DiDuSoNet: A P2P architecture for distributed Dunbar-based social networks

Barbara Guidi[1,2] · Tobias Amft[3] · Andrea De Salve[1,2] · Kalman Graffi[3] · Laura Ricci[1]

Abstract Online Social Networks (OSNs) are becoming more and more popular on the Web. Distributed Online Social Networks (DOSNs) are OSNs which do not exploit a central server for storing users data and enable users to have more control on their profile content, ensuring a higher level of privacy. In a DOSN there are some technical challenges to face. One of the most important challenges is the data availability problem when a user is offline. In this paper we propose DiDuSoNet, a novel P2P Distributed Online Social Network where users can exercise full access control on their data. Our system exploits trust relationships for providing a set of important social services, such as trustness, information diffusion, and data availability. In this paper we show how our system manages the problem of data availability by proposing a new P2P dynamic trusted storage approach. By following the Dunbar concept, our system stores the data of a user only on a restricted number of friends which have regular contacts with him/her. Differently from other approaches, nodes chosen to keep data replicas are not statically defined but dynamically change according to users churn. In according to our previous work, we use only two online profile replicas at time. By using real Facebook data traces we prove that our approach offers high availability.

Keywords Distributed Online Social Networks · P2P · Data availability · Dunbar

✉ Barbara Guidi
guidi@di.unipi.it

Tobias Amft
amft@cs.uni-duesseldorf.de

Andrea De Salve
desalve@di.unipi.it

Kalman Graffi
graffi@cs.uni-duesseldorf.de

Laura Ricci
ricci@di.unipi.it

[1] Department of Computer Science, University of Pisa, Largo B. Pontecorvo, 56127, Pisa, Italy

[2] IIT-CNR, via G. Moruzzi, 1 56124 Pisa, Italy

[3] Technology of Social Networks Group Department of Computer Science, University of Düsseldorf, Düsseldorf, Germany

1 Introduction

In the last few years Online Social Networks (OSNs) have achieved unprecedented success, changing the way of how people interact with each other and becoming storehouses of huge amount of data in the form of messages, photos and personal information. Once users have published their personal information within a centralized online social network, they lose the control over their data and they are under the risk of privacy breaches and malicious exploitation. A current trend for developing OSN services is towards the distribution of the social network infrastructure, often through a P2P approach. A Distributed Online Social Network (DOSN) [1] is an OSN implemented on a distributed information management platform, such as a network of trusted servers, P2P systems or opportunistic networks.

In a P2P Distributed Online Social Network there is no single provider but a set of peers that take on and share

the tasks needed to run the system. The development of the existing functionalities of OSNs in a distributed context requires finding ways for providing robustness against churn, distributing storage of data, propagating updates, defining an overlay topology and a protocol enabling searching and addressing, etc. One of the main challenge comes from guaranteeing the data persistence of a user data even when he is offline. P2P networks are dynamic, therefore users' data could become unavailable due to churn.

Some DOSNs rely on external storage systems, for example exploiting a distributed file system [2], while some other more recent approaches ([3–5]) propose to store social data of a user on the storage support provided by users' friends. In these proposals, the data owner serves his own data when he is online, and elects a subset of his friends to make the data available when he is offline. The two well-known techniques that are used to manage the problem of data availability in a DOSNs are *Distributed Caching* and *Dynamic Data Replication*. In particular, the replication technique has been widely used in existing DOSNs. One of the problems of the replication is regarding to the number of replicas and the data consistence.

The focus of the paper is to present DiDuSoNet (Distributed Dunbar-based Social Networks), a novel P2P Distributed Online Social Network in which the Dunbar's approach is used to manage a set of social services and in particular the data availability problem. Dunbar has found that maintaining social relationships is costly in terms of cognitive capabilities. These cognitive constraints impose an upper limit on the maximum number of relationships an ego can actively maintain. This limit is about 150 and it is known as the *Dunbar's number* [6–8]. By introducing a Trusted Social Storage Approach, our goal is to guarantee that there is, with high probability, always a copy of each profile available on at least one online node in the network, which is in the following called Point of Storage (*PoS*).

We have evaluated the system through a set of simulations conducted on real Facebook traces which show that a user has almost always at least one online friend. These experimental results show the soundness of our approach. Henceforth, we use the terms node, user, and peer interchangeably.

The rest of this paper is structured as follows: Section 2 shows the related work; Section 3 introduces the System Model and it provides a detailed overview of the system. Section 4 presents the Trusted Social Storage Approach which our system provides and an analysis of how our system manages the problem of the dynamism of the storage nodes. In Section 5 we show the strategy used to elect the trusted social storage, and in Section 6 the algorithms. In Section 7 we present the simulation methodology and we show information about the real Facebook dataset used for our simulation analysis. In Section 8 we show the experimental results and finally, in Section 9 we present the conclusions.

2 Related work

This section discusses the related work and presents:

1. an overview of existing DOSNs.
2. an overview of the existing studies on data availability in DOSNs.

2.1 Distributed online social networks

Recently, several P2P architectures for DOSNs have been proposed. LifeSocial [9] is a P2P-hosted OSN where users employ public-private key pairs to encrypt profile data that are securely stored in a FreePastry-based DHT. PeerSon [10] is a distributed infrastructure for social networks whose focus is related to security and privacy concerns. It exploits encryption, direct data exchange and access control. Safebook [11] addresses privacy in OSNs by storing encrypted profile content in a P2P storage infrastructure.

Most of DOSNs rely on servers or other external services to guarantee data availability.

Diaspora [12] is a social network in which users' profiles are hosted on servers that are administrated by individual users who decide themselves where their information will be stored. SuperNova [13] is a DOSN that manages the availability by using a super-peer architecture that provides highly available storage.

Others DOSNs built on Friend-to-Friend (F2F) networks do not require any complex encryption mechanism for data management. In [14] the authors argued that a user should choose the nodes with whom it shares data based on existing social links instead of choosing them at random.

2.2 Data availability in DOSNs

Important research efforts have been focused on studying the problem of data availability in DOSNs.

Friendstore [10] is a social back-up system that ensures availability by storing data on trusted nodes only. In [3], the authors propose a replication strategy in a DOSN in which replicas of users' profiles are stored only on a set of trusted proxies. Furthermore, different replica-placement techniques to guarantee data availability are proposed in [3] and [4]. Instead of replicas in [15] authors propose to store data blocks on nodes by considering their availability. Papers [16] and [17] present new techniques to predict users' availability based on their historical behaviour and authors in [18] demonstrate that the online presence of OSN users tend to be correlated to those of their friends.

Only a few studies have been made on data availability in F2F storage systems: in [19], authors show that F2F systems cannot guarantee high data availability for users with a few friends online and [5] focuses on the impact of availability correlation and very small friend sets on F2F systems' data availability. Furthermore, Li and Dabek [20] argued that a node should choose its neighbours where the data are stored based on existing social relationships instead of randomly.

3 The system model

DiDuSoNet is a two-tier level system, as shown in Fig. 1. It grounds on a Dunbar-based P2P Social Overlay where the connections between nodes correspond to the social relations of the Dunbar-based ego networks of the users. Users are able to communicate with each other directly via point to point connections through the Dunbar-based P2P Social Overlay (for sending private messages or private data). The Social Overlay provides a set of Social Services to manage trustness, information diffusion, and data availability. The trustness is provided from our Dunbar-based Social Overlay, where each node is connected to a set of trusted nodes. The information diffusion issue is managed as shown in [21]. Instead, the data availability issue is introduced in our previous work [22] and it is explained in whole in this paper. Two problems can be observed:

- contact information, namely the IP address, changes during the time. Therefore it is not recommendable to store contact information about friends only if they enter the network.
- if a node enters the network it has no contact information which could be used to establish an initial connection to any friend (bootstrapping). Even if a joining node might know about the contact information of one friend, it might be not possible to determine the IP addresses of all friends, for example if the first contacted node does not share these friends with the joining node.

We exploit a DHT [23] for managing these issues, therefore the problem of searching new friends, and for helping the data availability management.

To assist the reader, Table 1 summarizes all the symbols used in this paper.

In a DOSN each user is associated with a $profile$, which is a personal web page where each user freely posts content – e.g. text, snippets, pictures, videos and music. In response to these postings other users, usually friends, post comments and other content. Our system is profile-based. In modern OSNs communication is said to be *profile-based*, since around 90 % of server requests are related to profile

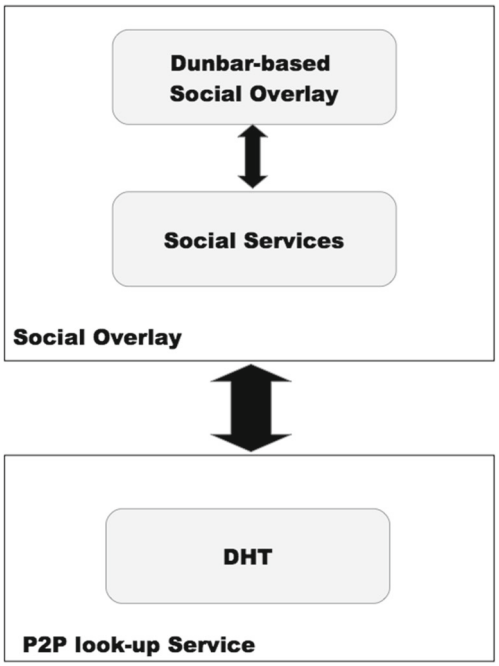


Fig. 1 The Framework

page content [24]. Profiles generally have *small* dimension, since pretty much everything in the profile pages – comments, links, thumbnails, small pictures – are small objects. The only exception are movies and large collections of high-quality pictures: however, these are usually not part of profiles anyway, and are linked from services such as YouTube and Flickr [25].

