Future Generation Computer Systems 40 (2014) 1–16

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Giving form to social cloud storage through experimentation: Issues and insights

Raúl Gracia-Tinedo *, Marc Sánchez-Artigas, Aleix Ramírez, Adrián Moreno-Martínez, Xavier León, Pedro García-López

Universitat Rovira i Virgili, Department of Computer Engineering and Maths, Av. Paisos Catalans 26, Tarragona, Spain

HIGHLIGHTS

- To define and explore a social cloud for storage.
- Impact of network topology/user availability on a social cloud storage service.
- We implement a hybrid model (contributory storage/cloud service) in FriendBox.
- Trade-offs between service quality and economic cost in a social cloud.
- Alternative mechanisms to make a fairer use of user resources.

ARTICLE INFO

Article history:
Received 8 January 2014
Received in revised form
23 April 2014
Accepted 11 May 2014
Available online 17 May 2014

Keywords:
Social cloud
Social networks
Cloud computing
Storage

ABSTRACT

In the last few years, we have seen a rapid expansion of social networking. Digital relationships between individuals are becoming capital for turning to one another for communication and collaboration. These online relationships are creating new opportunities to define socially oriented computing models. In this paper, we propose to leverage these relationships to form a dynamic "social cloud" for storage. While at first glance, the concept of social cloud looks very appealing, a deeper analysis brings out many problems, particularly in data availability. To overcome this issue, in addition to digital friends, we propose to the members of the social cloud the use of online storage services like Amazon S3 to store data and improve data availability. Through a real deployment in our campus, we study what aspects give form to the definition of social cloud storage and determine the difficulty of realizing this concept in the real world. Our analysis reveals interesting insights of how to reap the full potential of socially oriented storage.

1. Introduction

Online social networks, such as Facebook, Google+ and LinkedIn, are becoming a predominant service today. Catering for people of all ages, gender and class, social networking services have become the primary means of communication between friends, family and colleagues. These digital relationships are creating new opportunities to spur the adoption of socially oriented computing.

One representative example of this trend is the concept of "social cloud" as a means of facilitating resource sharing by utilizing the relationships established between members of a social network [1,2]. A social cloud leverages preexisting trust relationships between users to enable mutually beneficial sharing. This facilitates long term sharing with lower privacy and security requirements than those that are present in traditional cloud environments. For the time being, the cloud accrues massive amounts of private information to provide for instance highly targeted advertisements. Not surprisingly, security breaches, poor judgment, or even the lack of judicial oversight leaves users vulnerable. In this sense, the "social cloud" represents a new form for the users to retake control of the cloud service, avoiding to be tracked or give personal information against their will, or in a way in which they feel uncomfortable. In fact, as pointed out by S. Pearson [3], one of the "top six" recommended privacy practices for cloud systems is to maximize user control, which is one of the outstanding feature of the "social cloud".

Another distinguishing feature of the "social cloud" is that the network comes first. It is not a cloud or middleware extended with a social network; rather, it is a social network extended with

* Corresponding author. Tel.: +34 977558745.
E-mail addresses: raul.gracia@urv.cat (R. Gracia-Tinedo), marc.sanchez@urv.cat (M. Sánchez-Artigas), aleix.ramirez@urv.cat (A. Ramírez), adrian.moreno@urv.cat (A. Moreno-Martínez), xavier.leon@urv.cat (X. León), pedro.garcia@urv.cat (P. García-López).

cloud functionality. Users form the basic infrastructure and share resources around their social graphs. Such an organization brings out many benefits. For instance, one of those advantages is usability, since the interface and tools for resource sharing are already familiar to users. Another one is that it allows users to maximize the control of the cloud service by letting users choose how their resources will be used. Giving users the control over their personal information and resources engenders trust, but this can be difficult in a cloud computing scenario. This feature is very interesting for the adoption of the "social cloud", as it permits users to define a series of preferences for the management of their personal data, and take account for that, among other advantages.

1.1. Motivation

The social cloud also carries important deficiencies. The most critical one is that, contrary to commercial clouds, it is not feasible to establish a formal Service Level Agreement (SLA) within a social cloud system. Its operational feasibility is based on the premise that participants are socially motivated and subject to the personal repercussions outside the functional scope of the social cloud. This is primarily due to the existing level of trust that already exists between members. In this context, SLAs or "contracts" should be viewed as a best effort agreement between the social links. This weaker form of agreement translates into a limited availability of resources and capabilities. Although a social cloud system is built upon social incentives, peer pressure, etc., the discontinuous participation of social contacts, or even the abandon of the social cloud, is intrinsic to the nature of social relationships.

