A Decentralized Online Social Network with Efficient User-Driven Replication

Rammohan Narendula, Thanasis G. Papaioannou, and Karl Aberer
School of Computer and Communication Sciences, EPFL, Switzerland
Email: firstname.lastname@epfl.ch

***Abstract*—Unprecedented growth of online social networks (OSNs) increasingly makes privacy advocates and government agencies worrisome alike. In this paper, we propose My3, a privacy-friendly decentralized alternative for online social networking. The My3 system exploits well-known interesting properties of the current online social networks in its novel design namely, locality of access, predictable access times, geolocalization of friends, unique access requirements of the social content, and implicit trust among friends. It allows users to exercise finer granular access control on the content, thus making My3 extremely privacy-preserving. Moreover, we propose different replication strategies that users may independently choose for meeting their personalized performance objectives. A detailed performance study evaluates the system regarding profile availability, access delay, freshness and storage load. By using real-world data traces, we prove that My3 offers high availability even with low average online time of users in the network.**

***Keywords*-privacy; trust; decentralized social networks**

I. Introduction

Online social network paradigm has taken the Web 2.0 into unprecedented scales by offering innovative tools for networking among users and distributing the user-generated content. The conventional social networks (e.g. Facebook.com, Google Plus) have recently seen an explosive growth. Facebook has currently nearly 900 million users active on the service at least once in a month. As a result, these OSNs have become store houses of unprecedented amount of data in the form of messages, photos, links, and personal information. Online social network portals continue to evolve towards one-stop hubs for content in a way that influences the future of the Internet [1]. However, social networking portals that are operated in a cloud infrastructure maintained by a single authority (e.g., Facebook or Google) will strategically have greater stakes in protecting the interests of the advertisers than addressing the privacy requirements of the users. During sign-up time, users consciously or unconsciously permit the organizations to share their personal information with third-parties in whatever form the organizations choose to [2]. In addition, the leakage of personal information from OSNs can be associated with the user activity on non-OSN sites as well [3]. Moreover, as social information owned by a single authority grows, so does its financial power. Even if we trust that the provider is motivated to protect user data, large-scale data breaches are still possible as reported recently. In order to address privacy concerns of OSN users, research community has resorted to the decentralized (often P2P-based) paradigm for OSN content management. Replacing the big-brother with a community of users, enables OSN users to have more control on their profile content.

In this paper, we present *My3*, a decentralized online social network that operates based on the resources contributed by its users. We briefly outline the system architecture originally described in [4] and mainly focus on the distributed storage layer. Specifically, we propose a number of profile replication algorithms that can be independently employed by users to meet different performance objectives of their choice, namely high profile availability, high data freshness (i.e., low delay for data consistency), low access delay, low storage overhead or a certain combination of them. The design of the My3 system exploits several properties of the conventional online social networks: (1) a user's friends are clustered in a few geographical locations and almost all the content accesses are initiated from these locations (2) access patterns and times of content accesses can be approximated with high precision to a large extent. By employing real data traces from Facebook and Twitter, we experimentally prove the effectiveness of our replication algorithms towards their respective goals when jointly or independently chosen by users. According to our results, a total -not necessarily continuous- online time of 40 minutes by a user, is enough for achieving availability higher than 90% with 4-5 profile replicas.

The remainder of this paper is organized as follows: In Section II, we discuss the background behind the My3 system design. The storage layer is discussed in Section III, followed by replication algorithms. In Section VI, we evaluate the performance of the My3 design w.r.t. several criteria. In Section VII, we discuss the related work and, finally in Section VIII, we conclude and outline our future work.

II. Motivation

In this section, we explain the properties of the conventional social networks that drive the design of My3 and its replica selection algorithms.

We observe that every user in an OSN has friends scattered over a limited set of geographical locations (e.g. his home town, working location, home country, location of previous institute etc.) as shown in Fig. 1. This fact can be exploited to choose replicas located in one or more of these locations so that content stays in the proximity of the friends.

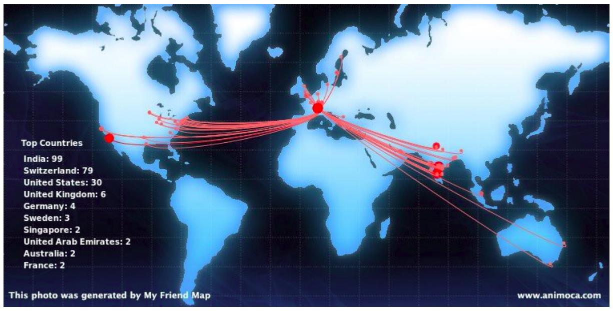


Fig. 1: Geo-clustering of a user's friends in Facebook.

Moreover, users in an online social network exhibit certain online time patterns [5], [6] which can be exploited in choosing the replication points so that replicas' online time patterns overlap with that of friends of a user. Note that, in a typical OSN, most of the user's profile content is accessible only to his friends, unlike typical web content which is world-accessible. The *My3* system exploits the trust relationships among friends to improve the availability. We assume that a user of *My3* runs the client on his office or personal laptop/computer. Hence, we use the terms *user* and *node* interchangeably.

We consider optimizing at a single user level in all our algorithms and not system-level optimization, as each user runs these algorithms independently from others in a distributed setting. My3 allows users to personalize their profile configuration in several dimensions such as availability, responsiveness, privacy risk, etc. We believe that a single global configuration policy for all the users in the system offered by typical conventional social networks like Facebook, deprive the users their autonomy on their own data.

A user u's profile content is hosted only on a set of self-chosen trusted nodes, which enforce access control on the content on behalf of the user. This set of trusted nodes for a user is referred to as his *trusted proxy set (TPS)*. The TPS members for a user are selected so as to fulfill certain performance goals.

III. Storage Layer

In this section, we discuss the storage mechanism of the *My3* system which is an enhancement from [4], and mainly address the construction of the set $TPS(u)$ for a user u from his social graph. For completeness, we revisit briefly the storage mechanism presented in [4].

The social network graph is denoted as $G(U, R)$, where U is the set of users represented by the vertices in the graph and R is the set of friendship relations represented by edges. For example, an edge between two vertices u_1 and u_2 models the fact that users u_1, u_2 are friends. We assume that friendship relationships are symmetric. This is the default assumption in current OSN applications, e.g. Orkut, Facebook. We use the notation $N_G(u)$ to represent the set of neighbors (i.e. friends on the OSN) of user u in the social graph G, and $N_G[u]$ to represent $N_G(u) \cup \{u\}$.