We consider that each node in the network has its profile and the profile contains public and private data. Which data are public or not is a user's choice. Users can only access to friends' private data through the Dunbar-based Ego Network. Therefore, private data are only stored in a controlled manner on well-known nodes. On the other hand, we store information in the DHT which enables each node which is interested in a certain profile to search for the responsible node which maintains either the data itself or a valid replica. Each user is identified by two different IDs: the first one is the *SocialID* and it is used to identify a node in the system; the second one is the ID related to the underlying DHT which is used to navigate within the DHT. We can assume here that two persons which are friends in our system know each other and know about the *SocialID* of each other.

For the data availability problem, data objects of a specific node n are copied and stored on elected nodes chosen from the set of friendly nodes. These nodes are called Point Of Storage (PoS). In case that a generic node n leaves the network, its friends have to find out which node is PoS for this node and has therefore a copy of n's data. Since the PoSs might change over time (due to churn), it is important

Table 1 Table of symbols used in this paper

Table of Symbols	
$PoS(u)$	A node which is a Point Of Storage for the node u
$PoS^{-1}(u)$	Set of nodes that have node u as Point Of Storage
$SS_{ego_n}(u)$	The Social Score assigned to a node u, which is a neighbour of the node n
$EN(u)$	Ego Network of node u
V_u	Nodes contained into the ego network of u and u
E_u	Edges contained into the ego network of u
$TieStrength_{nu}$	the tie strength from the node n to the node u
$CG_n(u)$	the gain in term of trusted connection obtained by the election of u as $PoS(n)$
$MSL(u)$	the medium session length of the node u
$Prof(u)$	the profile of the node u
$df(u)$	set of nodes that are included in the Dunbar-based ego network of u
$ef(u)$	set of nodes that are included in the ego network of u

that all friends of n are able to find n's PoSs. We use the DHT to maintain the knowledge about the PoSs: for every node n we store a list of its PoSs into the DHT. Whenever one PoS of n changes, the list is updated inside the DHT.

3.1 Dunbar-based social overlay

We refer to the concept of Social Overlay (SO) as introduced in [25]. In a SO nodes of a P2P system are only connected to each other if their owners are friends. Our intuition is to use a Social Overlay which is based on a well-known social concept: the Dunbar's approach [6–8]. Authors show that the number of friends a user can maintain stable social connections with is approximately 150. This limit is known as *Dunbar's number*. The stability of a connection may be defined as a function of the *tie strength* which characterizes the relation and it is a numeric value that quantifies the strength of the relationship between two users.

In OSNs, the tie strength can be calculated by taking into account different factors such as contact frequency between the two users, the number of likes, posts, comments, private messages, tags, etc. [26]. Everett and Borgatti in [27] define ego networks to be networks consisting of a single actor (ego) together with the actors they are connected to ($alters$) and all the links among those alters. Thus we can formally define the ego network of a user u as $EN(u) = (V_u, E_u)$, where $V_u = \{u\} \cup \{v \in V \mid (u, v) \in E\} \wedge E_u = \{(a, b) \in E \mid a = u \vee b = u \vee \{a, b\} \subseteq V_u\}$. The ego network of a user can be seen as a subset of G which represents the local view of the user.

The size of an ego network may be large, for instance the maximum amount of relationships a user can establish is generally very high (5000 connections in Facebook). As explained in our previous work [22], we use the Dunbar's number to create a dynamic social overlay where each user is connected only to a limited set of friends arranged in a hierarchical inclusive sequence ordered by increasing level of intimacy [8]. This permits us to reduce the total amount of social information each peer has to keep in memory and the number of connections of the P2P overlay. This ego network will be referred as *Dunbar-based ego network*.

Our resulting Social Overlay is an unstructured P2P Overlay Network and it is composed by the Dunbar-based Ego Networks of each node in the network.

3.2 P2P look-up service: DHT

The DHT provides the functionality to find nodes to a given ID (*SocialID*) and to store, search, and retrieve data objects in a distributed fashion. Another advantage of using the DHT is its robustness against churn. Further, we assume that a node n which joins the network knows at least about the *SocialID* of its friends. To ascertain the Overlay ID of a node in the DHT we simply hash the *SocialID*. The procedure of searching friends inside the DHT is the following now:

- We hash the *SocialID* of the friend we want to find.
- Obtaining the hash value, we can search for the node which is responsible for this value.
- The node which is found to be responsible and has the same Overlay ID as the hash value is the friend we are looking for. If no node with the same Overlay ID exists in the DHT, the friend is not online and the look up returns a reference to the node which stores the data of the friend. Note that, additionally, public key authorization can be used to verify the identity of a node in the network.

In the bootstrap phase, each node n performs a look up on the DHT to retrieve its Dunbar friends. For each friend

which is on-line, the current physical address is returned, so that n may open a connection to that friend and it can build the Dunbar-based Social Overlay. If the friend is not online, the node which is responsible for the data of n is retrieved by the look-up.

On top of the DHT we implemented a data availability service which autonomously selects two nodes in an ego network as points of storages for each profile that is published. By this means we need a mechanism which allows a node to search for profiles in the network. We utilize the features of a DHT which allow to search a value for a given key to realize the search of a profile in a distributed P2P Overly. For this reason we introduce the point of storage list that is referred as (*PoS Table*), in which the Overlay IDs of all PoSs for a given Social ID are stored. For every ego node which publishes private data for its friends such a PoS Table is stored in the DHT.

Whenever a node searches for a published profile of any other node it simply has to start a lookup to the PoS Table of the profiles owner. If the PoS Table is obtained from the DHT the searching node gets in contact with one of the PoSs from the list and requests the desired data (*profile*). Again, additional public key authorization can be used to prevent data to be accessed by non friendly nodes.

The costs of spreading the knowledge about the PoSs and of searching for PoSs is $O(1)$. Only $O(logN)$ messages are needed in most DHTs (Pastry, Chord) to store and retrieve data objects. Data, which are stored inside the DHT, are available to all nodes that participate in the network. Contrary to that, data which are stored inside the Dunbar-based Social Overlay are private and available only for friends. Private data are stored only on the owner of the data and are replicated to trusted nodes (PoSs). Due to churn it might happen that the owner of the data is not available so that two selected friends of the owner act as PoSs. Furthermore, each PoS of n maintains a list of n's friends so that it will be able to identify nodes which request n's data and will be able to select other PoSs in case they want to leave the network, too. In case a node wants to hide the Overlay IDs of its PoSs to prevent other non friendly nodes from spying, they can use attribute-based encryption (*ABE*) schemes [28] or ciphertext-policy attribute-based encryption (*CP-ABE*) schemes [29] to delimit the access of the PoSs list to selected friends only.

For our simulation we use Pastry as underlying DHT, which is a P2P overlay network with good performance and high reliability [30]. It is completely decentralized, scalable, and self-organizing. Nevertheless, any other DHT could be used for our proposed scheme as long as its routing mechanism is deterministic, as for example in Chord [31].

4 Trusted social storage approach

In our Trusted Social Storage Approach, each user dynamically elects a subset of his Dunbar friends, which we will refer as Points of Storages (PoS), to store a replica of his social data. We define *Prof(u)* as the profile of a node u, *PoS(u)* as the set of Point of Storages of node u and $PoS^{-1}(u) = \{n | u \in PoS(n)\}$ as the set of nodes for which u is Point of Storage. When a node n is online, it is a PoS for itself and it needs to elect another PoS to guarantee the two online copies of its profile. The node has to send a copy of all its data objects to the PoS via point-to-point connection by using the Dunbar-based Social Overlay, as well as a list of friends that belong to n's ego network. Additionally, both nodes exchange ping-pong messages frequently to detect node failures (sudden departures). Friends nodes can download data objects (e.g. profile) directly from one of both nodes. When a user u disconnects from the system with a notification, it executes the following two steps:

- it elects a second PoS for itself by considering its Dunbar's friends,
- it exploits $PoS^{-1}(u)$ to elect a new PoS for each node n in $PoS^{-1}(u)$ that is offline at that moment (if a node is online is able to elect another PoS by itself). To implement the election of a new PoS for n, u must also know the list of *all* friends of n and all the social information about them required to implement the PoS selection strategy, that we will describe in Section 5, for instance the tie strength and the average session length of these nodes.

When a node is elected as PoS for some user u, its *SocialID* is added in user u's entry list in the PoS Table. When it leaves the network, it deletes its entry in the PoS Table of u, but it keeps a copy of the *Prof(u)* until its reconnection. When it reconnects to the system, it has to check if there is at least one online PoS for the node u. In case it finds one online PoS, it destroys its local copy of *Prof(u)*, otherwise it elects itself as PoS of the node u and an old version of the *Prof(u)* will be available.

Through the PoS Table, each user always knows the current PoS of his friends and can access the newest version of their profiles currently online. We assume there is an authentication mechanism that allows the access to the user u's PoS information only to his friends.