In terms of storage, this means that the data stored within the social cloud may be subject to recurrent periods of unavailability. In a social cloud, the percentage of time that data is available is a function of the number of friends contributing their storage space over time. And such a dependence has deep implications for the correct operation of a social cloud, mainly in terms of data availability, understood as the probability to access a data item when needed.

First, while there may be a sizable number of individuals in a social network, typically only an insignificant number can be utilized as a destination for personal data. To inform this argument, over 63% of `Facebook` users have less than 100 friends, and the majority of social interactions occur only across a small subset of them [4]. More specifically, it has recently been observed that only 20% of the social links capitalize 70% of all social interactions [4]. This means that in practice the number of users willing to contribute their storage resources to sustain the social cloud will be small. If in addition to this we add the problem of the temporal correlation in the connection habits of users, the loss of data availability is inevitable. Real measurements from online social networks have detected the presence of strong daily and weekly interaction patterns [5,6]. Very succinctly, this means that the probability of finding simultaneously offline all the social links of a user is high, particularly during night hours, which makes it impossible to maintain data availability even under full replication where a replica is allocated to every member of the social cloud.

Second, the topology of the social network graph plays a central role. As such, it delineates the interaction events that may occur across social links and hence, the amount of resources to be contributed by a member. Although users with many friends have a greater opportunity to store their data with higher availability, they may possibly have to donate more disk space to reciprocate a larger number of friends. Real measurements of social networks [7,4] show that while clustering is very high, the existence of a few users with a large number of friends is characteristic of social interaction. For these users with abnormally high degrees, usually called *hubs* in the graph literature, the contribution of their storage resources may be high for little or no personal gain. In this sense, poor storage fairness may motivate the need for economic or non-economic mechanisms to regulate sharing within a social cloud. Determining the graph properties that have an important bearing on a social cloud is critical to answer questions like: *Is the clustering coefficient a valid indicator of resource contribution? If not, which graph properties determine the obligation to trade storage resources?*

Overall, understanding these factors is a necessary step in determining whether the vision of social cloud is realizable, and therefore, it can really emerge as an alternative to commercial cloud providers. Compared with cloud storage, the information is made only available to *trustable* members of the social network, thus significantly reducing the risk that personal data might be sold on, and without raising suspicions about how commercial storage services are monetized.

1.2. Contributions

However, to truly involve users, we believe that the promise of always available storage is essential. In a recent paper, we demonstrated that this promise cannot be fulfilled today using only social links as discussed above [8]. For this reason, we study in this paper a realistic model for building highly available social clouds. In this model, the storage resources contributed by each user are augmented with an external cloud storage service like Amazon S3. Each member of the social cloud brings out its online cloud storage service to store parts of its data and mask the recurrent, unavailability periods of "friends". Our objective is to improve the resilience of the social cloud to correlated failures and departures.

Our key insight is the following: *Since the central cloud maintains data availability during the time periods where most of the social links are disconnected, we are taking the first step towards the realization of storage as a service atop a social cloud*, i.e., the illusion that users can store their data to a socially motivated cloud and access them anytime from anywhere.

At this point, a natural question that arises is: *What can this storage model offer that more established sites, like* `Facebook` *or a combination of cloud storage plus social network like* `Google Drive` *with* `Google+`*, don't?* The answer is that data is not in possession of these sites and therefore, they cannot generate revenue, for instance, by targeting ads to specific demographics (e.g., single males up to 21 years of age). Specifically, in a social cloud, the control of data, and who can access it, is entirely in the hands of users. The role of the social network site is restricted to connecting and recruiting members for the social cloud through a familiar interface. But the data is out of the control of the social network site operators.

While that sounds good, the use of the cloud also poses a new question: *Does the use of a public cloud such as Amazon S3 carry the danger of undermining the security achieved by the social cloud?* Fortunately, the answer is negative, because our model operates by first encoding, and then distributing, the information between the social contacts and the cloud in such a manner that the cloud cannot recover the original data. In our particular case, we use a non-systematic Reed–Solomon code [9] for that purpose. The code was chosen to be non-systematic in order to make the encoded data not readable at once. Recall that threshold schemes like Shamir's scheme [10] for sharing a secret among multiple participants can be re-formulated in terms of Reed–Solomon codes [11]. As a result, we can blend "the best of both worlds" in a single approach: high data availability and security, the latter thanks to both the maximization of user control and the minimization of the data sent to and stored in the cloud.

