of the Physical IP Network layer. When a user enters the DOSN, her peer in the P2P overlay network is assigned a random identifier, creating thus a socially unaware identifier space. The routing, search and key-value functionality of the DOSN relies on the complexity that the overlay network provides.

Motivated by the absence of a decentralized topology construction protocol suitable for DOSNs, we propose a new algorithm that embeds the social links into the P2P topology construction protocol. Based on this algorithm, the interconnections between peers in the P2P overlay network are not designed to be generic. The peers provide connections between socially connected users, taking into account their network latency property. To sum up, we make the following contributions:

- We introduce an efficient and scalable P2P topology construction protocol for a DOSN. We propose an algorithm that incorporates the structural properties of the underlying social network.
- We demonstrate the feasibility of the proposed approach through the NewsFeed functionality by propagating the user's post to each of her social contacts. We employ an implementation of Pastry [4], an overlay network widely used by the literature for the design of DOSNs [1], [2], [5].
- We experimentally show on four real-world data sets that our approach provides superior performance compared to the basic Pastry approach.

The remainder of this paper is structured as follows. In Section II we conduct a review of the related work in the course of DOSNs. In Section III we provide a comprehensive analysis of the background in the structured P2P topology construction protocol and the formulation of the problem that current structured P2P overlay networks pose in a DOSN. The design of our proposed P2P topology construction protocol is presented in Section IV. We present an extensive evaluation of our proposed approach against the state-of-the-art approaches in Section V and conclude in Section VI.

II. Related Work

In the course of DOSNs, several approaches such as LotusNet [6], Safebook [7], Cachet [1] and SOUP [2] have been proposed. In LotusNet [6] the authors propose a two-tier architecture in which the first tier is a structured P2P overlay network and the second tier consists of social users and the connections they form. The authors tackle the problem of security and privacy in a DOSN. However, mechanisms that leverage the social graph in the construction of the P2P overlay network are not provided and information propagation presents long delays due to the complexity of the P2P substrate. Safebook [7] is based on a P2P overlay network named "Matryoska" in order to preserve privacy on the users data by leveraging the hop-by-hop trust. Although the authors retain privacy in the DOSN communication, they provide additional complexity to the information dissemination. The hop-by-hop communication in Matryoska is accomplished between peers that are several hops away in the P2P overlay network. The complexity that the P2P overlay network provides to the information propagation in a DOSN is presented in Cachet [1]. The authors in Cachet propose a hybrid structured and unstructured P2P overlay network in order to store the data to peers that are close to the social users. However, the gossip-based algorithm applied provides an additional overhead to the peers. SOUP [2] advocates an approach to store the users' data to their social friends, by obtaining the social friends that are the best mirror candidates. Although they achieve high data availability, they do not consider the number of messages required to store and retrieve the data to peers that are several hops away, and thus, provide high latency as the network grows.

The closest work to our approach is SPROUT [8], where links are added to the Chord [9] overlay network by establishing additional connections for each peer to each of her social friends. This approach enables the P2P overlay substrate to contain direct connections between friends, thus producing one-hop storage/retrieval of the data between those individuals. However this requires the peers in the overlay substrate to maintain an unlimited number of TCP connections which results in excessive communication overhead.

III. Background and Problem Statement

In this paper, we place specific emphasis on the Pastry [4] overlay network, since this is the overlay network found most in the literature for the design of a DOSN, e.g. in [1], [2], [5].

A. Pastry

Pastry [4] is a generic, efficient and scalable substrate for P2P applications. Each peer in Pastry is assigned a random 128-bit identifier D (ranges from 0 to $2^{128}-1$) that results in the uniform distribution of the peers in the NodeID space S. The NodeIDs are generated using a hash function and are represented as a sequence of $n \geq 1$ digits with base B (where $B = 2^b$ is a configuration parameter with typical value of $b = 4$).

For the lookup process, each peer p_i maintains a routing table T_i with $log_B N$ rows and $B-1$ columns. The k^{th} row of the routing table contains peers whose NodeID matches the $k-1$ first digits of the current peer's NodeID. In addition, the k^{th} digit of each NodeID in this row attains one of the $B-1$ possible values that differ from the k^{th} digit in the current peer's NodeID, and correspond to each of the cells in the row.