Friends of u might exchange his data, but if they are not mutual friends themselves they cannot host each other's data. We assume user u's knowledge of the social network corresponds to his Dunbar-based ego network, which is the set $df(u)$, the state (online/offline) of each user $\in df(u)$, and the tie strength of each link in $\{(x, y) \mid x, y \in \{u\} \cup df(u)\}$. A user u also knows the set of nodes n in $PoS^{-1}(u) = \{n \mid u \in PoS(n)\}$.

Another challenge of our approach is the robustness against single-node failure. A PoS may voluntarily disconnect suddenly or because of different types of failure without being able to pass a replica of its data to another node. To address this goal we keep two online copies of each user's profile at any time: an online node n must have one online PoS so that a copy is still present in the system also in case of failure of n or of $PoS(n)$. For the same reason an offline node with more than one online friend must have two online PoSs.

5 PoS selection strategy

The choice of friends where to store replicas of users' profiles is a fundamental issue in our system. We consider social and structural properties of the social graph, and the user's behaviour to define our approach. We assign to each node a score, called *Social Score* (SS), measuring the users' suitability as PoS of a specific node. The *Social Score* of a user x which is a Dunbar neighbour of a user n is defined as follows:

$$SS_{ego_n}(x) = (\alpha \cdot TieStrength_{nx}) \cdot (\beta \cdot CG_n(x)) \cdot (\gamma \cdot MSL(x))$$

where all the terms $TieStrength_{nx}$, $CG_n(x)$ and $MSL(x)$ are normalized and α, β and γ are weights autonomously chosen by the users according to the relevance given to each criteria.

The $ConnectionGain$ ($CG_n(x)$) for an online node x in the ego network of a user n is the gain in terms of trusted connections obtained by the election of x as $PoS(n)$ and is proportional to the number of common neighbours of x and n. If $CG_n(x)$ has a high value, then x may transfer the profile of n to neighbours of n through a trusted connection.

$MSL(x)$ is the *Medium Session Length* and represents the average duration of a user session. We approximate this measure so that it is able to distinguish at least between users almost always connected to the social network (e.g. through mobile devices), those that connect only for short periods of time and those that use the social network as a means of social interaction and therefore have quite long sessions on average. We will name *Fair Strategy* the strategy that gives to all the weights the value 1.

5.1 Dynamism and points of storage (PoS)

In this section we explain our data availability service protocol to manage the problem of dynamism (due to churn) from the point of view of a PoS. As long as there is at least one online friend for each node the availability can be always satisfied but if no friend is online, then data stored in the system will not be accessible by any means. As long as no friend of the user is online that is a minor turn-off, since no one is interested in accessing the user's profile. But if a friend of the user reconnects to the system and does not keep in memory a copy of the user's profile he cannot access the user's data until one PoS of the user or the user himself reconnects to the system. We have to consider that in most of the famous OSNs, users usually don't disconnect from the system through a log-out phase and this represents a hard challenge to face for the problem of data availability.

Let consider the ego network of the node n shown in Fig. 2, in which there are just selected the 2 PoSs for n, which are itself and the friend node A. We explain the three scenarios managed by our system:

Case 1 (**Voluntarily disconnection of a PoS**) When the node A is going to disconnect from the system, it sends a notification to the other PoS, in this case the node n, and n is the responsible for the election of the second PoS, the node B as shown in Fig. 3. Nodes n and B exchange ping-pong messages now. The changes are also stored in the PoS table.

We have the same behaviour even if is n the PoS that wants to leave the system. The difference is that the disconnecting PoS is the owner of the profile (n), so it elects the second PoS for itself.

Case 2 (**Involuntary disconnection of a PoS**) This happens when a PoS leaves the network suddenly without notification about its departure. In Fig. 4a node n leaves the network suddenly. B is able to detect this failure since n will not respond to incoming ping messages. Nodes which request data from n will be informed about its absence and can still ask node B for the required data. Furthermore, node B is able to select a second PoS (node C as shown in Fig. 4b)

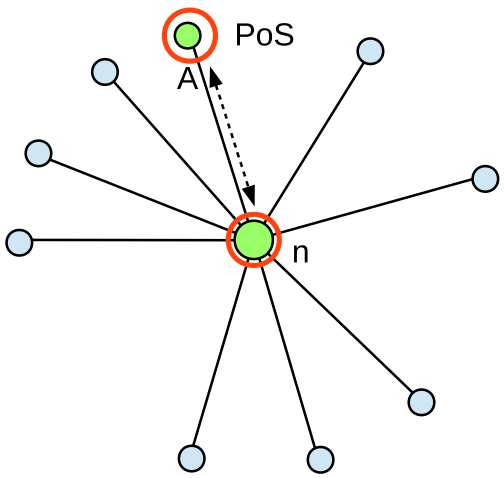


Fig. 2 Ego Network of Node n

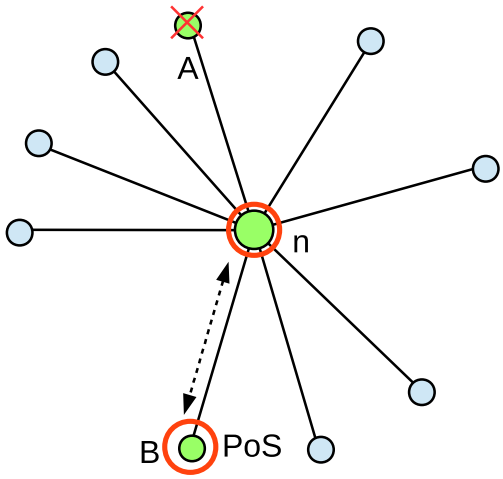


Fig. 3 Case 1: Ego network of the node n after the disconnection of A

in the ego network of n. Similar to **case 1**, node B sends a copy of n's data to node C which is second PoS now. Keep-alive messages are exchanged between nodes B and C.

Case 3 (**Involuntary disconnection of both PoSs**) This happens when both PoSs leave the network involuntarily at the same time (e.g. due to failures in the network) as shown in Fig. 5a. None of the two PoSs is able to inform the network about its departure. Nodes which request n's data will find the PoS Table but will also detect both PoSs to be unavailable. In this case, no data is available in the network.

If during this time node A joins the network again which has been PoS before and has a copy of n's data, it searches for the current PoSs (which are still listed in the PoS Table) to check if there is an online copy of the n's profile and to delete its old copy. A will detect the absence of any PoS and it will be the only node which still has a copy of n's data (there might be other friends which have a copy, but do not know that both PoSs are down). A can now elect itself as PoS of n and update the PoS Table.

As a next step node A selects a second PoS and updates the PoS Table again (Fig. 5b). D, which is the new PoS for n, will receive a copy of n's data and a list of n's friends.

If node n joins the network again it knows about the current PoSs by accessing the PoS Table. In according to our model, when n is online, it has to be a PoS for itself, so n will decide which node will be the second PoS between A and D.

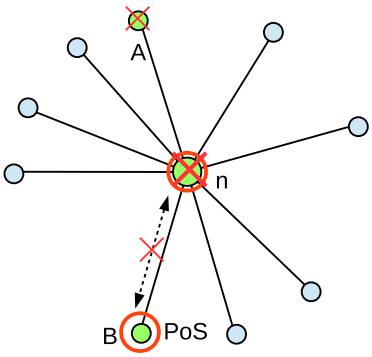
5.2 Data consistence

In this section, we discuss about the process of updating data objects to manage the consistence of data. In according to situation explained in the case 3, if node n creates a new data object or changes existing data, it informs the second PoSs about this (the node A as shown in Fig. 6).

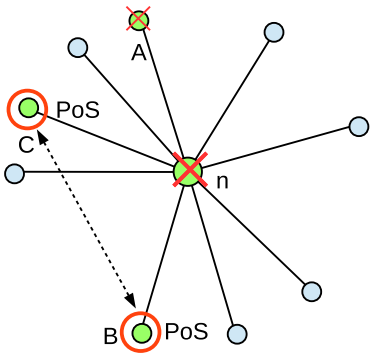
In Fig. 6 can be seen that the node A sends the updated data to n that is online. The node D should delete its n's profile copy because there are just other online copies of this profile.

Now consider **case 3** again in which both PoS fail (Fig. 7a). If a node which has been PoS before detects the absence of both PoSs (Fig. 7b) it will select a second PoS immediately (Fig. 7c) and elects the second PoS between the online friend nodes of n.

When node n is joining the network again and detects old data objects on both PoSs, it sends copies of the new data objects to the PoSs.

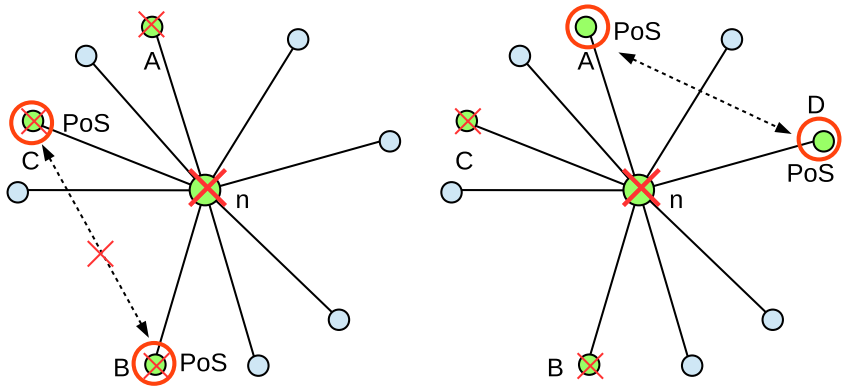


(a) Ego Network of Node n after its departure.