To gain a better understanding, this paper contributes to the state of the art by quantifying the influence of the above factors, putting special emphasis on the topological effects, while outlining some of the challenges to make the concept of social cloud storage a reality. To conduct this study, we have instantiated this model into FriendBox [12], a social cloud storage application embedded into Facebook. Through a real deployment in our campus, we spot evidence of the bearing of these factors on the definition of social cloud storage. The fact that our results has been obtained through experimentation gives the additional advantage of measuring the real impact that these factors and design choices may have on performance and cost. Our results provide interesting insights of how to reap the full potential of socially oriented storage and confirm the feasibility of this model.

The remainder of this paper is structured as follows: In Section 2, we overview related works. Section 3 motivates and characterizes the social cloud storage paradigm. In Section 4 we describe FriendBox, our social storage application to conduct our experimental assessment, which is included in Section 5. Finally, we provide some discussion about our empirical insights in Section 6 and we conclude in Section 7.

2. Related work

Many works in the literature discuss about the use of social networks for building computing systems and incentivizing resource sharing. One can find countless examples of applications that leverage existing social networks to manage and authenticate users and even recruit volunteers. For instance, both ASPEN [13] and PolarGrid [14] use social networks to manage users and facilitate resource sharing.

The social cloud model, first proposed in [1], takes a different tack by extending cloud-like functionality to online social networks instead of incorporating social networking to existing computation platforms. Since its born, a plethora of works have been examining the potential of this new social paradigm, particularly for underpinning computation [15,16].

In the case of storage, only the research works [1,2] partially explain some of the barriers to overcome towards the realization of socially oriented cloud storage. More specifically, these works concentrate on how to support storage trading through various social market metaphors but do not give any discussion on the operational requirements of the social cloud storage like data availability and the amount of storage space to be contributed by friends. Our work aims at filling this gap by spotting concrete evidence of the existing operational hurdles in the social cloud storage model.

In addition, there is a great deal of synergy between the social cloud and P2P networking paradigm in that services are provided by a network of peers. The P2P literature is full of examples of storage systems where the storage capacity is contributed by a pool of distributed peers such as Samsara [17] and PAST [18]. However, these systems lack of accountability, familiar interface, and the social incentives that minimize the administrative overhead, which are precisely the costs that P2P systems are meant to avoid.

Much closer to our work are, however, peer-assisted storage systems where the spare network bandwidth and storage space of peers complement that of a cloud storage service such as Amazon S3. The key feature of peer-assisted storage is that it is comparable to the traditional client–server architecture but at a fraction of its costs [19]. A representative example was Wuala,[1] a commercial storage service that now only stores files in data centers but that in the past it stored (encoded) fragments of the data on peers to save bandwidth at the server side [20]. Another example is AmazingStore [21], which augments centralized cloud-based storage service with a P2P network to improve its resilience to correlated failures.

Unlike peer-assisted systems, a social cloud exists within the context of social network and is governed by existing social ties. Users retain the control of the service and benefit from social incentives and peer pressure to minimize the administrative overhead needed to ensure that peers contribute their storage space. Because the system benefits users individually, but the costs are shared, users have little or no incentive to contribute to the system. This is expected to happen in a significant less degree in a social cloud since participants are not anonymous and very often know each other's real identities. Hence, negative actions such as promising to store data and then immediately discard it can have repercussions far beyond the social cloud. The major threats are associated with the fact that the data is stored in a remote data center operated by a third party. However, data can be easily protected using obfuscation techniques and encryption. In our specific case, we encode data using Reed–Solomon codes to protect data and distribute the encoded pieces between the social ties and the cloud.

Perhaps the closest vein of related work is our prior research on friend-to-friend storage systems [8,12,22], where we inadvertently instantiate the definition of "social cloud" for our study of data availability [8] and data transfer scheduling [22]. Actually, FriendBox as originally published in [12] matches the definition of social cloud given by [2] except for the presence of a social marketplace as a means of regulating sharing: it extends Facebook with cloud storage functionality such that people and their social contacts form the basic infrastructure, obeying the principle that a social cloud must be controlled and managed by its users.

Compared with our previous work, this article goes a step further by asking questions like: *"What is the role of social graph in the obligation to trade storage space? Is there any significant asymmetry in the level of contribution by users such that an altruistic model is infeasible? Is the availability of a user indicative of its real contribution level to the social cloud?"* Questions that have not been raised in the existing literature. We believe that answering these questions is vital to appraise to what extent the social cloud can emerge as true alternative to existing commercial and non-profit storage systems.