Given a query with a key q, the peer p_i answers the query if its NodeID D_i is the numerically closest NodeID to the given key q. Otherwise, the peer p_i routes the query q to the peer $p_j \in T_i$ whose NodeID D_j has the longest prefix matching with the given key q. This lookup process is performed until the peer p_q, that is the numerically closest to the given key q, receives the query. Each hop in the lookup process exponentially decreases the NodeID space required to reach the destination peer p_q. Therefore, the lookup process is accomplished in a bounded number of overlay hops l, where approximately $l = \lceil log_B N \rceil$, unless 2^{b-1} adjacent peers fail simultaneously.

In order to improve routing performance, Pastry employs a network proximity metric for the maintenance of peers' routing table. A peer which presents lower proximity value is assumed to be more desirable and its NodeID is assigned to the appropriate cell in the routing table. By applying this metric, the total network latency for the lookup process is reduced.

B. Problem Definition

Consider a social graph $G = (V, E)$, where V is the set of N users ($N = |V|$) and E is the set of edges in the social graph. An edge $(u_i, u_j) \in E$ exists if and only if $u_i \in V$ and $u_j \in V$ are two users connected in the social network. The set of all connections of node u_i, defined as $C_i = \{u_j \in V : (u_i, u_j) \in E\}$, is called u_i's *social neighborhood.*

In a DOSN, each social user $u_i \in V$ participates as a peer node p_i in an overlay network $G' = (P, F)$, where P is the set of peers in the overlay and F is the set of direct connections that each peer maintains in its routing table T_i, i.e. $(p_i, p_j) \in F$ if p_j is an entry in p_i's routing table T_i. We assume that each user $u \in V$ is mapped onto only one peer, and thus $|P| = |V| = N$.

When a social user $u_i \in V$ joins the DOSN, a new peer $p_i \in P$ is generated into the overlay network along with its routing table T_i. The peer p_i is assigned a random NodeID D_i, using a hash function, that indicates its position in the NodeID space S. Therefore, both peers and social users are uniformly distributed in the NodeID space S. The NodeID D consists of all nonnegative integers with $n \geq 1$ digits in base $b \geq 2$. In the remainder of the paper, when we refer to a peer p_i we consider the peer that hosts the social user u_i.

In current DOSN approaches, P2P overlay networks are exploited without taking into consideration the social network. The probability in Pastry P2P overlay network that the social connections E overlap with the overlay connections F is extremely small. In particular, the expected number of peers that are eligible to be inserted into a specific cell in the k^{th} row of the routing table T_i of the peer p_i is approximately N/B^{k+1}, where $1 \leq k \leq log_B N$. On the other hand, the expected number of socially connected peers that are eligible to be inserted into the same cell of T_i is approximately $|C_i|/B^{k+1}$. This means that the probability that a direct connection in the overlay network connects two peers who are friends in the social network is $|C_i|/N$, a quantity that tends to 0 as the network size grows faster than the the cardinality of C_i.

Thus, in a socially unaware Pastry overlay network, the number of hops required for a social user p_i to send a message to his social friend p_j is, on average, $O(\log N)$. We denote by $p_i \rightarrow^+ p_j$ the path that connects peer p_i to peer p_j. The length of that path is $l_{ij} = |p_i \rightarrow^+ p_j|$. When a social user u_i publishes a post, the system needs to inform all of his friends, i.e. the entire neighborhood set C_i. The total number of messages required to propagate a message to u_i's neighborhood set C_i is:

$$|p_i \rightarrow^+ C_i| = \sum_{p_j \in C_i} l_{ij} \tag{1}$$

Each overlay hop provides an additional delay to the propagation of the message, which affects the network *latency* $\tau(p_i, p_j)$ of the routing process from p_i to p_j. We measure the total network latency required for a user u_i to propagate an update to each of the nodes corresponding to users in the neighborhood set C_i as:

$$|p_i {\rightarrow^+} C_i|_\tau = \sum_{p_j \in C_i} \tau(p_i, p_j) \tag{2}$$

An overlay network that reduces the number of overlay *hops* and the total network *latency* for the communication between social friends is required. To this end, the peers in the overlay network should establish a carefully selected number of direct connections in order to reduce the communication overhead. Therefore, each peer p_i should maintain direct connections in its routing table T that minimize the total number of hops. We denote by $\rho(p_i, C_i)$ the minimum number of hops required for a social user u_i to inform all of his friends in C_i, as follows:

$$\rho(p_i, p_j) = \min_T |p_i {\rightarrow^+} p_j| \tag{3}$$

Finally, the selection of the direct connections that each peer establishes should preserve the minimum network latency, in terms of time, required at each overlay hop l to propagate a message to his social friend, as follows:

$$\tau(p_i, C_i) = \min_T |p_i {\rightarrow^+} C_i|_\tau \tag{4}$$

IV. Our Approach: augmenting the routing table

In our approach, we assume that a social user communicates mostly with his social friends and communication between non-socially connected users is rare. Therefore, we propose an algorithm that augments the routing table of Pastry P2P overlay network with social links. The main objective of this algorithm is to establish direct connections between as many pairs of social friends as possible. Simultaneously we aim to achieve $\log N$-overlay hop communication between the remaining pairs of social friends, as well as between pairs of social non-friends. Based on the routing table refinement produced by Algorithm 1, the number of overlay hops $|p_i {\rightarrow^+} p_j|$ required for a user u_i to propagate a post to each of his friends $p_j \in C_i$ is reduced to 1 hop for the majority of p_j.

A. Peer State

Each peer maintains a social neighborhood set C and a routing table T in an encrypted manner to prevent an attacker learn social contact information. Each routing table T contains $\lceil log_B N \rceil \times (B-1)$ entries.

The social neighborhood set C contains the NodeIDs of his social friends. The neighborhood set is used to augment the routing table entries with direct connections to the social user u_i's social friends, and maintained to reflect changes in the social graph.

B. Routing Table Refinement Algorithm

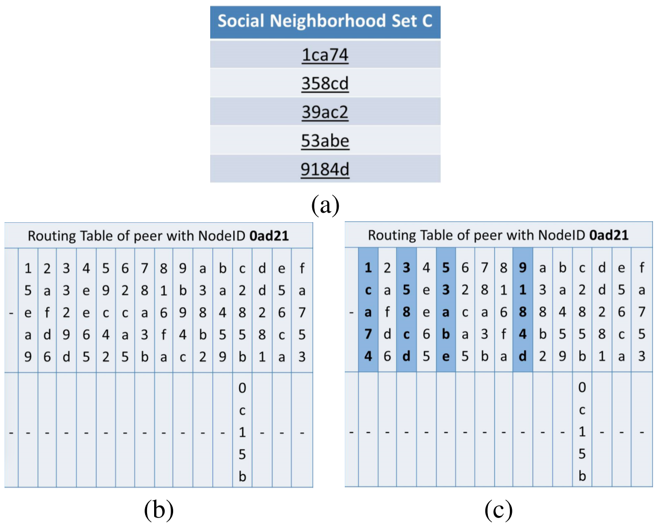
Having initialized the state tables of the peers, each peer p_i replaces the existing direct connections in T_i with a direct connection between two peers that are neighbors in the social graph. Since a bounded number of direct connections ($\lceil \log N \rceil \times (2^B - 1)$) are maintained in the routing table T, more than one friends of u_i will fall in the same cell of the routing table. In our algorithm, we consider two strategies for the selection of the friend who will replace the current entry in $T_i[\text{row}, \text{col}]$: uniformly at random, or based on proximity. In the former, each friend that fits in the cell has an equal probability of being selected. In the latter we pick the friend with the lowest network latency. In both ways, we establish connections with the same number of social friends. However, in the proximity-based solution we also ensure that we keep the network latency low.

The outline of our proposed Routing Table Refinement Algorithm that each peer p_i runs independently is presented in Algorithm 1. Recall that the peer has only local knowledge of both the social network and the overlay network, in the user's neighborhood set C_i and the peer's routing table T_i, respectively.