(b) Ego Network of Node n and its two PoSs after its departure.

Fig. 4 Case 2



(a) Ego Network of Node n after a case of involuntary disconnection of its PoSs B and C.

(b) Ego Network of Node n: reconnection of its old PoS A after a involuntarily disconnection of both B and C.

Fig. 5 Case 3

Afterwards, n can decide which of C and D will be the second PoS and it updates the PoS Table.

When the owner of a profile, the node n in the previous case, is joining the network again and there are just two online PoSs in the network, it has to decide which one is the most suitable to be the second PoS. This choice is done by considering the Social Score value. n computes and evaluates which of the two nodes has the highest SS. It notifies to the unselected node that is not a PoS yet and it updates the PoS Table.

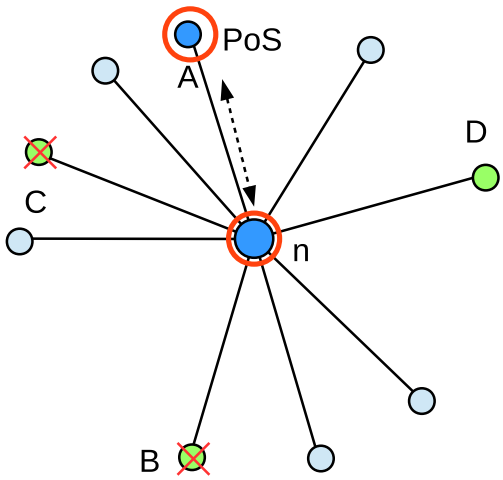


Fig. 6 Ego Network of Node n after the reconnection of n. D is not a PoS and the updated data are stored on A and n

6 PoS election algorithms

In this section we describe the algorithms for the election of the PoS which must guarantee that each online node has one PoS among its online friends and that each offline node with more than one online friend has two PoS among them. The three possible scenarios in which the algorithm is executed are:

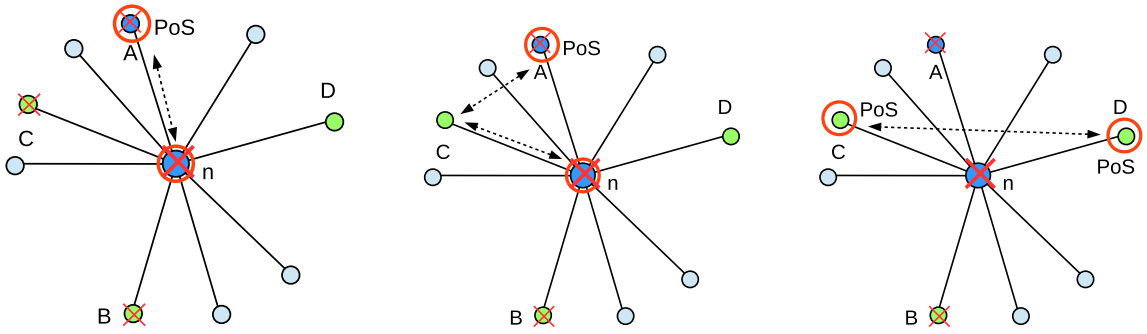
- when a user u is going to disconnect from the system;
- when a user u is going to disconnect from the system and it is a PoS for at least one node in the system;
- when a user u does disconnect from the system without a notification (e.g. crash).

First, we define an auxiliary procedure `SelectBestPos()` that, given user p and a given set of nodes Set, selects the best PoS for p among them.

Algorithm 1 Best PoS Selection

```
1: function SELECTBESTPOS(p, Set)
2:     if (∃x ∈ Set | x.isOnline()) then
3:         get SS_ego_p(v) ∀v ∈ Set | v.isOnline();
4:         select n | SS_ego_p(n) = max{SS_ego_p};
5:         return n;
6:     else
7:         return null;
8:     end if
9: end function
```

Algorithm 1 gets $SS_{ego_p}(v)$ for all online nodes in the set and elects as the new $PoS(p)$ the node with the highest



(a) Ego Network of Node n after both PoSs fail.

(b) Reconnection of the old PoS C.

(c) C checks if at least one PoS is online. It detects that both PoSs are offline, it elect itself as PoS and elects D as second PoS.

Fig. 7 PoS dynamism and Data Consistence

Social Score. If all nodes in the set are offline the algorithm returns null, otherwise the selected node is returned.

Algorithm 2 PoS election for a node p

```
1:  function ELECTION(p)
2:      Set = df(p) − PoS(p);
3:      n = SelectBestPoS(p, Set);
4:      if n ≠ null then
5:          send < Prof(p),ef(p), TieStrength_p* > to n;
6:          update PoSTable(p);
7:      else
8:          Set = ef(p) − PoS(p);
9:          n = SelectBestPoS(p, Set);
10:         if n ≠ null then
11:             send < Prof(p),ef(p), TieStrength_p* > to n
12:             update PoSTable(p);
13:         end if
14:     end if
15: end function
```

Algorithm 2 shows the procedure to elect a new PoS for a node p. The best PoS is chosen among *df(p)* at first and among *ef(p)* if all nodes in *df(p)* are offline (we recall the *df(p)* is the set of the Dunbar friend of a node p, while *ef(p)* denotes the set of all ego friends of a node p. Selected PoS receives from p *Prof(p)*, *df(p)* and all the tie strength values between p and one of its friends ($TieStrength_{p*}$). Afterwards, p's entry in the $PoS\ Table$ is updated with the information about p's new PoS. Algorithm 2 is executed when an online node p receives a disconnection notification from a node that is its only current $PoS(p)$ and when a node is going to disconnect from the system and needs to elect its second PoS.

Let us now consider the procedure executed by a node n which is going to disconnect from the system with notification and which is the PoS for at least a user. n notifies its disconnection to online nodes that belong to $Pos^{-1}(n)$ and these nodes will be able to elect a new PoS on their own. As far as concerns offline nodes $\in Pos^{-1}(n)$, n must select a new PoS for each of them among their neighbours. Notice that n received the list *ef(p)* at the moment of its election as *PoS(p)*. As for the election procedure the best PoS is the node with the highest Social Score. Algorithm 3 shows the procedure previously described.

Algorithm 3 PoS election for an offline node executed by a disconnecting node

```
1: function ONDISCONNECT(p)
2:     for all u ∈ Pos^-1(p) | !(u.isOnline()) do
3:         Election(u);
4:     end for
5: end function
```

7 Simulation methodology

The lack of data concerning online presence of users in OSNs is currently the main limitation for the data availability management. As far as concerns the study of the users' behaviour, most studies utilize synthetic users' availability traces, and in the context of distributed systems, some studies used synthetic availability traces provided by P2P well-known churn models. However, the scientists realized that these traces were not useful since the nature of the P2P systems compared to OSN is very different [32]. At the best of our knowledge, no existing up-to-date dataset is able to provide complete information, such as information

regarding the social graph, interactions among users and temporal information (online sessions) for a real OSN.

7.1 The Facebook'14 data set

We have implemented a Facebook application, called *SocialCircles!*, which exploits the former Facebook API (applications exploiting this API will be supported till 1st May 2015). The application is able to retrieve the following sets of information from registered users:

Topology and Profile Information We are able to obtain friends of registered users and the friendship relations existing between them. Moreover we download profile information of registered users and their friends, such as *complete name*, *birthday*, *sex*, *location*, *works*, *schools*, *user devices*, *movies*, *music*, *book*, *interest* and *language*.

Interaction Information We have collected information about interactions between users registered to the application and their friends, such as *posts*, *comments*, *likes*, *tags* and *photo*. Due to technical reasons (time needed to fetch all data and storage capacity), we restrict the interaction information retrieved up to 6 months prior to user application registration.

Online presence data By requesting the online presence permission, we are able to monitor the *chat presence status* and obtain information about the time spent online by registered users and their friends. The chat status can assume a limited set of value: 0 if user is *offine*, 1 if user is in *active state* and 2 if user is *idle* (i.e. the user is online but they have not performed actions for more than 10 minutes).

The dataset obtained from the *SocialCircles!* application contains 337 complete ego networks related to the registered users, for a total of 144.481 users (ego and their alters). Since few users de-authorized our application, we were able to retrieve complete interactions information of 328 users. The resulting Facebook population have the advantage of representing a very heterogeneous population: 213 males and 115 females, with age range of 15-79 with different education, background and geographically location. Only 308 users were active in the period of crawling. The characteristics of the ego network are: *Minimum size* = 21, *Maximum Size*= 2932, *Mean Size*= 470.916, and *STD Deviation*= 358.038.

The contact frequency between each couple of friends was computed by dividing the number of interactions between an ego and the alter by the duration of their social relationship. Since Facebook does not allow us to acquire information related to the start time of a social relationship, we estimated the duration of the social relationship as the time elapsed between the first interaction between the users and the end time of crawling (in hours). Since no timestamp is paired with some interactions (such as Tags and Likes), we approximated the correspondent friendships' duration with the overall duration of the period considered by the crawling process (six months). Then, we exploited contact frequencies to calculate the tie strength from a user j to a user k, as proposed in [33].