3. Social cloud storage

Online social networks are becoming the primary means of communication between friends, family, and colleagues, which is evidenced by their rapid and ongoing growth. Only Facebook has over 1.11 billion active users of which half log on every day.[2] The potential of this large user base and the inherent trust in digital relationships is huge but has been relatively unexplored for socially oriented computing, although the first applications are starting to appear around the concept of "social cloud". Concretely, a "social cloud" is defined in [2] as:

"A social cloud is a resource and service sharing framework utilizing relationships established between members of a social network".

The "social cloud" is built upon the principle that the trust carried by social links can be applied in any scenario where sharing and collaboration takes place, e.g., to reach out to non technical users who otherwise would rarely donate their computational resources for scientific projects [15,16]. The outstanding feature of the social cloud is that the social network comes first. It is not a middleware, P2P infrastructure or cloud system extended with social networking. Rather, it is a social network extended with cloud-like functionality. Users form the basic infrastructure and share resources around their unique social graph. Such a structure presents a number of advantages, such as usability and intrinsic motivation: *usability*, because the interface and tools for resource

1 http://www.wuala.com/.

2 Facebook Reports 1st Quarter 2013 Results, http://investor.fb.com/releasedetail.cfm?ReleaseID=761090, last accessed June 2013.

sharing are already familiar to users; *intrinsic motivation*, because the resources shared have been invested by the users themselves and are subject to the socially corrective mechanisms inherent in social networks. But perhaps the most interesting advantage is that *users retain the control of the service*, which is very difficult if not impossible in a cloud computing scenario.

Storage is perhaps the most intuitive resource to share in a social cloud. Online cloud storage is becoming mainstream today. A rush of online cloud storage services are entering the market, ranging from services with basic functionality like Amazon S3 to services like `Dropbox` with a full suite of features. Social cloud storage can be viewed as an alternative approach where individual users of a social network contribute their surplus storage capacity to the members of the social cloud, and utilize the trust exhibited in social networks as a guarantee for good behavior.

While prior research has focused on economical models (posted price, auction, ...) to control and regulate sharing [1,2], the nature of interaction in social networks poses numerous technical challenges associated with the level of service that members hope to receive. Unlike commercial storage service, it is not feasible to establish a formal SLA within a social cloud: its correct operation is based on the principle that individuals are socially motivated and subject to the personal repercussions outside the functional scope of the social cloud. Rather, SLAs or "contracts" should be re-interpreted as a best effort agreement between the members of the social cloud. This weaker form of agreement has implications at various levels, starting from data availability, which is the general quality that any user hopes to receive from a storage service.

3.1. Data availability

Although the concept of "social cloud" is built upon social incentives, peer pressure, etc., the discontinuous participation of social links is intrinsic to the nature of interaction in online social networks (OSNs). In terms of storage, intermittent participation means that data may be subject to recurrent periods of unavailability, which may be long depending on the activity pattern between pairs of users. Unlike commercial cloud storage systems like Amazon S3 and `Windows Azure` that offer several nines of data availability, the availability of any particular file cannot be guaranteed in a social cloud. At any given time, *data availability depends on the number and availability of the social links with whom content is shared*.

While there may be a sizable number of users in a social network, only an insignificant number may be sufficiently trustable to be recruited to store personal information about oneself. Although 100 friends per user can seem a big number at first sight, the reality is that the majority of social interactions occur only across a small subset of the social links. For instance, it has been recently observed in `Facebook` that only 20% of the social links capitalize 70% of all social interactions [4]. In practice, this means that the total available storage for a user is contributed by tens of friends. Given that friendship in a social graph can refer to a number of possible relationships in real life, going from family or close friends to mere acquaintances, the total available storage can be even smaller if a user only considers "strong tie" relations such as family.

This problem can be aggravated by the existence of availability correlations in connection habits. Indeed, real measurements from online social networks have found the presence of strong *daily* and *weekly* patterns [5,6]. In practice, this means that *the probability of finding logged off all the social links storing some specific content is high*, particularly during night hours.

More formally, let us observe a social cloud for a period of τ *discrete* time units (minutes, hours, or other quanta) s.t. the observation period can be described by the totally ordered set $T = \{t_1, t_2, t_3, \ldots, t_\tau\}$. By using this simple formulation, the uptime of a member v of the social cloud can be described with a time trace T_v where time unit t_i will be in the trace if and only if v was online at time t_i. In general, every user v will be online for a subset of time $T_v \subseteq T$ and its availability will be $\frac{|T_v|}{|T|}$.

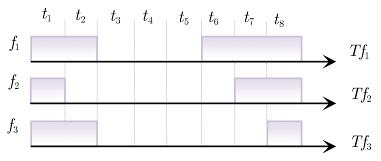


Fig. 1. Example of data unavailability due to availability correlation.