Algorithm 1: Routing Table Refinement Algorithm

```
Input:
$C_i$: the set of neighbors of social user $u_i$ in $G$.
$T_i$: the routing table of the peer.
choiceOfFriendCriterion: flag indicating the method of
choosing a friend to be proximity-based or uniformly at
random.
Output:
${T_i}'$: the augmented table of the peer.
1  $T_i' = T_i$;
2  Shuffle $C_i$;
3  for $u_j \in C_i$ do
4      if isOnline($u_j$) then
5          (row, col) = findRoutingTableCell($p_i, p_j$);
6          case choiceOfFriendCriterion==“random”
7              ${T_i}'$(row, col) = $p_j$.NodeID;
8          end
9          case choiceOfFriendCriterion==“proximity”
10             if getProximity($p_j$.NodeID) <
               getProximity($T_i$(row, col)) then
11                 ${T_i}'$(row, col) = $p_j$.NodeID;
12             end
13         end
14     end
15 end
16 return ${T_i}'$;
```

We begin by shuffling the social neighborhood set C_i (step 2) in order for each entry of the C_i to be inserted into the routing table T_i, with equal probability, when the random strategy is applied. Thereafter, in steps 3-15, we distribute the neighbors of the user $u_i \in V$ that are online, into the routing table T_i of u_i's peer $p_i \in P$. To do this, for each social friend u_j of u_i, we determine the appropriate cell in the routing table T_i, where p_j can be inserted according to its NodeID as follows. As Pastry's routing table is designed based on Plaxton's prefix-based routing algorithm [10], we identify the number of common digits in the prefix between the NodeID of p_i and p_j: this determines the row of the routing table. The column is determined by the first digit in which the two node identifiers differ. When the proximity strategy is applied (steps 9-13), we compare the network latency, in terms of time, between the two candidate social friends and select the one



Social Neighborhood Set C
1ca74
358cd
39ac2
53abe
9184d

(a)

Routing Table of peer with NodeID **0ad21**

-	15ea9	2afd6	3329d	4ee65	59c42	62ca5	78a3b	816fa	9b94c	a38b2	ba459	c285b	dd281	e56ca	fa753
-	-	-	-	-	-	-	-	-	-	-	-	0c15b	-	-	-

(b)

Routing Table of peer with NodeID **0ad21**

-	**1ca74**	2afd6	**358cd**	4ee65	**53abe**	62ca5	78a3b	816fa	**9184d**	a38b2	ba459	c285b	dd281	e56ca	fa753
-	-	-	-	-	-	-	-	-	-	-	-	0c15b	-	-	-

(c)

Fig. 1. The Neighborhood set C of the peer with NodeID $0ad21$ (a) and the Routing Table (b) before and (c) after our algorithm is applied.

with the minimum latency.

As shown in Figure 1b, peer $0ad21$ maintains direct connections that differ from the NodeIDs included in the neighborhood set C (Figure 1a). Pastry establishes direct connections with peers at each partition of the NodeID space S in order to exponentially decrease the distance in the NodeID space at each hop. Replacing a direct connection of one partition with a peer of the same partition that hosts a social user's friend will not affect the routing process of Pastry. In Figure 1c, the output of the algorithm is presented, where the NodeIDs of the social user's friends are included into the routing table T.

C. Peer Arrival

When a social user joins the DOSN, a new peer p_i is created and a random NodeID is assigned using the SHA-1 128-bit hash function. The peer p_i initializes its routing table T_i following the peer arrival process of Pastry. Therefore, the peer p_i contains a routing table T_i with $\lceil log_B N \rceil \times (B-1)$ direct connections to other peers in the overlay network.

We assume that a social user u_i joins the DOSN by accepting an invitation received by an already existing social user u_j. As the social users u_i and u_j are socially connected, peers p_i and p_j include in their respective neighborhood sets C_i and C_j each others' NodeID, and each proceeds to call Algorithm 1 to update their routing tables.

D. Peer Departure

To maintain a routing table with direct connections that are online, our algorithm periodically requests each peer of the routing table for its state. When a peer is unresponsive, we replace the unresponsive peer with another peer from the neighborhood set C following Algorithm 1. If no NodeID in C is suitable to replace the unresponsive connection, the peer evokes the Pastry routing table maintenance process to create a new direct connection with another peer. Using this approach, our method maintains direct connections with many of the social user u_i's friends, and preserves the $O(\log N)$ overlay hops when a direct connection with a social friend is not feasible.

E. Add/Remove Social Friends

In any social network friendships are formed and dissolved. At such events, peers p_i and p_j must not only update their neighborhood sets C_i and C_j, but also maintain their routing tables in order to augment or repair the entries. To accomplish this, p_i and p_j execute Algorithm 1 when either a new social connection occurs or an existing one is removed.