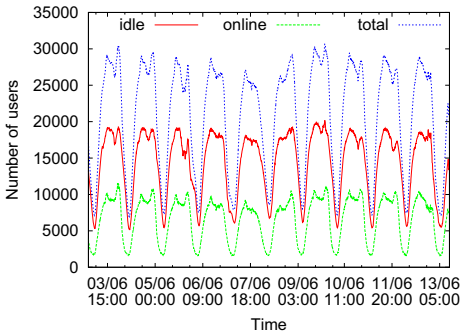
Network statistics By analysing the dataset, we discovered that the median Facebook network has about 390 friends totally. We sampled all the 337 registered egos and their friends every 8 minutes for 10 days (from Tuesday 3 June 2014 to Friday 13 June 2014). Using this methodology we were able to access 308 active registered users and the set of their friends (for a total of 95.578 users). In order to characterize OSN workloads at the session level we consider the availability trace of each user to determine the start of a session (when a user switches from offline to online or idle) or the end of a session (when a user switches from online or idle to offline). Furthermore, we examined the number of concurrent users that accessed the OSN site (see Fig. 8a). The plot indicates clearly the presence of a cyclic day/night pattern (confirming the results in [34]). Since the majority of the registered users live in Italy or in central Europe, time-zone differences are negligible.

The analysis of graph depicts the presence of two peaks: on average, most users seem to be connected after lunch time with a peak around 14:30. The other peak is usually in the evening, around 22:30, probably preceding the sleeping time. It is interesting to notice that the presence of weekend seems to have influence on users: Friday and Saturday night seem not to expose the above mentioned evening peak, reflecting the fact that many people may go out.

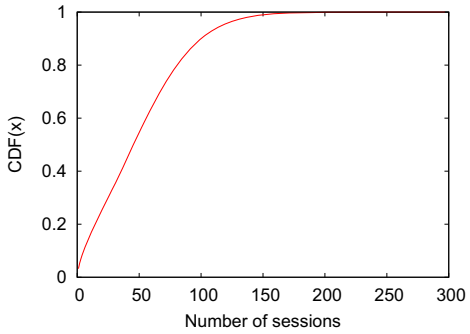
It is important to notice that these patterns describe just a global tendency, and cannot be exploited to make any prediction nor assumption of single user behaviour.

In order to estimate how often and for how long users connect to OSN we measure the frequency and duration of sessions for each user.

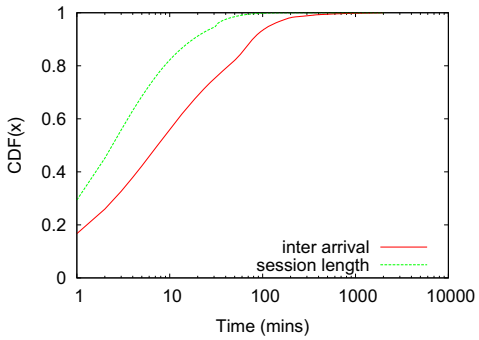
Figure 8b shows how many sessions are done by users (95 % C.I. $\pm$ 0.23). We can notice that the majority of users (90%) exposes on average less than 100 daily sessions while the average number of sessions for all users is less than 4 sessions per day. Figure 8c shows for all users the CDF of the sessions length (95 % C.I. $\pm$ 0.65) and the elapsed time (*inter arrival time*) between two consecutive user's sessions (95 % C.I. $\pm$ 4.44). There is a large variation in the OSN usage among users. However, almost half of users sessions are shorter than 20 minutes (median value of 24 mins), and a significant percentage of 34 % last less than 10 minutes. Only few users sessions (less than 13 %) have a long duration, exceeding the 2 hours. We can notice that almost 50 % of users present an inter-arrival time shorter than 1



(a) Availability of all users.



(b) Number of sessions for all users.



(c) Sessions length and inter arrival time for all users.

Fig. 8 Analysis of the temporal properties

hour. These plots confirm therefore the fact that in OSN the typical session has a short duration. Small inter-arrivals times correspond to users who constantly use the OSN service while large inter-arrival times correspond to users who connect occasionally to the OSN. It is important to notice that the size of the active network of each user is slightly correlated to the temporal properties of the ego, such as the average session length (0.36) and the number of daily sessions (0.20): intuitively we would expect that ego with more active contacts are likely to spend more time on the OSN.

8 Simulation results

To obtain realistic results we use the event-based simulator PeerfactSim.KOM [35, 36] for our simulations which contains already implementations of several DHTs like Pastry [30] and Chord [31]. Each simulation uses GNP coordinates [37] to estimate delays in the fundamental network. Furthermore, measurements from the PingEr project [38] are integrated into the simulator for reasonable approximations of jitter. Each of our simulation has been run with 10 different random seeds, therefore the values we obtained represent the average of 10 different values. In the following, we describe the setup of our simulation in Section 8.1 and evaluate the results in Section 8.2.

8.1 Simulation setup

In Table 2 we briefly describe the setup of our simulations. The experiment size is set to 1859 nodes and contains a small subset of the Facebook'14 data set (*SocialCircles!*)[1] described in Section 7.1. From this set we extracted a subset of nodes which form a connected graph and represent real friendship relationships. From minutes 1 to 10 in the simulation each node starts the application and therefore joins the DHT Overlay. Although our application and data availability service is not limited to a specific DHT overlay, we decided to use Pastry as underlying DHT for our simulations since routing in Pastry is deterministic, fast, and reliable. During simulation each of the simulated nodes observes its interactions with other nodes and calculates thereupon an ordered list in which known contacts (*friends*) are sorted according to their Social Score. Friends with a high degree of interaction are placed at the top of the list and more likely considered as PoS for the data availability protocol than nodes with low contact frequency. In minute 14 those nodes which represent an ego node in the network start to

[1] Available at: http://socialcircles.eu/

Table 2 Simulator setup

Simulator Settings	
Simulator details	PeerfactSim.KOM [35].
Network model	GNP [37], jitter based on [38], no packet loss.
Churn model	User churn according to the Facebook'14 data set from *SocialCircles!*.
Analyzer	Focus on: number of online nodes, profile lookups, and message consumption.
Scenario Settings	
Experiment size	1859 nodes, out of which 28 nodes are egonodes.
DHT Overlay	Pastry.
Parameter α	Different values for frequency of ping-pong messages.
Experiments	
A	One half of the nodes selects new PoS when leaving, whereas the other half fails suddenly.
B	All nodes fail upon churn event, nodes leave without notification.
C	Churn behaviour same to *A*, simulation time is set to 48 hours
Scenario Actions	
Minute 1 –10	Nodes join the network, especially the DHT.
Minute 12	Start periodic calculation of interactions among friends.
Minute 14 –16	Egonodes start the data availability service.
Minute 30	User churn is turned on. Most nodes go offline.
Minute 120	End of the simulation.

publish private data (*profile*) and start the data availability service protocol. By this means they select a good friend from the sorted interaction list, which is available, to be second PoS for their data. In our simulations 28 nodes in total publish their profiles in this phase of the simulation. In order to investigate the rate of available data in the network we start lookups to the published profiles every minute. From minute 30 to the simulation's end user churn is activated. Nodes decide now to join or leave the network according measurements from *SocialCircles!*. Approximately half of the nodes which leave the network decide to leave the network without informing the other PoS about their departure, whereas the other half informs the other PoS about it's intention to leave the network. Therefore it is likely that both PoSs for a given profile leave the network simultaneously without selecting another PoS for the profile. Since more nodes leave the network than join it again in our simulations it is furthermore quite likely that certain profiles are not available throughout the rest of the simulation as soon as both responsible PoSs leave the network simultaneously. Therefore we expect lookup failures at the end of the simulation time when many nodes have left the network.

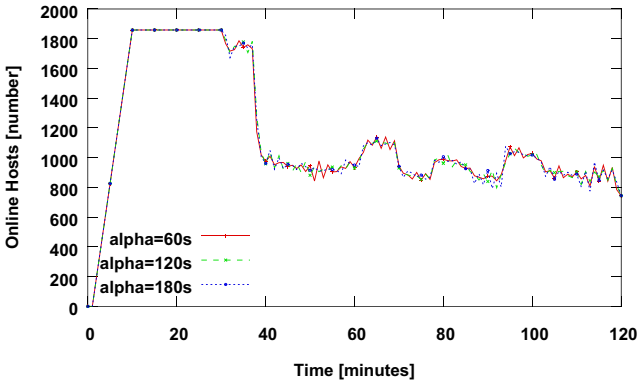
8.2 Evaluation

The goal of our simulations is to investigate the quality of our data availability service in ego networks for a given real-world user behaviour. In order to judge the quality of our proposed protocol we focus on the number of available profiles in the network under churn. Additionally we observe the quantity of nodes which are responsible for storing the published data objects (*PoSs*) and we investigate the costs of our protocol to prevent data loss due to failing nodes.

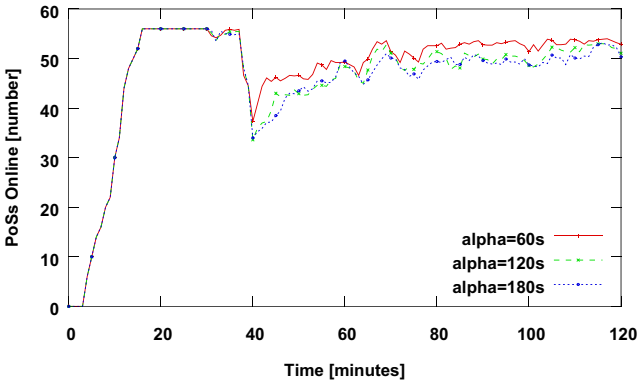
As can be seen in Fig. 9a the number of participating nodes in the network decreases abruptly as soon as the simulation of user churn starts. Approximately 50 percent of the participating nodes leave from minute 30 until the end of the simulation, from 1859 online nodes at the beginning, only 800 to 900 are online at the end of the simulation. In general we observe that more nodes leave the network during our simulations than join the network.