Let us now denote by $F_v = \{f_{v_1}, f_{v_2}, \ldots, f_{v_n}\}$ the set of social links for a member v of the social cloud. Clearly, the periods of data unavailability U_v where v will not be able to access the data stored in its friends even in the case that one replica is allocated to each friend will be:

$$U_v = T - \bigcup_{i=1}^{|F_v|} T_{f_{v_i}}.$$

The fraction of time that a data item shared with friends will be *unavailable* is thus $\frac{|U_v|}{|T|}$. An example of data unavailability due to availability correlation is shown in Fig. 1. As can be seen in the figure, the three friends are *simultaneously* offline during the time range between $t3$ and $t5$, which results in a data availability of $1 - \frac{|U_v|}{|T|} = 1 - \frac{3}{8} = 62.5\%$, very far from the 99.9% data availability offered by cloud storage services like Amazon S3. This example shows that highly available storage cannot be provided within a social cloud, even replicating a data object to all friends, due to the small friend graphs and the correlated availability patterns. This clearly makes it infeasible to define formal SLAs to specify the requirements and obligations of a storage trade, as they will be frequently violated, contrarily to what was promoted in [2].

The poor data availability is one of the reasons that motivates the addition of a cloud storage provider like Amazon and RackSpace into the social cloud. The idea is that the cloud storage service helps covering data availability during the periods of data unavailability.

3.2. Contributory storage

Another important issue not discussed in the incipient social cloud literature is the contribution a user should make in return for the storage service provided by its social ties. Despite the fact that hard disks are usually half empty [23], users are often reluctant to relinquish their free storage space, because they consider storage space as a limited resource. Even when users are given the opportunity to restrict the amount of contribution, this option requires users to decide a priori what is a reasonable contribution.

Given the need to introduce redundancy to improve data availability, *social ties will generally need to donate much more storage space than the amount they consume*. Such asymmetry might make it hard to sustain a social cloud based purely on socially corrective mechanisms (*incentives*, *peer pressure*, ...) and altruism. Clearly, this emphasizes the need for understanding to what extent the asymmetry in contributory levels requires control and regulation. If the level of asymmetry is high, then an economic market like a posted price or an auction market can be set up to regulate sharing as in [2].

It is important to note here that a social cloud is not crowdsourcing as the relationships in the social cloud are generally symmetric. In other words, members are more or less seen as equals who provision resources to benefit from sharing, whereas crowdsourcing operates in the master–worker model where work flows in one direction, which does not in itself constitute sharing.

In the absence of any marketplace, users contribute resources for little or no personal gain, which enables the study of contribution asymmetry free of any bias introduced by the underlying market protocol.

In this idealistic sharing model, the topology of the underlying friendship graph plays a central role in the operational requirements of a social cloud. The social graph governs the interaction between pairs of users and determines the storage space to be contributed by them. Although users with many friends have more chances of storing their data with higher availability, they may possibly have to donate more disk space to socially reciprocate a larger number of friends. This is especially visible for those users with higher degrees, usually called *hubs*, whose level of contribution may be very high for comparatively little benefit.

From a global perspective, it is not hard to imagine that the degree distribution of the friendship graph is one of the main factors impacting the system operation. To better understand this, pretend that two users, say a and b, want to store 3 data blocks each. Also, assume that they have a friend in common, say c. Depending on the number of friends, then a and b will store more or less data blocks in c. If a and b had two additional friends, then c would need to store only 2 data blocks, one from a and one from b. However, if c was the only friend of a and b in the social cloud, $3 \cdot 2 = 6$ blocks would be allocated to c. This shows the importance of social connectivity on contributed storage, specially for hubs.

In addition to the graph degree, we make use of the clustering coefficient (CC) to measure to what extent the social links in the friendship graph tend to cluster together. The local CC of a user v is defined as:

$$CC_v = \frac{2 \cdot E_v}{D_v(D_v - 1)}, \tag{1}$$

where E_v is the number of edges between neighbors of v and D_v the degree of user v. Loosely speaking, the CC_v quantities to what extent the neighbors of v are linked to one another. In our tests, we will mainly use this metric to study the contribution level of hubs.

Real measurements of social networks [7,4] show that while clustering is very high, the presence of hubs is characteristic of social interaction. Understanding the influence that graph properties have on the extent of storage contribution is crucial to decide the suitable market metaphor to regulate sharing, mainly for hubs.