We assume that each user u in the social network is characterized with two parameters: his *geographical location* and *online time period*. For instance, the location can be set to the country/city where the user is currently located. We exploit location information of friends of a user, in order to place data as close as possible to the nodes that most frequently access the data for getting profile updates etc.

This is quantified by the metric *access cost* $C_{u_1}^{u_2}$ between two geographical locations/users/nodes u_1 and u_2, which is defined as the IP latency between those two locations, for example, a measured in [7].

Online time period represents the usual time that the user is online in the social network. This is a continuous/discrete time period, with a predefined granularity (e.g., minutes, hours), during which the user is active on the network and contributes bandwidth, storage, etc. through his OSN client. This parameter can be either a user input to the client or approximated by the client from the user's online history (for example, as done in the later part of the paper). Beyond this time window frame, the user is offline. We denote the location and online time period parameters for a user u as L_u and OT_u respectively. Given two users u_1 and u_2's locations and online time settings, we argue that they can contact each other and thus exchange data if and only if their online time intervals overlap, which we represent by the condition that $OT_{u_1} \cap OT_{u_2} \neq \emptyset$.

A. Trusted Proxy Set

Each user u selects some of his neighbors as *trusted* nodes. The user trusts these nodes both for storing his profile content and for enforcing access control on the access requests. We believe leveraging mutual trust relationships for access control enforcement, in place of encryption-based-access control simplifies the system to a great extent, especially given that a typical user in any OSN has millions of data objects but of very small size. We envision that users mutually cooperate for hosting content and delegating access control with some social contracts, even though *My3* design does not assume that trust is mutual between friends. However, the intuition is that users do not breach the delegation responsibilities because of social pressure and monitoring. Alternative solutions, which employ encryption mechanisms for access control and content storage [8], not only involve complicated key management issues, but also, are highly inefficient in terms of storage overhead, as the same data item may need to be encrypted multiple times for different users with different access rights. However, trust in a user, may not translate to trust in his machine/ node. We acknowledge that detecting compromised nodes is a research problem in itself and assume that any of the existing mechanisms [9] can be deployed for this purpose.

Let $T(u) \subseteq N_G(u)$ be the set of trusted users/nodes for a user u. $T[u]$ also includes the user u himself in the set of trusted nodes. The user selects a subset of these trusted users for hosting his content. We call this set as *trusted proxy set (TPS)* ($TPS(u) \subseteq T(u)$). The content of u is stored on the members of the set $TPS[u]$ ($= TPS(u) \cup \{u\}$).

Next, we describe several algorithms for the computation of the set $TPS[u]$. These algorithms build an *online time graph*

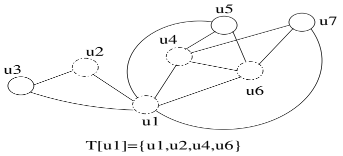


Fig. 2: The graph OG_{u_1}

for each user and compute the above set from this graph.

Definition 1: Online time graph: for a user u (denoted by OG_u) is defined as $(N_G[u], E)$ where $N_G[u]$ is the set of vertices and E is the set of edges, such that

$$\forall v_1, v_2 \in N_G[u], \ \exists \text{ an edge}(v_1, v_2) \in E \ \text{ iff}$$
$$(v_1 \in T[u] \vee v_2 \in T[u]) \wedge (OT_{v_1} \cap OT_{v_2} \neq \emptyset)$$

Next, we specify the following two conditions on the graph OG_u, which are necessary and sufficient in order to compute a valid storage configuration.

1) OG_u must be connected. Only then, every user in the set $N_G[u]$ can access u's content.
2) The sub-graph induced by the set $T[u]$ i.e., the graph $OG_u[T[u]]$ must also be connected, in order to allow content synchronization across TPS members pass through only trusted nodes[1].

We suppose that each user constructs OG_u offline locally from online time (OT) specifications of his friends. The construction of OG_u is explained with the following example. Assume a user u_1 with neighbors in the OSN u_2 to u_7 and their locations set as follows: L_{u_1} is *Switzerland*, L_{u_2} and L_{u_3} are *India*, and finally the rest are *US-West*. Assume OT set to 8am to 5pm local time for all users. Let $T[u_1] = \{u_1, u_2, u_4, u_6\}$. The resulting OG_{u_1} is shown in Figure 2.

We found that, in the case of real-world datasets (explained in Section VI, such a graph $OG_u[T[u]]$ is connected for a realistic online time model for more than 95% of the users. We skip the discussion of handling the modifications in the social graph, for brevity reasons.

IV. Replication Strategies

In this section, we describe several algorithms used by My3 for replica selection, which compute TPS optimizing a certain objective function, as explained below.

A. Maximizing the availability

In this approach, the trusted friends that maximize the availability of the user profile are chosen as replica locations. The maximum achievable availability for a user u is limited by $\mid \cup_{f \in T[u]} OT_f \mid$. Hence, the replica selection algorithm should choose the minimum number of replicas/friends that jointly achieve this availability. We model this problem as the conventional *set cover problem* with the set to be covered (the *universe*) chosen as $\cup_{f \in T[u]} OT_f$. Since finding an optimal solution for the set cover problem is NP-hard, we solve the problem with a greedy algorithm as follows: Initially, the node u is added to the TPS. At each step, a trusted friend that has the longest non-overlapping (i.e. uncovered) online time as compared to the current TPS members is added to the $TPS[u]$ until no improvement is observed in the achieved availability. Only the friends that are connected in the online time graph to any of the current $TPS[u]$ members are considered in each step. The availability of TPS is the fraction of sum of its member's online times over a day (i.e., 24 hours).

B. Minimize the number of replicas (MNR)

The MNR approach minimizes the number of replicas to be maintained for a user, so as to minimize the storage and replica management overhead. This approach exploits the fact that the set TPS can be modeled as the *minimum connected dominating set (MCDS)* on the graph OG_u, with the additional constraint that the members of the MCDS must belong to $T[u]$. Hereby, we modify a greedy algorithm from [10] to solve this variant of the MCDS problem.