During the simulation 28 nodes publish their profiles and select a second PoS to increase the availability of the profiles. Whenever any PoS leaves the network, voluntarily or abruptly without further proclamation of their departure, another node in the network will be selected to take on the tasks of the leaving node. In Fig. 9b we present the number of online PoSs during the simulation. In Fig. 9a it can be seen that most nodes leave the network abruptly in minutes 38 and 39 of the simulation. Although in this time many PoSs fail due to churn the amount of PoSs slightly recovers after the sudden reduction.

We obtain that some of the published profiles are unreachable throughout the whole simulation time. This circumstance is caused by the sudden failure of both PoSs of a certain profile. If none of the current PoSs of the profile nor



(a) Number of nodes online.



(b) Number of points of storages online.

Fig. 9 Experiment *A*: Quantity of online nodes and points of storages

any other node which has been PoS for the profile before rejoins the network again the published profile will remain unavailable, none of the lookups for this data item will be successful then. Figure 10 shows the ratio of available profiles to the number of total profiles disseminated in the network. In the fortieths minute when the churn rate reaches its maximum the number of reachable profiles decreases rapidly and thereafter increases again so that after 25 minutes approximately 95 % of all profiles are available in the network. The remaining profiles are unavailable unless at least one of the current PoSs for this data item join the network again.

We tested our data availability service with different values of α, where $1/\alpha$ is the frequency with which ping messages are sent by the first PoS to the second PoS and the frequency with which pong messages are sent back respectively. The lower α is chosen the higher the frequency of sent ping messages and the higher the data availability is. Figure 10 reveals that higher values of α do not necessarily lead to an availability rate worse than that with lower values.

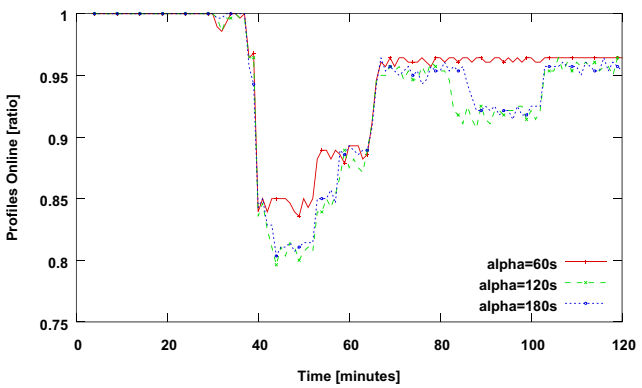
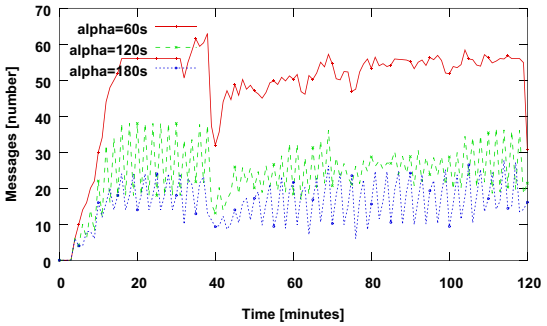


Fig. 10 Experiment *A*: Ratio of available profiles to total number of profiles

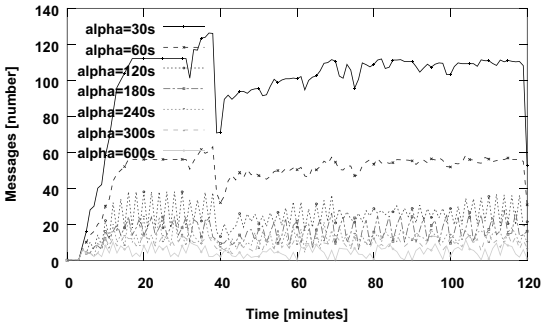
Next, we focus on the costs of the data availability service in terms of message consumption. In order to react on failing nodes, PoSs which are responsible for a certain profile exchange keep-alive messages. For every created ping message a pong message is sent back to inform the supporting PoS about the presence of a PoS. By this means, for every published profile two messages are frequently created to keep the availability of the data object alive. The total number of keep-alive messages sent in a given time-interval T is therefore proportional to $2n\frac{T}{\alpha}$, where n is the total number of profiles published in a given network and $1/\alpha$ is the frequency with which ping messages are created. Figure 11a, b present the simulated amount of keep-alive messages per minute which are used to prevent PoS failures.

In our second experiment, nodes which leave the network due to user churn do not inform other nodes about their departure, instead they leave without selecting a new PoS which could take over the responsibility for the published profile the PoS maintains. Figure 12a is equal to Fig. 9a, approximately half of the nodes fail due to churn. The number of available PoSs during the simulation of Experiment *B* is shown in Fig. 12b. Although failing nodes do not select another PoS the data availability rate is similar to that given in Experiment *A*. In Fig. 13c it can be seen that data availability is the higher, the faster keep-alive messages are exchanged between two PoSs, and the faster node failures are detected. The message costs remain proportional to $2n\frac{T}{\alpha}$, where n is the total number of profiles published, T is the observed time interval and $1/\alpha$ is the frequency with which ping messages are sent, as can be seen in Fig. 12d.

Figure 13 shows the results of Experiment *C* in which we investigate our data availability service with given user behaviour from the Facebook'14 data set (*SocialCircles!*) and with 48 hours of simulated time. During the whole simulation the quantity of online nodes fluctuates around 800 nodes, which is less than 50 percent of all simulated



(a) Message consumption for different *alpha*.



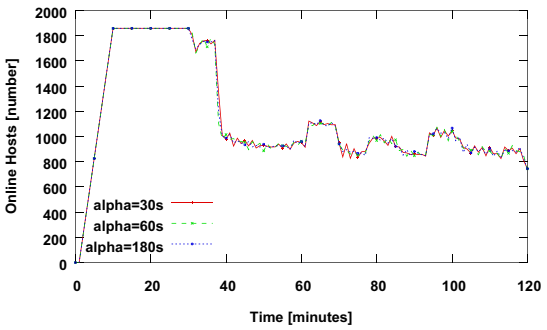
(b) Message consumption is proportional to ping-pong frequency.

Fig. 11 Experiment *A*: Quantity of ping-pong messages for different values of $alpha$

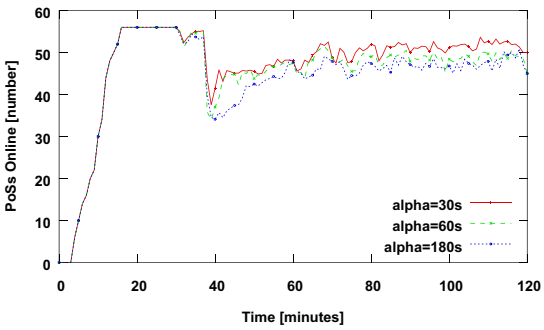
nodes (Fig. 13a). The results given by Fig. 10 and 13c indicate that our proposed data availability service protocol with only 2 PoSs provides high availability of data objects under realistic circumstances and realistic user behaviour. Furthermore, the number of keep-alive messages in this simulation is proportional to the number of published profiles and proportional to the inverse of α. As shown in Fig. 13d for $\alpha = 30s$ the amount of messages is higher than for higher values of α (lower frequency). For $\alpha = 180s$, the highest value for α, the fewest messages are sent.

8.3 PoS election evaluation

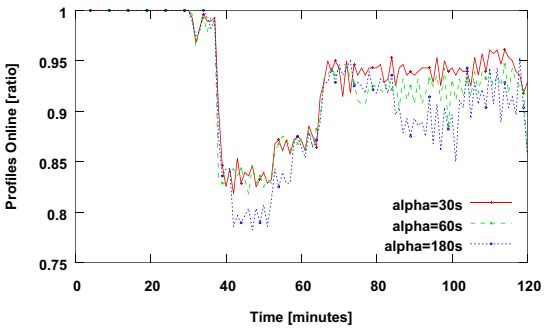
To evaluate the PoS election strategy and how is good in term of data availability we use the complete Facebook



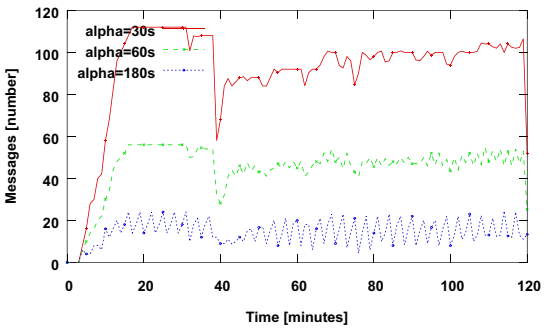
(a) Number of nodes online.



(b) Number of points of storages available in the network.