V. Experiments

We start our experimentation with large-scale simulations of our proposed approach. In order to measure the efficiency of our augmented Pastry-based overlay network, we implemented the *NewsFeed* functionality and used the following metrics:

- **Number of Hops:** The average number of overlay hops required to communicate two social friends.
- **Network latency:** The average time spent to propagate a message between two social friends.
- **Percentage of social connections:** The percentage of social links that are also overlay network connections.
- **Percentage of social entries:** The percentage of the routing table entries that correspond to social links.

To evaluate the performance of our approach for a DOSN service, we compared our approach with the basic Pastry P2P overlay network. Finally, we measured the performance of our approach against the socially augmented P2P overlay network of Chord, also called SPROUT [8].

A. Datasets

Our experimental evaluation is performed on four real-world data sets, listed in Table I. These datasets cover a wide range of social graph features, from less-connected graphs (Epinions) to high-connected graphs (Facebook), that help evaluate our proposed approach on several graph types. Moreover, we conduct experiments on the large-scale data set of Twitter in order to demonstrate the scalability of our algorithm. The characteristics of the data sets are presented in Table I.

TABLE I. The four real-world data sets [11], [12].

Data Set	Users	Connections	Average Degree
Facebook	63,731	817,090	25.642
Twitter	456.631	14,855,874	28.642
Slashdot	82,168	948,463	11.543
Epinions	75,879	508,837	6.706

B. Experimental Evaluation

For the experimental evaluation, we used the Discrete Event Simulator of FreePastry 2.1[1], an open source implementation of the Pastry P2P overlay network [4]. For each metric we report the average results out of 100 trials in order to reduce the statistical error.

[1]http://www.freepastry.org/ (last accessed 11/09/2015)

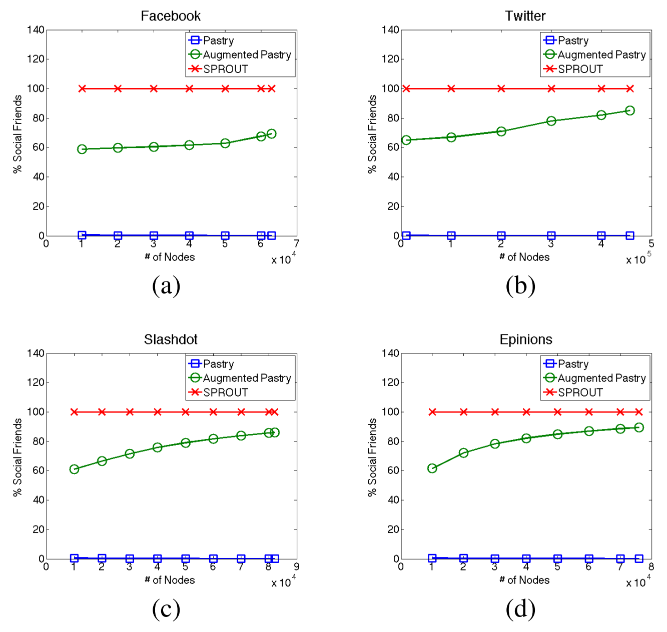


Fig. 2. Percentage of the social friends found in the routing table for the (a) Facebook, (b) Twitter, (c) Slashdot and (d) Epinions datasets.

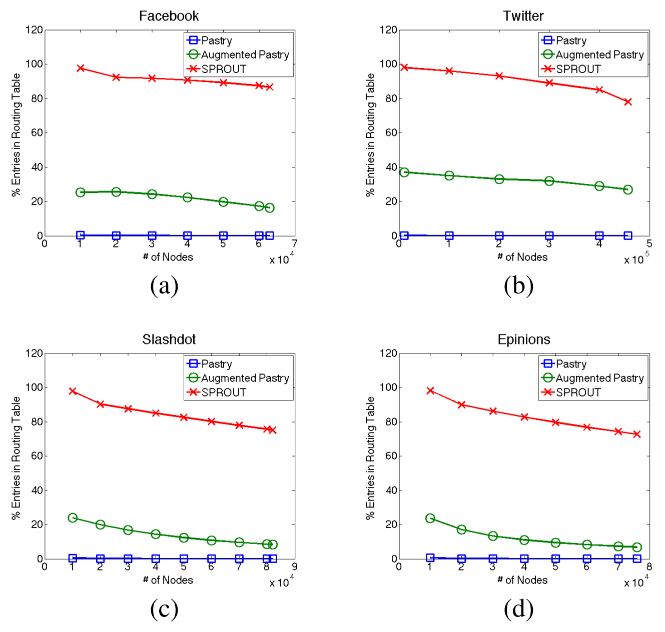


Fig. 3. Percentage of the entries in routing table that are social friends for the (a) Facebook, (b) Twitter, (c) Slashdot and (d) Epinions datasets.