Algorithm 1 The MNR algorithm.

```
Mark all $v \in OG_u$ as $white$
Mark $u$ as $black$
Mark all neighbors of $u$ in $OG_u$ as $grey$
while $\exists$ a $white$ node in $OG_u$ do
    Select a $grey$ $v' \in T(u)$ such that $v'$ has the highest number of $white$ neighbors in $OG_u$
    Mark $v'$ as $black$ and its neighbors as $grey$
end while
$TPS[u]$ is the set of all $black$ nodes in $OG_u$
```

C. Minimizing update propagation delay (MPD)

This algorithm minimizes the *update propagation delay* among replicas, which is the delay in time between the time instance, an update occurs on a user profile at one of the replicas and the instance, the update reaches all the other replicas. We explain the calculation of this delay in the example of Fig. 3. We assume three replicas of a certain user's (say user v) profile residing at nodes v_1, v_2, and v_3 with different continuous online times represented with begin (t_s) and end (t_e) times as $OT_{v_1} = [t_s^{(v_1)}, t_e^{(v_1)}]$, $OT_{v_2} = [t_s^{(v_2)}, t_e^{(v_2)}]$, $OT_{v_3} = [t_s^{(v_3)}, t_e^{(v_3)}]$, for which the replica time connectivity graph is also shown in the figure. Let an update event happen at replica v_1 at time t. Then, this update would be communicated to v_2 at time t', which would take $24 - d_1$ hours, where d_1 is number of overlapping hours between v_1 and v_2. Furthermore, since at time t' node v_3 is not online, in order for the update to reach the replica v_3, it would take an additional $24 - d_2$ hours, where d_2 is the gap between t' and $t_s^{(v_2)}$ in hours. Thus, in total the update propagation delay between v_1 and v_3 would take $48 - d_1 - d_2$ hours, which is the worst possible case for communicating a profile replica update at node v_1 to node v_3.

Given this, the *Update Propagation Delay* for a user is the maximum of propagation delays between all pairs of the replicas. It is the weight of the longest of the shortest paths

[1] However, as long as the first condition is met, nodes from the set $T[u]$ can be removed one by one until the resulting induced graph becomes connected.

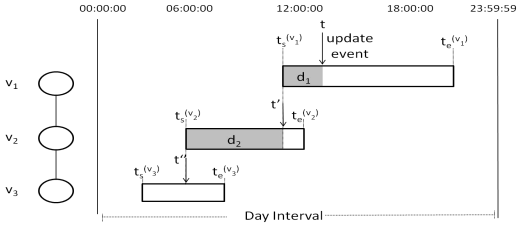


Fig. 3: Propagation of an update from replica v_1 to v_3.

among all pairs of replicas in the above graph. Hence, in above example, the update propagation delay for the user v is $48 - d_1 - d_2$ hours. This metric captures the maximum/worst case update propagation delay for transferring updates among replicas of a given user profile. This metric directly impacts the data freshness.

This algorithm minimizes the worst case propagation delay for the user u. We compute a modified weighted online time graph OG'' from OG by assigning weights to edges between trusted nodes, equal to the propagation delay between the corresponding end nodes. Then the problem of the TPS computation is finding the $MCDS$ in graph OG'' such that the weight of the longest shortest path between any two nodes in TPS in OG'' is minimum. The $WP(S)$ function returns the weight of the longest shortest path in the sub-graph induced by the set S on OG''.

Algorithm 2 The MPD algorithm.

```
Mark all $v \in OG''_u$ as $white$
Mark $u$ as $black$
add $u$ to $TPS$
Mark all neighbors of $u$ in $OG''_u$ as $grey$
**while** $\exists$ a $white$ node in $OG''_u$ **do**
    Select a $grey$ $v' \in T(u)$ such that $WP(TPS \cup v')$ is the minimum
    Mark $v'$ as $black$ and its neighbors as $grey$
    add $v'$ to $TPS$
**end while**
```

D. Minimizing the access cost (MAC)

The MAC approach prioritizes only the access cost for each friend in a user's social network. Hence, for every user v in OG_u, it chooses the nearest (i.e., with minimum access cost) trusted node into the set $TPS[u]$. This algorithm always chooses all the trusted nodes into the TPS set. Thus, it uses all the possible replicas resulting in extensive replication.

E. Maximizing the replication gain

This approach quantifies the *replication gain* of a given subset of trusted nodes set (x) and, explores the entire solution space to pick the right set with the minimum effective cost as TPS. The storage cost is measured in terms of the total cost incurred for accessing and updating the profile content by friends, in addition to that of replica synchronization among all TPS members.

Replicating a user's profile increases the availability of the profile, reduces the average access cost per friend in accessing the profile. But it induces an overhead in the form of update propagation delay among replicas. All these three factors are merged into a weighted objective function with tunable weights to each of the factors. The algorithm initializes TPS (i.e. set x) to node u to begin with. Then it adds one member (i) at a time, from the trusted set which maximizes the following objective function, until the resulting TPS is a minimum connected dominating set over the graph OG_u.

$$\Big[w_1 \cdot \frac{availability(x \cup \{i\}) - availability(x)}{availability(x)} + w_2 \cdot \left| \frac{avg(\{C_v^x\})}{avg(\{C_v^{x\cup\{i\}}\}) - avg(\{C_v^x\})} \right| - w_3 \cdot \frac{WP(x \cup \{i\}) - WP(x)}{WP(x)} \Big] \text{where-} v \in N_G(u)$$

The access cost between a node (v) and a set of nodes (x) is the access cost between v and its nearest node in x. The function WP is explained above. This algorithm is referred to as *Hybrid* in the evaluation (Section VI).

V. Data Consistency

As different replicas of the profile accept update requests from the friends of the user in an asynchronous way, there is a need for synchronizing the profiles on all replicas. We propose that after every update, the concerned replica *pushes* the update to other TPS members during their online time frame. Note that $OG_u[T[u]]$ is connected. Assume that each TPS member is informed of other members by the user u during TPS creation. Until recent updates reach a replica, it continues to serve access requests with out-dated content, which is acceptable, as *My3* aims for eventual consistency among replicas with tolerable temporary inconsistencies.

My3 views the content of a profile as a collection of data objects e.g., a status message, user's metadata, a photo album, a photo. A data object (say, a photo album) can be further decomposed into another collection of objects (resp. photos). *My3* employs vector clocks of size the number of TPS members in the system. It maintains one vector clock per object. A vector clock of an object is updated when there is an update in the object and however, they are not updated when its constituting objects' vector clocks are updated. For example, when a comment is posted on a photo, thus the photo object is updated, the corresponding vector clock of the photo is updated and that of the photo album object is kept intact. If the photo is deleted by the owner user, then the vector clock of the album object is updated as this deletion is an update on the album as such.

Updates on a profile are *pushed* immediately by a replica to all other replicas. In addition, when a replica comes online, it announces itself to all other online replicas and *pulls* any buffered updates, as explained below.