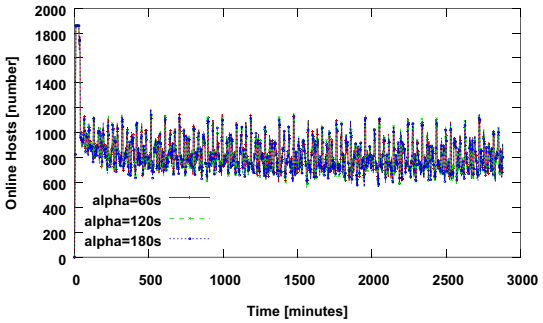


(c) ratio of availability profiles to total amount of published profiles.

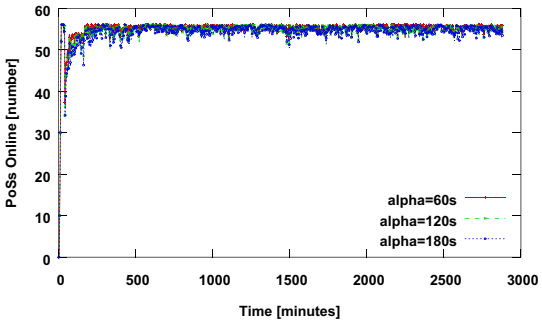


(d) Costs in terms of messages.

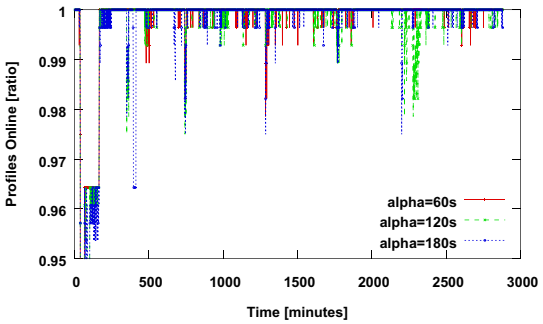
Fig. 12 Experiment *B*: Simulation results, all nodes leave the network without selecting another point of storage (failure)



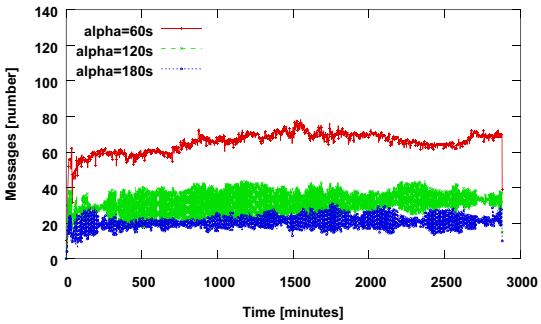
(a) Number of nodes online.



(b) Number of points of storages available in the network.



(c) ratio of availability profiles to total amount of published profiles.



(d) Costs in terms of messages.

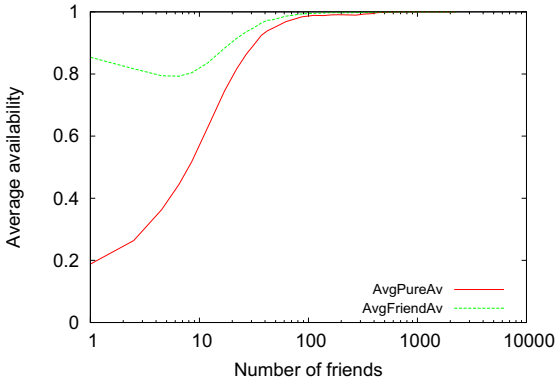
Fig. 13 Experiment *C*: Simulation results, half of the leaving nodes select another PoS wheres the other half fails. The simulation time is set to 48 hours

dataset (308 ego networks). The simulation replicates the actual online status of the users by considering the timestamp information. We exploit the metrics proposed in [22] to evaluate our system:

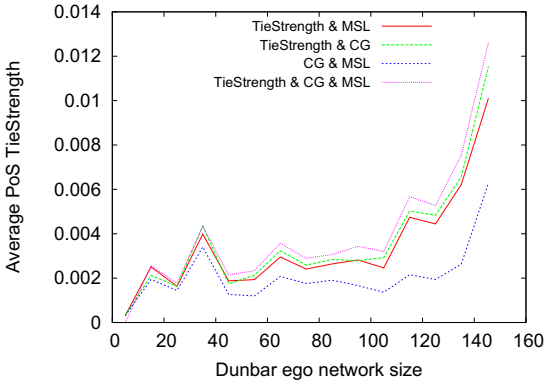
1. *Pure Availability:* [3] fraction of time in a day a user's profile is reachable through his PoS, that is the union of times the user has at least one online PoS.
2. *Friend Availability:* [3] fraction of time in a day a user's profile is reachable through his PoS from his friends. In other words, since the profile of a user is accessible only from his friends, this metric focus on the fraction of time the profile of a user is available when his friend are online. In a social network higher *Friend Availability* (even with a lower *Pure Availability*) is desirable.
3. *Average PoS Tie Strength:* average tie strength value between the user and his online PoS. A high value means that the PoS have been chosen among more trusted friends.
4. *Average Number of Elections:* average number of PoS elections made by a user.

In [22], we have already assessed the individual PoS selection strategies and the social score strategy which takes into account all of them. In this work, we consider all possible pairs of individual selection strategies; namely *TieStrength* and *ConnectionGain* (TieStrength & CG), *TieStrength* and *Medium Session Length* (TieStrength & MSL) and *ConnectionGain* with *Medium Session Length* strategy. Furthermore, we compare each heuristic with the social score (i.e. TieStrength & CG & MSL) and consider a larger dataset obtained from a Facebook application (see Section 7.1).

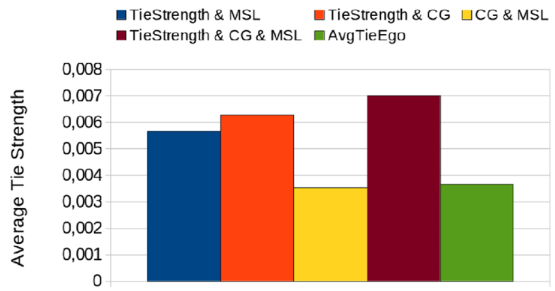
Pure availability* and *friend availability *Pure Availability* and *Friend Availability* in the *Facebook Dataset* are shown in Fig. 14a. Since availability depends on users' traces and only marginally on PoS selection strategies, results are very similar for all the proposed strategies. Hence, we show only the graphs corresponding to the TieStrength & CG & MSL selection strategy. *Pure Availability* increases monotonically and an availability of 90 % is achieved for nodes with more than 40 friends. As we expected, *Friend Availability* is always greater than or equal to *Pure Availability*. It is remarkable to note that also a user with a relatively small number of friends (e.g. 20) achieves more than 90 % of *Friend Availability* on average. For nodes with a very small number of friends (i.e. < 10) the difference between *Pure Availability* and *Friend Availability* is remarkable, since neither the node nor its friends are online for long periods of time.



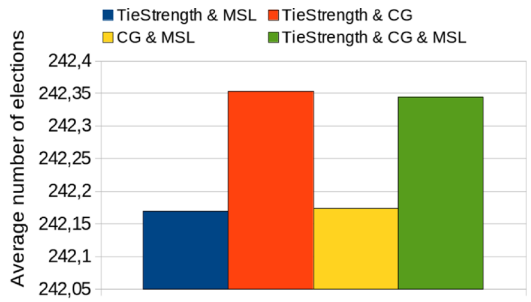
(a) Pure Availability and Friend Availability



(b) Average PoS tie strength and Dunbar ego network



(c) Average tie strength



(d) Average Number of Elections

Fig. 14 Simulation results

Average tie strength Since a user's PoS are always chosen between his online friends, we expect load increases proportionally to the users' ego network size and to the percentage of time they spend online in the system.

The maximum load is about 70 profiles and it is almost the same for all the considered strategies. Figure 14b shows a plot of the average tie strength values between the nodes and their online PoS as function of the Dunbar ego network size by considering the different paired strategies. The average value of the tie strength returned by the different strategies is almost the same for small sized Dunbar ego networks, since the choice of PoS is always restricted among nodes that are online at the moment of the election. As the Dunbar ego network size grows, the difference between the average tie strength values obtained with the different strategies increases and, as expected, the highest tie strength value is obtained with the TieStrength & CG & MSL strategy. Since the CG & MSL strategy selects PoS regardless the tie strength value with the ego, it presents the smallest tie strength among all heuristics.

The histogram in Fig. 14c shows the average value of tie strength between egos and their online PoSs. The average values of the different strategies are compared with the average tie strength between the same egos and all their ego friends (*AvgTieEgo*). As before, the average PoS tie strength is much higher for TieStrength & CG & MSL than for the other strategies while CG & MSL strategy is quite similar to the average tie strength of the Dunbar ego network.

Average number of elections The average number of elections made by each user is shown in the histogram of Fig. 14d. The minimum average number of elections is obtained with the TieStrength & MSL and CG & MSL strategies that exploit information about users' sessions for PoS elections while for the remaining strategies these values are quite similar. Currently, medium session length is computed during the simulation, by considering previous sessions of the users during the considered period. We believe that the number of elections may be further reduced, by using longer periods of times and/or by refining the methodology used to predict users' availability.