Simulations are performed in evolving networks, where users join the DOSN at different phases. In order for a user to join the DOSN she must be invited by one of her social friends that is already a registered user. We initiate our simulation by selecting a social user u_i from the data set at random. Thereafter, we insert into the DOSN a portion of the user u_i's social friends, following the model of [13]. It is observed that social users establish friendship connections at high rate in the beginning of the join process, and that this rate decreases exponentially over time. Therefore, at each phase, we select a registered social user and insert into the social graph a number of her social friends that preserves the exponentially decreasing rate of the model.

In Figure 2, we present the average percentage of social connections that are also connections in the routing table T. The non-socially aware Pastry implementation contains almost none of the social links in the routing table since the probability that the social connections E overlap with the overlay connections F is extremely small (see Section III-B). However, SPROUT [8] creates direct connections to all of the social user's friends along with the $\log N$ direct connections that Chord maintains. In our approach, the percentage of social connections populating the routing table T increases from 60% to 90% as the network grows. This phenomenon occurs because the routing table size is increased based on the size of the network ($|T| = \lceil log_B N \rceil \times (B-1)$).

The impact that our approach has on the social make-up of the routing table, is presented in Figure 3. Our approach results in between $10\% - 30\%$ of the routing table corresponding to social connections in all datasets, compared to virtually zero percent for the standard Pastry overlay. Note that in our approach more than 60% of our social friends are included in the routing table T (recall Figure 2). Therefore, the rest of the NodeIDs share the same prefix with the an already included social connection. In comparison to our approach, SPROUT includes 70% more direct connections to socially connected users, which is to be expected as SPROUT connects to all of a user's friends.

By augmenting the routing table T, each user u_i communicates with more than 60% of his social friends in 1 overlay hop, while the communication with the rest of his friends is accomplished in $O(\log N)$ hops. As shown in Figure 4, our approach reduces the number of overlay hops required to disseminate the data between two social users by 1 to 6 hops in comparison to Pastry.

Finally, we measure the network latency required to propa-

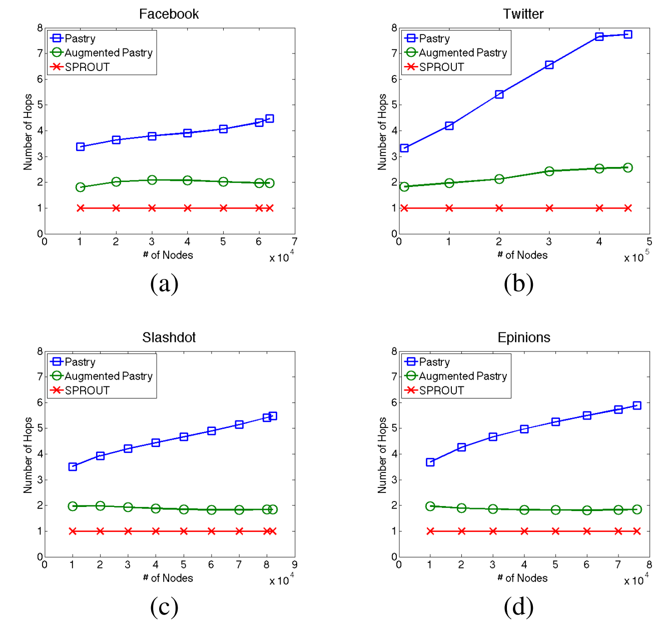


Fig. 4. Number of hops per social lookup for the (a) Facebook, (b) Twitter, (c) Slashdot and (d) Epinions datasets.