Each replica buffers a transaction record of an update on its copy of a user's profile until a time period (e.g., twice the maximum propagation delay for the user). This record holds all the meta information corresponding to the update so that other replica, on receiving the record, can replicate the update exactly. Two replicas when come in contact, exchange all the records in the buffer and apply the records on the corresponding data objects. When concurrent events are detected on an object (using the vector clocks stored in the records), the two replicas have to decide on ordering the events. Even though any ordering of these events results in a valid profile only (as typical updates on an object will be append-only updates), we propose the events to be ordered according to the replica identifiers in order to achieve a total order of events among all the replicas. This results in a consistent view for the users when they access the profiles across replicas. In order to achieve this, the transaction record should contain the replica id of the replica which originally received that corresponding update event as a replica may receive a transaction record from multiple replicas because of asynchronous update propagation and nodes' intermittent online connectivity.

However, this ordering resolution does not take the actual semantics of the updates into account and hence may lead to semantically incorrect profile objects occasionally. We expect the owner of the profile to inspect his profile updates time to time and fix such semantic incorrectness, though we believe that such an intervention is needed very rarely. The resulting ordering of the events must be *forced* onto other replicas which should replace corresponding object parts with the one received from the owner. Thus, the owner replica can be said to the *leader* of all other replicas. For brevity, additional details of the consistency mechanism are skipped.

VI. Evaluation

In this section, we illustrate the performance evaluation of the proposed My3 storage in detail, using real-world datasets of Facebook and Twitter social networks. First, we present description of the datasets and then layout the criterion for evaluating the performance by introducing the metrics, followed by analysis of the results.

A. Dataset description

In order to model the essential parameter of My3 algorithms, the online times of users in a social network, we needed, apart from the social network graph, a dataset with users' usual activities on the social network including the timing of the activities information [11] and their geographical locations. Two datasets- a Facebook dataset [12] and a Twitter dataset [13] met our requirements. The user degree distribution of both the datasets is presented in Fig. 4, which is the number of friends (resp. followers) in the social network Facebook (resp. Twitter). The activity considered were the wall posts (for Facebook dataset), the user's tweets (for Twitter dataset).

1) Facebook dataset: The Facebook dataset employed is the NewOrleans Network dataset [12], which has a total of 63,731 users creating a total of 876,994 wall-posts. A wall-post has a receiver, a creator, and a timestamp.

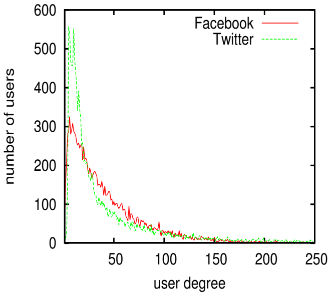


Fig. 4: Distribution of user degree distribution.

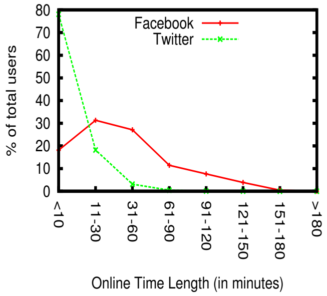


Fig. 5: Distribution of derived user online times.

In a decentralized Facebook realized using the My3 system, a user's profile is accessed (by his friends) from any of the profile replicas which are online at that instant and a wall-post/update made should be sent to the corresponding replica.

2) Twitter: We employed a simplified version of the Twitter dataset of [13], which originally included 158,324 tweets made by a total of 23,162 users in Twitter between 10-Sep-2009 and 24-Sep-2009. From this dataset, we excluded all the users whose followers are not present in the dataset. A tweet has a receiver, a creator, and a timestamp, similar to a wall-post described before. The dataset contains users from a wide range of geographical locations.

In a decentralized Twitter over the My3 storage, a user's profile can be replicated on his followers. This is a natural choice as the majority of the information flow in Twitter is from the user to his followers. When a user is offline, his replicas are used by his followers to access his tweets and by his followees (users followed by him) to communicate their tweets to him from (i.e., tweets of followees) are communicated to his replicas when he is offline. Moreover, followers of the offline user can access his past tweets from his profile replicas.

We filtered out users with very little activity (less than 10 wall-posts or tweets) from the above datasets. We ended up with a total number of 13884 users for Facebook, with the average degree 41 (i.e. friends) and an average number of 50 activities per user. For Twitter, the filtered dataset contains $14,933$ users with average degree of 76 (i.e. followers).

B. Methodology

We built a Java-based My3 simulator for the evaluation. All the users in the datasets are modeled inside the simulator and the activity stream present in the datasets is replayed among the user objects, separately for Facebook and Twitter. The proposed performance metrics are quantified at the end.

However, the My3 algorithms need two important inputs: one, the online times of users in the network and second, the trusted friends to be used as replicas for a user's profile. We modeled the two inputs as follows:

1) Modeling user online times: In order to approximate typical users' online durations, two possibilities exist for the context of My3, a decentralized social network. One possibility is to position My3 as an alternative to the conventional social networks like Facebook and emulate the online behaviors of the users on Facebook [6] or Orkut [5] in the My3 system for the evaluation. Alternatively, position the My3 clients as analogous to P2P clients for communication, like Skype and emulate typical Skype nodes' session times [14] to approximate the My3 client's online times.

In this paper, we model My3 client's online times as follows: from the study of user session times in Facebook, we derive an online time distribution for My3 clients. During the beginning of the simulation, we choose an online time for a user from this distribution and the user runs his client for this amount of time, every time he is active on the social network. From this online time distribution, we plot percentage of users in the system that have a particular online time for both Facebook and Twitter cases, which is shown in Fig. 5.

Once the length of a user's online times is chosen like above, the actual online times (time-of-the-day) are computed as follows: for each activity present in the input activity dataset, the user is assumed to be online for the above duration with the activity occurrence positioned at a random instance in this online window. This is done using the time stamp information present with each activity in the datasets. For example, assume that the online time length chosen for a user is 5 minutes. If the user's activity is found in the dataset to be done at 8:03am, the user is said to be active on the social network from 8:00am to 8:05am if the activity is positioned at 3rd minute of the session. Likewise, a user is online during the day for a total time equivalent to number of activities in the dataset times the length of the online time associated with the user.

For the case of Facebook dataset, My3 clients are online on average for 42 minutes with a minimum and a maximum online time of 2 minutes and 194 minutes respectively. Corresponding online times for the Twitter dataset case are: average-9min, min-2min, and max- 156min. From Fig. 5, one can note that for the case of Facebook, 80% (and for Twitter, 20%) of the users are online in a day for a total of less than 10minutes.