4. Social storage with FriendBox

To give form to the definition of "social cloud storage" and determine what aspects should be integrated into its definition, we have employed our social cloud storage application, called FriendBox [12], which has been developed and deployed as a Facebook application. We chose Facebook for its popularity, development environment and API, and very importantly, because Facebook identification allows users to define policies regarding who can store and access their personal data. For example, a user could limit the sharing of their data with close friends only, or users in the same group. This gives individuals high control over their data, engendering trust and some level of accountability, properties that are hard to find in a cloud environment. From a privacy standpoint, while Facebook learns the interactions between the members of the social cloud, personal information is never revealed to this online service, as it is stored and shared through peer-to-peer exchanges.

A distinctive feature of FriendBox is that lets a user add an external cloud storage service like Amazon S3 to its social cloud in order to improve the availability of its data. By no means this signifies that all data will be stored to the cloud. Following the spirit of the social cloud approximation, FriendBox lets the user decide the amount of data to be stored in the cloud, which can be zero if the user wishes so. This feature is particularly useful, as it allows to trade data availability for monetary costs and adapt the storage service to the user needs.

Further, the use of the cloud requires another layer of preprocessing the data in order to protect it from unauthorized access, disclosure and theft. This could be accomplished in many ways. A simplistic approach could be to encrypt each sensitive piece of data and share the key with the authorized users. Instead of this simple encryption scheme, we use Reed–Solomon codes [9] for that purpose, blending storage efficiency [24] and privacy in a single scheme. Other approaches would be equally possible with no significant changes in the proposed method. However, we do not want to involve ourselves in this question here, since our focus is on analyzing the feasibility of this new storage model.

In what follows, we will describe the components of FriendBox, whose general architecture is illustrated in Fig. 2. We will give the essential details to make our results understandable and refer the reader to [12] for full details.[3]

4.1. Social front-end: Facebook application

In our social cloud, the storage overlay is bootstrapped by the underlying social structure. Accordingly, every node in the friendship graph acts as a storage service to their adjacent neighbors. In practice, the friendship graph can include members of the family, close friends only, or even friends of friends, which can be viewed as directly connected to each user that selects them as storage servers.

As social substrate, FriendBox uses Facebook for user management, because Facebook exposes access to their social graph through a simple API, called the Graph API.[4] This API exposed through a REST service gives access to many objects, including friends, profile information, groups and photos. To control access to the Graph API, Facebook utilizes the OAuth protocol [25] to authenticate both users and applications. This authorization model allowed FriendBox to delegate access control to Facebook, simplifying considerably user management and accountability.

The integration of FriendBox with the Facebook look and feel was by means of the Facebook Markup Language (FBML). FBML includes a subset of HTML with proprietary extensions that enables the creation of applications that follow the Facebook style. Code written in FBML is retrieved by the Facebook server, parsed, and then inserted into their surrounding code. This facilitated the creation of a familiar and intuitive GUI for FriendBox. Through this GUI, the user can keep track of its monthly storage consumption in the cloud provider of its choice and the distribution of its data within the social cloud, among other operations. Such state information is maintained in a separate component called Application State, which we discuss in the following section.

4.2. Application state

Essentially, the Application State maintains up to date the data management information about any file stored in the system. This information includes the specific set of friends that store each data object along with the network address of each one. Without this information, the clients would be unable to perform the necessary peer-to-peer storage operations to store and retrieve any data file from the social cloud. The logic of keeping the Application State

[3] FriendBox webpage: http://ast-deim.urv.cat/friendbox/.

[4] http://developers.facebook.com/docs/reference/api/.

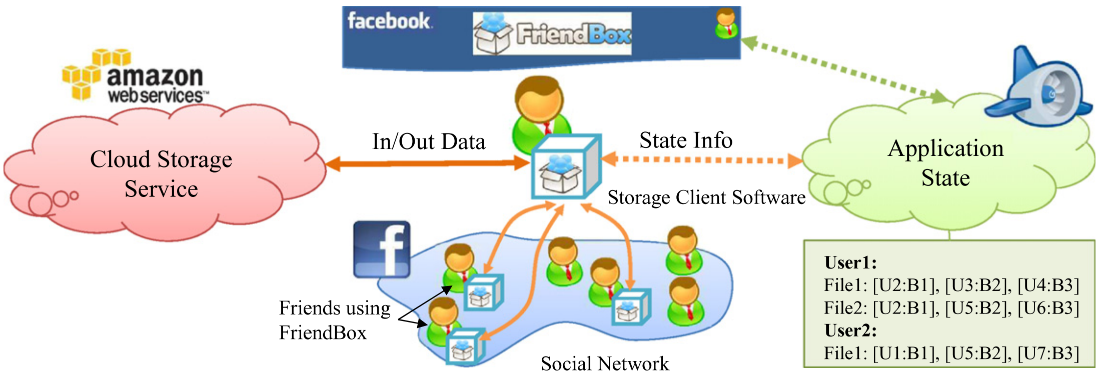


Fig. 2. A user maintains storage links with some of his friends in Facebook. Moreover, this user is able to store a fraction of his data in a cloud storage service. The state information of a user's data is stored in the FriendBox Application State. Finally, users manage their storage relationships and check the state of their storage service in the FriendBox Social Front-end.

current lies on the desktop clients themselves. The clients update the Application State via a REST API.