9 Conclusion

This paper has presented an architecture for Distributed Online Social Network with the aim to guarantee data availability even when users are offline. We have presented a

distributed storage support in which users' data are stored on trusted friends. It guarantees the users' data persistence and the data consistence. Each node dynamically elects a minimal set of Point of Storages among his friends by choosing at first between Dunbar friends. User data are dynamically transferred between online users in order to maximise the availability of users' profiles in the social network. Our experimental results show that our system is able to guarantee an high availability with only two online copies of each profiles. Furthermore, our Point of Storages management guarantees an high availability of the online profile data by considering real users' online traces obtained from a real Facebook dataset. We plan to investigate how to manage the load balancing in our system and we want to apply a more sophisticated policy which considers the Dunbar-based ego network and users' information to allow a better control of where user data are stored. Furthermore, we plan to consider privacy techniques to manage the access to the PoS Table which is stored into the DHT and to study more in detail the problem of data consistence.

References

1. Datta A, Buchegger S, Vu L-H, Strufe T, Rzadca K (2010) Decentralized online social networks. In: Handbook of social network technologies and applications. Springer
2. Adya A, Bolosky WJ, Castro M, Cermak G, Chaiken R, Douceur JR, Howell J, Lorch JR, Theimer M, Wattenhofer RP (2002) Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In: Proceedings of the 5th symposium on operating systems design and implementation (OSDI)
3. Narendula R, Papaioannou TG, Aberer K (2012) A decentralized online social network with efficient user-driven replication. In: International Conference on Social Computing (SocialCom)
4. Schiöberg D, Schneider F, Trédan G, Uhlig S, Feldmann A (2012) Revisiting content availability in distributed online social networks. CoRR. arXiv:1210.1394
5. Tinedo RG, Artigas MS, López PG (2012) Analysis of data availability in F2F storage systems: When correlations matter. In: Proceedings of the 12th international conference on peer-to-peer computing (P2P)
6. Dunbar RIM (1998) The social brain hypothesis. Evolutionary Anthropology: Issues, News, and Reviews vol. 6
7. Roberts SGB, Dunbar RIM, Pollet TV, Kuppens T (2009) Exploring variation in active network size: Constraints and ego characteristics. Soc Networks 31
8. Sutcliffe A, Dunbar R, Binder J, Arrow H (2012) Relationships and the social brain: Integrating psychological and evolutionary perspectives. Br J Psychol 103
9. Graffi K, Gross C, Stingl D, Hartung D, Kovacevic A, Steinmetz R (2011) LifeSocial.KOM: A secure and p2p-based solution for online social networks. In: IEEE CCNC
10. Tran DN, Chiang F, Li J (2008) Friendstore: Cooperative online backup using trusted nodes. In: Proceedings of the 1st workshop on social network systems (SNS)
11. Cutillo LA, Molva R, Strufe T (2009) Safebook: A privacy preserving online social network leveraging on real-life trust. IEEE Commun Mag 47
12. Diaspora Website https://diasporafoundation.org/
13. Sharma R, Datta A (2011) SuperNova: Super-peers based architecture for decentralized online social networks. CoRR. arXiv:1105.0074
14. Li J, Dabek F (2006) F2F: Reliable storage in open networks. In: Proceedings of the 5th international workshop on peer-to-peer systems (IPTPS)
15. Pamies-Juarez L, López PG, Artigas MS (2009) Heterogeneity-aware erasure codes for peer-to-peer storage systems. In: International conference on parallel processing (ICPP)
16. Blond S, Fessant F, Merrer E (2009) Finding good partners in availability-aware P2P networks. In: Proceedings of the 11th international symposium on stabilization, safety, and security of distributed systems (SSS)
17. Mickens JW, Noble BD (2006) Exploiting availability prediction in distributed systems. In: Proceedings of the 3rd symposium on networked systems design and implementation (NSDI)
18. Boutet A, Kermarrec A-M, Le Merrer E, Van Kempen A (2012) On the impact of users availability in OSNs. In: Proceedings of the 5th workshop on social network systems (SNS)
19. Sharma R, Datta A, Dell'Amico M, Michiardi P (2011) An empirical study of availability in friend-to-friend storage systems. In: International conference on peer-to-peer computing (P2P)
20. Li J, Dabek F (2006) F2F: Reliable storage in open networks. In: Proceedings of the 5th international workshop on peer-to-peer systems (IPTPS)
21. Conti M, De Salve A, Guidi B, Ricci L (2014) Epidemic diffusion of social updates in dunbar based dosn. In: 2nd workshop on large scale distributed virtual environments on clouds and P2P (LSDVE 2014)
22. Conti M, De Salve A, Guidi B, Pitto F, Ricci L (2014) Trusted dynamic storage for dunbar-based P2P online social networks. In: On the move to meaningful internet systems: OTM 2014 conferences
23. Balakrishnan H, Kaashoek MF, Karger D, Morris R, Stoica I (2003) Looking up data in P2P systems. Commun ACM 46
24. Benevenuto F, Rodrigues T, Cha M, Almeida VAF (2009) Characterizing user behavior in online social networks. In: Proceedings of the 9th conference on internet measurement
25. Mega G, Montresor A, Picco GP (2011) Efficient dissemination in decentralized social networks. In: International conference on peer-to-peer computing (P2P)
26. Arnaboldi V, Guazzini A, Passarella A (2013) Egocentric online social networks: Analysis of key features and prediction of tie strength in Facebook. Comput Commun 36
27. Everett MG, Borgatti SP (2005) Ego network betweenness. Soc Networks 27
28. Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Advances in Cryptology - EUROCRYPT 2005. Springer
29. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: Symposium on security and privacy (SP)
30. Rowstron AIT, Druschel P (2001) Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Proceedings of the international conference on distributed systems platforms
31. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) 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
32. Bhagwan R, Savage S, Voelker G (2003) Understanding availability. In: Proceedings of IPTPS'03

33. La Gala M, Arnaboldi V, Conti M, Passarella A (2012) Ego-Net digger: A new way to study ego networks in online social networks. In: Proceedings of the 1st ACM international workshop on hot topics on interdisciplinary social networks research
34. Golder S, Wilkinson D, Huberman B (2007) Rhythms of social interaction: Messaging within a massive online network. In: Communities and technologies 2007, pp 41–66
35. Graffi K (2011) PeerfactSim.KOM: A P2P system simulator experiences and lessons learned. In: Proceedings of the international conference on peer-to-peer computing (P2P)
36. Feldotto M, Graffi K (2013) Comparative evaluation of peer-to-peer systems using PeerfactSim.KOM. In: International conference on high performance computing and simulation (HPCS)
37. Ng TSE, Zhang H (2002) Predicting internet network distance with coordinates-based approaches. In: Proceedings of the international conference on the joint conference of the ieee computer and communications societies
38. Matthews W, Cottrell L (2000) The PingER Project: Active internet performance monitoring for the HENP community. IEEE Commun Mag 38

Barbara Guidi is a PhD student at the Department of Computer Science, University of Pisa. She received his Bachelor degree in February 2007 and the M.Sc degree in October 2011, from University of Pisa. She started her Phd in January 2012 and she is currently an associate researcher at the Istituto di Informatica e Telematica (IIT), CNR, Pisa. Since the beginning of her PhD, Barbara Guidi has been working on a new promising approach to the definition of Distributed Online Social Networks based on Dunbar's theory .

Tobias Amft he started to study computer science, at the Heinrich Heine University of Dusseldorf. In summer 2012 he received the B.Sc. degree after submitting his bachelor thesis. Tobias Amft received his M.Sc degree in March 2014 from the Heinrich Heine University after submitting his master thesis in the field of peer-to-peer systems and computer networks. Since April 2014, Tobias Amft is a doctoral student and researcher in the Lab for Technology of Social Networks, led by Jun.-Prof. Dr.-Ing. Kalman Graffi. Beside his research he is pursuing the M.Sc. degree in physics.

Andrea De Salve is a PhD student at the Department of Computer Science of the University of Pisa. In 2009, he received the Bachelor degree and in 2012 he obtained the Master degree in Computer Science at the University of Pisa. His research interests include: P2P systems and its application to distributed of the OSNs, and Protocols and algorithms for large scale distributed system. Since 2012, he is working in the field of privacy-preserving systems for Decentralized Online Social Networks.

Professor Dr.-Ing. Kalman Graffi is heading the lab for Technology of Social Networks, in the Institute of Computer Science, at the Heinrich-Heine-Universität Dsseldorf, Germany. Dr. Graffi received his diploma degrees in Computer Science and Mathematics from the Technische Universität Darmstadt in 2006. From 2006 to 2010, he was a PhD student at the Multimedia Communications Lab (KOM), at the Technische Universitt Darmstadt. In 2010, he finished his PhD (Dr.-Ing.) on the "Monitoring and Management of Peer-to-Peer Systems" at the Technische Universitt Darmstadt "mit Auszeichnung" (summa cum laude). From 2011 to 2012, he was postdoctoral researchers at the Theory of Distributed Systems Lab at University of Paderborn. Findings from his research have been published in more than 30 refereed scholary publications. He serves as reviewer for prestigious conferences and journals.

Laura Ricci is a Professor of the Department of Computer Science, University of Pisa where she has taught several courses in the area of Computer Networks. She is a Research Associate of ISTI CNR, Pisa, where she has collaborated to international projects in the area of cloud, high performance and P2P computing. She is the co-chair of the LSDVE workshops series, Large Scale Distributed Virtual Environments on Cloud and P2P, held in conjunction with EUROPAR. She has been the guest editor of several special issues in international journals and has chaired workshops in International Conferences. Laura Ricci is author of more than 90 papers published in in refereed journals, books and conference proceedings. Her main research interests are in the area of distributed computing, in particular cloud, P2P and data intensive computing.