2) Selecting trusted friends: We imagine a use case for My3 where a user manually feeds the set of trusted friends to the My3 algorithms. However, for the evaluation sake, we model the trusted friends as the most active friends, friends who made majority of the activity on a user's profile: wallposts in the case of Facebook and tweets in the case of Twitter. We argue that this is a natural choice as friends with very close acquaintance usually interact with a user the most, thus enjoy high degree of trust.

In our evaluation, we choose *top-k* most active friends as the trusted friends with k varied from 0 to 10.

3) Modeling access latencies: We model the network latencies between two My3 clients as the network latency between the corresponding geographic locations of the users as given in the input datasets. For Facebook dataset, all users are based in a single location and thus network latency between any two users is the same and set to 1 in our experiments. For the case of Twitter, we queried Twitter APIs to retrieve the locations of the users appearing in the dataset. For actual network latency statistics, we used Verizon [7] network latency dataset for the month of April 2012. If a user's (in Twitter dataset) location is not found in the latency dataset, we chose a random location from the dataset as his location. The problem of how latencies can be approximated between two geo locations is beyond the scope of the paper and the Verizon dataset is used for exemplary purpose only. The *My3* performance trends, as such, are in general applicable to any other latency computation techniques [15].

C. Performance metrics

We enumerate several metrics to evaluate the My3 system [11].

1) Availability: is the fraction of time in a day, a user's profile is reachable through his replicas. For example, if a user's profile is available for 12 hours in a day, the availability is 50%. Note that availability of a user's profile in My3 is limited to the union of online times of all of his trusted friends.

2) Availability-on-Demand: measures to what extent a user's profile is accessible to only his friends (in contrast to the availability of a user described above). It is the fraction of total time, a user's friends are online (which is size of the union of their online times), his profile is available through his replicas. In a privacy-conscious social network, a user's profile is typically accessible only to his friends and hence higher availability-of-demand (even with a lower availability) is desirable.

3) Propagation Delay: is the delay in time between the time instance an update occurs on a user profile at one of the replicas and the instance where the update reaches the last replica. The calculation of delay for a user is detailed in [11].

4) Access Cost: is the average network latency between friends of a user and the nearest replica.

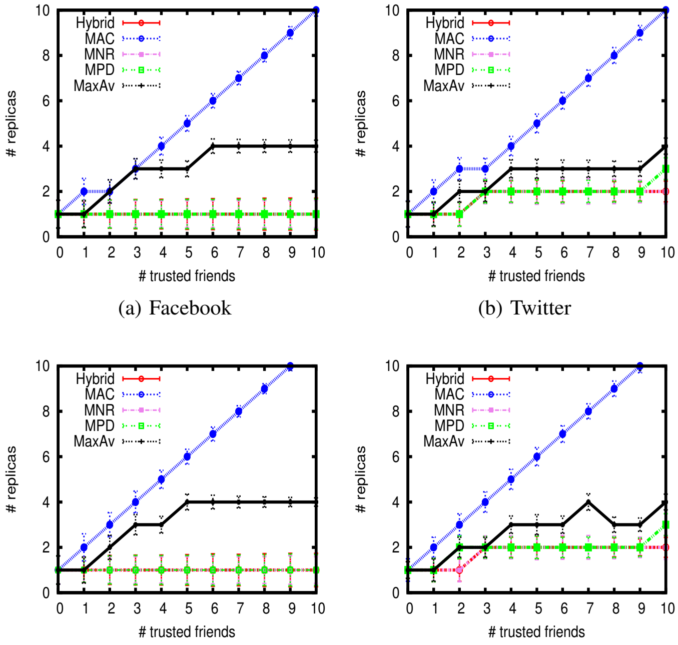
5) Load: of a given user is the number of profiles stored on the user as part of whole social network level replication. A popular and typically most active user in the social network may end up being the trusted friend for a very high number of users in the network, thus hosting all of their profiles. A good storage algorithm should balance the load on the replicas in order to ensure fairness and minimize the maximum load in the system.

D. Results

We explore two scenarios where a single system-level choice on replica placement is made and all the users run same corresponding algorithm. In second case, each user chooses locally the most preferred replica placement strategy that meets his criterion. In the $Hybrid$ algorithm case, all the factors in the object function were given equal weight (i.e., $w_1 = w_2 = w_3$).

1) System level replica placement choices: Here, we present the observed results first, considering all the users in the system and then, considering users with a particular number of friends (i.e., degree). We considered a degree of 20 for this case.

As mentioned earlier, the number of trusted friends is varied from 0 to 10. However, based on the objective of individual replica selection algorithms, not all the trusted friends might be used for replication. The actual number of replicas thus, chosen is presented in Fig. 6a and 6b for the case of Facebook and Twitter respectively (the number of replicas counts the replica on the user client also). The system level averages are shown. As expected, the MAC algorithm uses the highest number of replicas. It grows linearly with the number of trusted friends. Since some users may have a lesser degree than the input number of trusted friends, the number of replicas is lesser for MAC case (however, for the case of users with degree 20 (Fig. 6c and Fig. 6d), we can see that number of replicas chosen is same as that of number of trusted friends counting the user himself in addition). All the other algorithms show a flattened behavior after a point as no improvement in their objective criterion is observed with increase in number of replicas. In Fig. 7, the performance of the replica selection algorithm w.r.t the metrics described in VI-C. The metrics for quantified for each user in the system and average of all users is plotted in Fig. 7a to Fig. 7c for the Facebook case and in Fig. 7d to Fig. 7f. It is to be noted that for each point in the plots, the corresponding values for the metrics the achieved performance with the corresponding number of replicas shown in Fig. 6a and Fig. 6b. The MAC and $MaxAv$ achieve the highest availability (Fig. 7a). But $MaxAv$ achieves the same availability as MAC with much lesser replication degree, for example at half of the replication for $k = 10$ as evident from Fig. 6a. It is impressive to note that an availability of 90% is achieved in spite of a very less total online time of users, an average of 40 minutes from Fig. 5 for the Facebook case and mere 9 minutes for Twitter case. Similar availability performance of the MNR and MPD algorithms can not be explained alone with the fact that both choose the same number of replicas as shown in Fig. 6a. Because they exhibit differently in the case of other metrics. It is due to the chosen replicas are together online for same time window. However, the $Hybrid$ algorithm makes a better selection of replicas and results in better availability with reduced delay (Fig. 7c).