The role of the Application State is depicted in Fig. 2. In this figure, we show how a user communicates with the Application State to transfer state information. In this example, a user sends a message informing that a new file has been stored in the system. As shown in the figure, Application State stores this information using mappings that relate data blocks with the friends who are responsible for them.

The Facebook application code for FriendBox along with the Application State is nowadays hosted in Google App Engine.[5] The reason for this choice was that this PaaS for developing web applications offers elasticity in the service. Note that if we wanted to protect the metadata from possible threats such as theft, unauthorized access, and copying, an additional layer of protection would be indeed necessary. One simple way of doing this would be to encrypt the metadata before storing it in Google App Engine. This issue is, however, beyond the scope of this paper.

4.3. Desktop client

In addition to the integration with Facebook, a social cloud storage application necessitates a desktop client to store and access remote data. To efficiently achieve the desired level of data availability, FriendBox lets users select the set of friends to store each content and decide which part of the data should go to the cloud. In the current version of FriendBox, the desktop client only permits to store data in Amazon S3, though other cloud storage services like Windows Azure and Google Drive can be easily supported.

To achieve high availability, the best strategy would be to store all data in the cloud to guarantee 24/7/365 access availability. However, at \$0.120 per GB of data transferred out of the cloud, these costs might quickly add up. To decrease monetary costs, FriendBox uses the friends in the social cloud to store data but at the expense of a lower data availability. The fundamental idea behind FriendBox is to provide data availability during the hours of the day where friends are mostly logged in to benefit from availability correlations. We introduced this new notion of data availability, termed daily data availability, in our recent work [8], for we refer the reader to [8] for further details. Going back to our formulation in Section 3.1, a user may want to achieve a daily data availability of δ time units for its data. By viewing daily data availability D as a subset of T_{day}, i.e., the set including all the time units of one day according to a particular quanta, D contains those time units of T_{day} being covered by at least one friend, and preferably those with a greater number of friends. The reason is that a greater number of friends supply more flexibility to allocate data for load balancing.

[5] http://code.google.com/intl/en/appengine/.

4.4. Data redundancy and privacy

To maintain the desired level of data availability, it must be carefully decided the degree of redundancy. While replication is suitable for storage of small objects that are accessed frequently, we use Reed–Solomon codes (RS) [9] for storage space efficiency [24]. Given a data object of size B, a RS(n, k) code partitions the data object into k equal-sized fragments, each of size B/k bits. These k fragments are then encoded to a set of $n = k + h$ redundant blocks. Since this code is a maximum distance separable (MDS) code, the stored object can be reconstructed from any k-subset of redundant fragments. The consequence of this property is that a RS(n, k) code can tolerate the loss of any $h = n - k$ blocks with a redundancy ratio of only n/k. For instance, if we split a file into $n = 14$ blocks so that any $k = 10$ blocks suffice to reconstruct the original file, we can tolerate 4 failures with a storage-space overhead of only 40%. If we had used instead replication, we would have needed 5 replicas to achieve the same level of fault tolerance, yielding a storage-space overhead of 400%. The use of coding is thus highly desirable in this environment where the social ties storing the data will not be available at all times.

Another important advantage of Reed–Solomon codes is that the generator matrix of the code can be chosen to be non-systematic. If a code is non-systematic, then the original data fragments will not appear in the code, preserving data confidentiality. Note that this statement is valid provided that no subset of blocks of cardinality greater than $k - 1$ is in the hands of a non-authorized party, which in our case is basically the cloud provider. We must note, however, that while a non-systematic RS code is a (k, n) threshold scheme, and can be interpreted in terms of Shamir's secret sharing [11], its security guarantees are less than Shamir. The reason is the lack of randomness in the generator matrix of RS codes. So, attackers looking for known or patterned data can find it more easily without reconstructing the original data [26]. For FriendBox, this level of protection is sufficient. A higher protection level can be simply achieved by first encrypting the data and then encoding it, or by using more elaborated dispersal schemes such as AONT-RS [26]. Since all of these variants also transform a file into n distinct blocks, our analysis is equally valid for all of them.