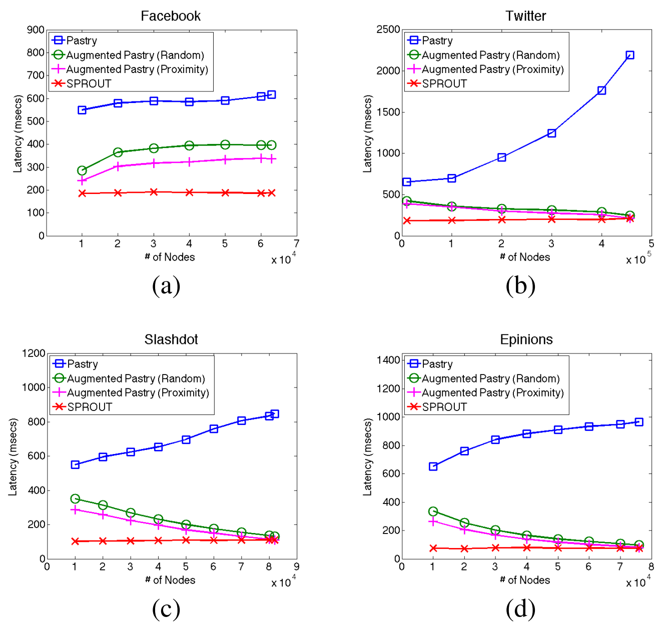


Fig. 5. Network latency per social lookup for the (a) Facebook, (b) Twitter, (c) Slashdot and (d) Epinions datasets.

gate a message between social friends. In Figure 5, we present the results of our approach against the baselines using both the random and the proximity strategy of augmentation. Regarding network latency, our approach reduces the propagation time by 600 and 100 milliseconds in comparison with the basic Pastry and the random selection strategy, respectively. The increasing rate that our approach presents in the Facebook data set, occurs because the Facebook graph is dense and the size of the routing table is increasing according to the size of the network. Therefore, at the first steps of our simulation each user u_i has more friends in his neighborhood set C_i than his routing table T_i is able to store.

As expected, SPROUT performs 200 to 50 milliseconds better latency than our approach in all data sets. In order to accomplish this, SPROUT establishes direct connections to all of the peer's social friends. However, in real social networks that follow the power-law distribution, a small portion of peers will need to establish a huge number of TCP connections due to their high degree. This results in extremely high communication overhead. Bounding the number of direct connections at each peer, as in our approach, balances the communication overhead, at the cost of increasing the network latency.

VI. Conclusion and Future Work

In this paper we proposed a socio-aware decentralized topology construction protocol for a DOSN with an emphasis on information dissemination between social users. Our approach uses an efficient P2P overlay network, called Pastry, and augments the routing table of each peer in order to maintain direct connections between friends. We preserve the Pastry properties, in terms of bounded number of direct connections and the logarithmic number of messages for each lookup process. We experimentally evaluated the superior performance of our approach against the basic Pastry implementation using simulations. Additionally, we compared our approach with a Chord alternative, called SPROUT. Integrating the social users' geographical location and interaction behavior [14] in the selection strategy of the peers' direct connections is a topic of our future work.

Acknowledgment

This work is supported by iSocial EU Marie Curie ITN project (FP7-PEOPLE-2012-ITN).

References

[1] 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 8th International Conference on Emerging Networking Experiments and Technologies*, 2012, pp. 337–348.

[2] D. Koll, J. Li, and X. Fu, "Soup: An online social network by the people, for the people," in *Proceedings of the 15th International Middleware Conference*, ser. Middleware, 2014, pp. 193–204.

[3] N. Kourtellis and A. Iamnitchi, "Leveraging peer centrality in the design of socially-informed peer-to-peer systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2364–2374, Sept 2014.

[4] 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 IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, ser. Middleware, 2001, pp. 329–350.

[5] 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.

[6] L. M. Aiello and G. Ruffo, "Lotusnet: Tunable privacy for distributed online social network services," *Computer Communications*, vol. 35, no. 1, pp. 75–88, Jan. 2012.

[7] L. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *Communications Magazine, IEEE*, vol. 47, no. 12, pp. 94–101, Dec 2009.

[8] S. Marti, P. Ganesan, and H. Garcia-Molina, "DHT routing using social links," in *Proceedings of the Third International Conference on Peer-to-Peer Systems*, 2004, pp. 100–111.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[10] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures*, 1997, pp. 311–320.

[11] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, 2009, pp. 37–42.

[12] "Stanford large network dataset collection," http://snap.stanford.edu, accessed Jul. 02, 2015.

[13] K. Zhu, W. Li, and X. Fu, "Modeling population growth in online social networks," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, 2013.

[14] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, and B. Y. Zhao, "Understanding latent interactions in online social networks," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010, pp. 369–382.