(a) Facebook

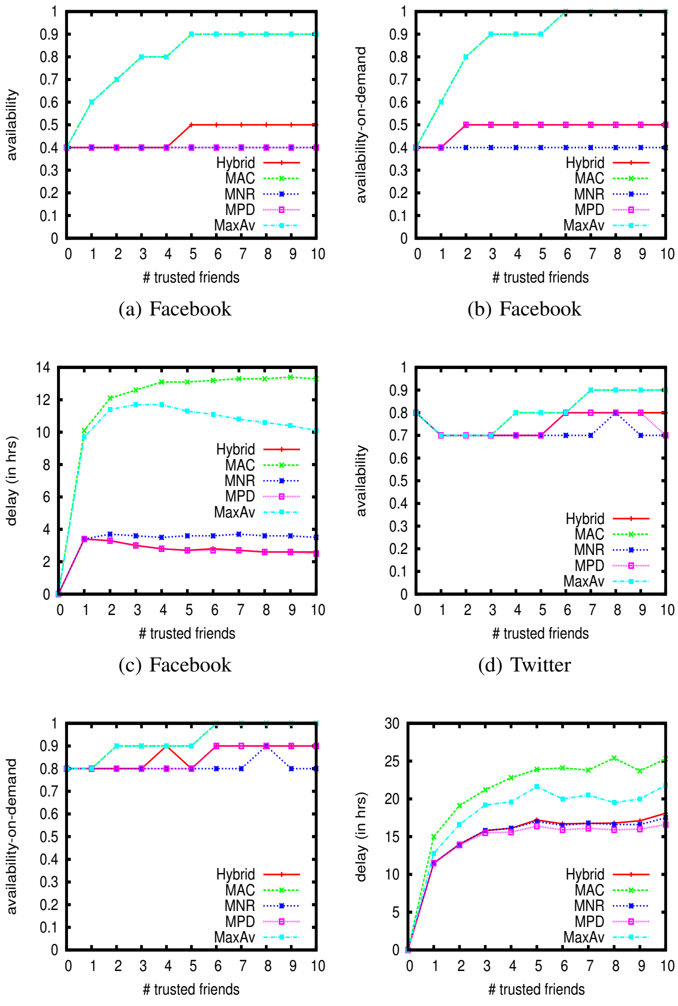
(b) Twitter

(c) Facebook

(d) Twitter

Fig. 6: Number of replicas chosen: (a),(b): average of all users, (c),(d): average of users with degree 20.

The average availability-on-demand reaches 1 for $k = 6$. Given the average degree of 41 for Facebook and 76 for Twitter, a replication degree of 6 is very promising. Note that in a privacy-friendly OSN, the profile content should be more available to friends of a user only and the actual availability is of secondary importance. The update propagation delay (denoted as “delay” in the plots) performance is depicted in Fig. 7c and Fig. 7f. Thanks to the highest number of replicas chosen, the MAC incurs the worst update propagation delay among replicas due to incomplete overlap among online time windows of the replicas. Even though the MPD algorithm shows up the least delay which is inline with its objective.



(a) Facebook

(b) Facebook

(c) Facebook

(d) Twitter

(e) Twitter

(f) Twitter

Fig. 7: Average of all users.

Note that MPD chooses same number of replicas as MNR, but different ones that minimize the delay. In case of Twitter also, the MPD shows the best delay performance (Fig. 7f). The access cost performance for Twitter case is presented in Fig. 8. Note that for the Facebook case, all the users are from a single location (NewOrleans) and hence, this study is not applicable. As the number of replicas grows in MAC, the average access cost is significantly reduced because increased number of replicas place the content in the close proximity of many friends. This decreasing trend can be observed with other algorithms as well. $MaxAv$ stands as the next best due to higher number of replicas chosen compared to other algorithms (from Fig. 6a).

Given the system- level averages, the actual distribution of different values of the metrics is illustrated in Fig. 9 for the case of availability and load. We chose the case of 10 trusted friends for these plots. For around 75% users in Facebook, the availability touches 1 for $MaxAv$ and MAC algorithms. For the other algorithms, the availability is uniformly distributed. Regarding the load metric, MAC typically increases the load on users in the network because of extensive replication. There are around 0.7% of users in the network hosting more than 100 profiles in Facebook case. In case of Twitter too, we can see that MAC increases the load significantly compared to other algorithms.

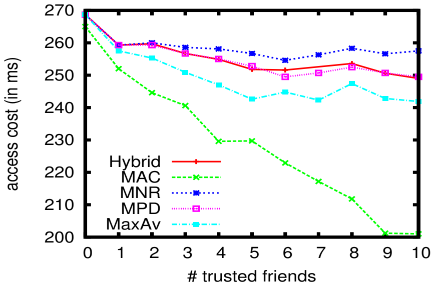
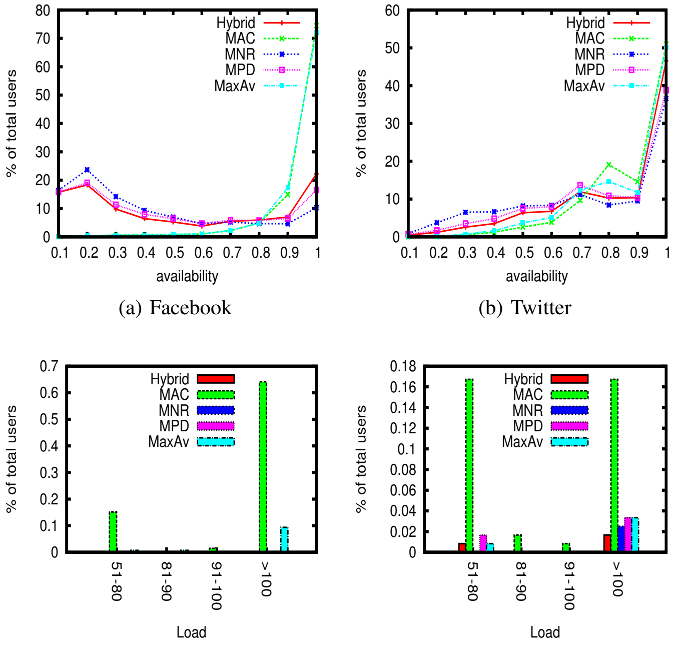


Fig. 8: Twitter average of all users.

In Fig. 10, we consider only the users for a particular degree of 20 to see the trends in the performance for such users. They exhibit trends comparable to the system-at-large as shown in Fig. 7.

2) User level replica placement choices: Since My3 centers the storage design around a single user, a user in the network can locally decide the replica selection criterion based on his objectives. To this end, we studied system level performance in case of informed personal choices users make in the network. The distribution of availability (for $k = 10$) is depicted in Fig. 11. Other metrics are skipped for brevity. Users choose one of the listed storage selection algorithms with a uniform probability. It is interesting to note that users retain their performance benefits of a particular replication choice, even individual users in the system choose different algorithms locally. The trends observed in Fig. 11 match exactly the ones observed for the case of single system level algorithm choice in Fig. 9a and Fig. 9b. The total load in the network is observed to be less than 5 per each user which is several times improvement over system-level choice (for example MAC).



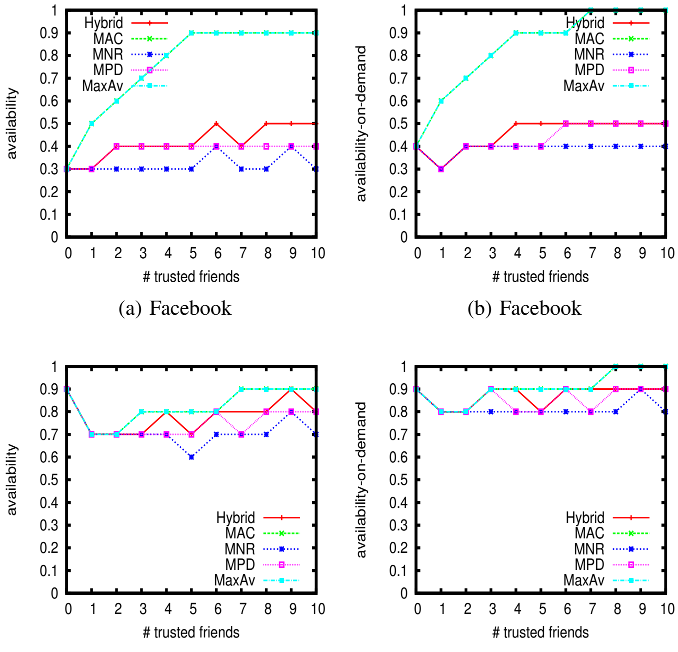
(a) Facebook

(b) Twitter

(c) Facebook

(d) Twitter

Fig. 9: Distribution of availability and load.



(a) Facebook

(b) Facebook

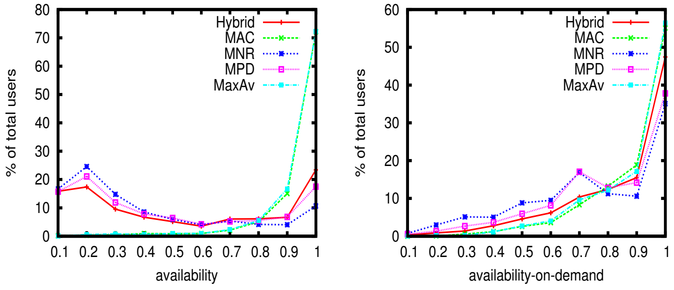
(c) Twitter

(d) Twitter

Fig. 10: Average of users with degree 20.

E. Discussion

The MAC and $MaxAv$ improve the availability at cost of increased replication. An extremely privacy-conscious user



(a) Facebook

(b) Twitter

Fig. 11: Distribution of availability: User- level replica choices.

prefers to have as lesser replication as possible and thus may choose MNR for replica selection. The availability can be increased in such case only by increasing the span of online window of the user himself or his trusted friends. The $Hybrid$ algorithm should be preferred in general as it exhibits increased availability over MPD still showing similar performance for other metrics. In addition, its tunability of different factors in the objective function offers additional flexibility in replica selection. The update propagation delay can be nullified by suitable 3rd party infrastructure (like a cloud or DHT) for update propagation. One of ongoing works to enhance My3 is to build an encryption based updated propagation infrastructure using storage resources of users in the social network alone. In the case of highly dynamic unpredictable user online behaviors, such a storage can be used to store profile content, in addition to update propagation among replicas.

VII. Related Work

There is significant related work on privacy issues in social networks. The work in [16] highlights the disparity in the desired privacy settings on OSNs and the actual settings provided. It also quantifies the process of managing privacy. In [17] authors show that configuring privacy settings in the online social networks is a daunting task. The possibility for involuntary personal information leakage in current social networks is highlighted in [18], e.g., by means of certain OSN features like annotating or tagging user photos, and its effects are demonstrated in [3].

The Lockr system [19] improves the privacy of centralized and decentralized content sharing systems. It allows users to control their own social information by decoupling the social networking information from other OSN functionality using social attestations, which act like capabilities. However, these social attestations are used only for authentication and authorization is enforced using separate authorization policies. Persona [20] uses attribute-based encryption to realize privacy-preserving OSNs. The attributes a user has (e.g., friend, family member, colleague) determine what data he can access. The NOYB approach [2] adopts a novel approach for preserving content privacy. They observe that if users address their privacy issues themselves by hosting encrypted content on OSNs, they could be expelled from the OSN by the OSN operator. Hence, they propose to replace users profile content items with "fake" items randomly picked from a dictionary. NOYB encrypts the index of the user's item in this dictionary and uses the ciphered index to pick the substitute. On the other hand, flyByNight [21] encrypts the users' content that hosts on the OSN.

Recently, the issue of using decentralized infrastructures for organizing OSNs in a privacy-preserving manner, was addressed by the research community [22], [8], [23]. In [24], the authors perform an experimental evaluation of hosting OSN content from homes as a possible decentralized OSN. PeerSon [23] adopts encryption mechanisms for content storage and access control enforcement. It uses a two-tier architecture in which the first tier is a DHT, which is used as a common storage by all participants. The second tier consists of peers and contains the user data. The DHT stores the meta-data required to find users. Peers connect each other directly, exchange the content, and then disconnect. The work in [8] addresses privacy in OSNs by storing profile content in a P2P storage infrastructure. Each user in the OSN defines his own view ("matryoshka") of the system. In this view, nodes are organized in concentric rings, having nodes at each ring trusted by the nodes in its immediate inner ring, with the user node being the center of all rings. The user's profile data is stored encrypted at the innermost ring, which is accessed by other users through multi-hop anonymous communication across this set of concentric rings. In the DHT, an entry for a user with the list of nodes in the outermost ring is added. Thus, [8] achieves both content privacy (using encryption) and anonymity of searcher and hosting nodes, yet limited content discovery and profile availability, as opposed to our approach. DECENT [25] proposes a DHT based storage for OSNs with a special focus on security and privacy using encryption mechanisms.

A decentralized OSN, Vis-à-Vis is proposed in [22], where, a user's profile content is stored at his own machine called as virtual individual server (VIS). VISs self-organize into P2P overlays, one overlay per social group what has access to content stored on a VIS. Three different storage environments are considered: cloud alone, P2P storage on top of desktops, a hybrid storage, and their availability, cost, and privacy trade-offs were studied. In the desktop-only storage model, a *socially-informed replication scheme* was proposed, where a user replicates his content to his friend nodes and delegates access control to them. However, normally, a uses trusts only a fraction of his friends to the extent of delegating access control enforcement, as considered in our *My3* approach along with online time information. Our earlier work [26] considered access control delegation in P2P systems in terms of trust transitivity.

Tribler [27] is a P2P file sharing application which exploits friendship relationships, tastes and preferences of users to increase the performance of file sharing. However, in Tribler, users host their own profile and therefore profile placement for high availability and low access or consistency cost are not considered. Finally, LifeSocial [28] is a P2P-hosted OSN where users employ public-private key pairs to encrypt profile data that is stored in a distributed way and is indexed in a DHT. Friends can read a user's profile based on a symmetric key that is encrypted with their public keys. However, data privacy and profile availability are not considered in [28].

The authors in [29] pursue the notion of online times for a P2P client in detail. Various replica placement strategies are studied analytically. The Diaspora [30] project aims to build a user- owned decentralized online social network. It consists of independently owned *pods* (or servers) which host user profiles and form the network. However, the Diaspora system needs the pods to be online always. We believe that the *My3* model for decentralized OSNs where users can run their clients for a fraction of time compared to always-on availability of Diaspora, is more amenable.

To the best of our knowledge, *My3* is the first system that uniquely identifies the availability of decentralized OSNs as the critical concern for their adaptability and considers user behavior characteristics on OSNs and exploit them in its design.

VIII. Conclusion and Future Work

In this paper, we presented the design of My3, a privacy-preserving decentralized OSN. We evaluated its performance regarding different evaluation criteria using real world data traces. As experimentally found, high availability is achievable with a profile replication factor of 4-5. Moreover, by employing the hybrid replication algorithm, a combination of performance objectives can be met and thus the My3 system could be a viable decentralized alternative to centralized infrastructures. Our system also involves dealing with access control policies, trusted identity management and data integrity management issues; we leave these for future work. Moreover, we plan to explore system fairness aspects with respect to storage overhead and richer online time models especially considering the user degrees in the OSNs given that, a high degree node stays online longer.

Acknowledgment

This work was partially funded by the grant *Reconcile: Robust Online Credibility Evaluation of Web Content* from Switzerland through the Swiss contribution to the EU.

References

[1] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proc. of the 7th Internet measurements conference*, 2007.

[2] S. Guha, K. Tang, and P. Francis, "Noyb: privacy in online social networks," in *Proc. of the WOSP*, Seattle, WA, USA, 2008.

[3] B. Krishnamurthy and C. E. Wills, "On the leakage of personally identifiable information via online social networks," in *Proc. of the WOSN*, 2009.

[4] N. Rammohan, T. Papaioannou, and K. Aberer, "Privacy-aware and highly-available osn profiles," in *Proc. of Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, 2010.

[5] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in *Proc. of the IMC*, 2009.

[6] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding online social network usage from a network perspective," in *Proc. of the IMC*, New York, NY, USA, 2009.

[7] "Ip latency statistics," *http://www.verizonbusiness.com/about/network/latency/*.

[8] L. A. Cutillo, R. Molva, and T. Strufe, "Privacy preserving social networking through decentralization," in *Proc. of the WONS*, 2009.

[9] C. Fung, "Collaborative intrusion detection networks and insider attacks," vol. 2(1), pp. 63 – 74, 2011.

[10] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and K.-I. Ko, "A greedy approximation for minimum connected dominating sets," *Theoretical Computer Science*, vol. 329, no. 1-3, pp. 325 – 330, 2004.

[11] N. Rammohan, T. Papaioannou, and K. Aberer, "Towards the realization of decentralized onlinesocial networks: an empirical study," in *Proc. of the ICDCS Workshops: HOTPOST*, 2012.

[12] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proc. of the WOSN*, 2009.

[13] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer, "Outtweeting the Twitterers - Predicting Information Cascades in Microblogs," in *Proc. of the WOSN*, 2010.

[14] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer voip system," 2006.

[15] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards ip geolocation using delay and topology measurements," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 71–84. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177090

[16] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove, "Analyzing facebook privacy settings: user expectations vs. reality," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. Proc. of the IMC. New York, NY, USA: ACM, 2011, pp. 61–70.

[17] M. Madejski, M. Johnson, and S. M. Bellovin, "The failure of online social network privacy settings," Department of Computer Science, Columbia University, Tech. Rep. CUCS-010-11, February 2011. [Online]. Available: https://mice.cs.columbia.edu/getTechreport.php?techreportID=1459

[18] I.-F. Lam, K.-T. Chen, and L.-J. Chen, "Involuntary information leakage in social network services," in *Proc. of the 3rd International Workshop on Security*, 2008.

[19] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman, "Lockr: better privacy for social networks," in *Proc. of the CoNEXT*, 2009.

[20] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: an online social network with user-defined privacy," in *Proc. of the ACM SIGCOMM*, 2009.

[21] M. M. Lucas and N. Borisov, "Flybynight: mitigating the privacy risks of social networking," in *Proc. of the WPES*, 2008.

[22] A. Shakimov, A. Varshavsky, L. P. Cox, and R. Cáceres, "Privacy, cost, and availability tradeoffs in decentralized osns," in *Proc. of the WOSN*, 2009.

[23] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "Peerson: P2p social networking: early experiences and insights," in *Proc. of the ACM EuroSys Workshop on Social Network Systems*, 2009.

[24] M. Marcon, B. Viswanath, M. Cha, and K. P. Gummadi, "Sharing social content from home: a measurement-driven feasibility study," in *Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video*, ser. NOSSDAV '11. New York, NY, USA: ACM, 2011.

[25] J. Sonia, N. Shirin, M. Prateek, B. Nikita, and K. Apu, "Decent: A decentralized architecture for enforcing privacy in online social networks," in *Proc. of the SESOC*, 2012.

[26] N. Rammohan, Z. Miklos, and K. Aberer, "Towards access control aware p2p data management systems," in *Proc. of the 2nd International workshop on data management in peer-to-peer systems*, 2009.

[27] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, "Tribler: a social-based peer-to-peer system: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 20, no. 2, pp. 127–138, 2008.

[28] K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, and R. Steinmetz, "Practical security in p2p-based social networks," in *Proc. of the IEEE LCN*, October 2009.

[29] K. Rzadca, A. Datta, and S. Buchegger, "Replica placement in p2p storage: Complexity and game theoretic analyses," in *Proc. of the ICDCS*, June 2010.

[30] *http://diasporaproject.org/*.