Once explained the advantages of Reed–Solomon codes, we are now ready to discuss how we distribute the encoded data objects across the social ties and the cloud service. Concretely, after applying the RS coding scheme, a fraction F_C of the encoded k fragments is allocated to the cloud. The remaining $\lceil (1 - F_C) \cdot k \rceil + h$ blocks are allocated to the social friends in a round robin fashion to achieve an even use of their disk capacity. Compared with replication, one of the most valuable assets of RS codes is that the amount of data assigned to a friend is typically only a fraction of the original file size, saving significant storage space.

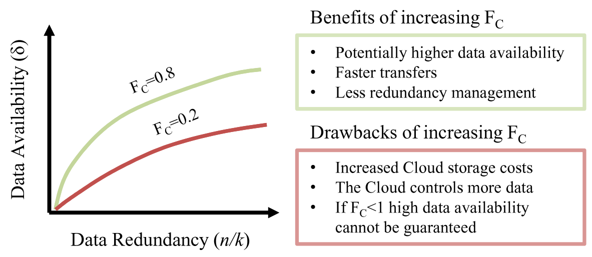

Data Availability (δ)
Data Redundancy (n/k)
F_C=0.8
F_C=0.2
Benefits of increasing F_C
Potentially higher data availability
Faster transfers
Less redundancy management
Drawbacks of increasing F_C
Increased Cloud storage costs
The Cloud controls more data
If F_C<1 high data availability cannot be guaranteed

Fig. 3. Implicit trade-offs between data availability, redundancy and cloud costs in FriendBox.

It is important to mention here that the exact value for F_C depends on the parameter δ and the connection pattern of the friends in the social cloud. For instance, let us consider we want to cover δ time units of data availability. Depending on the number of online friends at each of these time units, the storage requirements and the appropriate value for F_C will vary. To illustrate this, we consider two extreme cases. At one extreme lies the case where one of the δ time units is covered by a single friend. In this case, in order to ensure the reconstruction of the object, this single friend will be forced to store at least $k - \lfloor F_C \cdot k \rfloor$ out of the n blocks. And here the chosen value for F_C makes a big difference. The reason is that the value of F_C determines the storage requirements for this friend, which will be maximal and lead to the storage of a complete replica of the data file for $F_C = 0$. At the other extreme is the case that all δ time units are covered by at least $k - 1$ friends, requiring to store only one block in the cloud. In this case, however, a small value of F_C will be not so problematic, because the storage capacity contributed by each friend will be significantly smaller: just one block. Hence, a high level of correlation in availability patterns can help to reduce the fraction F_C for a fixed δ.

For completenesses, Fig. 3 illustrates the relationship between data availability δ, the redundancy ratio n/k, and the fraction of data allocated to the cloud F_C, which are the three parameters of our storage model. Let us first consider that the redundancy ratio n/k is kept fixed. In that case, the result of increasing F_C by pushing more blocks to the cloud is that data availability increases. This suggests that by choosing the right F_C, one can achieve the same data availability with less redundancy. Consequently, a user will experience shorter transfer times and he will require less resources from his friends. However, increasing F_C may present some drawbacks, specially related with a higher cost of the storage service and the amount of data control relinquished to the cloud operator. Furthermore, even in the case of storing $k - 1$ blocks in the cloud, 100% availability cannot be guaranteed: *If all storage friends are simultaneously unavailable, the missing block will not be reachable* [8]. FriendBox gives to the user the opportunity to decide the most adequate storage service depending on his needs.

4.5. Maintaining data redundancy

In FriendBox, data blocks may be lost in the event of a user permanent departure/crash or when the friendship between two users comes to its end. Therefore, to maintain an adequate level of data availability over time, we should repair lost redundant data blocks.

In our context, a *data repair* consists of generating a new redundant block to replace a lost one,[6] storing the generated block again in the system [27]. There are three main approaches used to recreate redundancy when nodes fail:

- *Eager repairs*: Lost redundancy is repaired on demand immediately after a node failure is detected.
- *Lazy repairs*: The system waits until a certain number of nodes had failed and repairs them all at once.
- *Proactive repairs*: The system schedules the insertion of new redundancy at a constant rate, which is set according to the average node failure rate.

FriendBox advocates for an *eager repair* strategy. In other words, when the friendship between two users ends, the FriendBox *Application State* requests both users to start a repair process, i.e., a *graceful repair*. Moreover, if a friend of a user has been disconnected from the system for a certain time period, e.g., 1 week, it can be assumed that this disconnection is a permanent departure and trigger a repair.

Compared to lazy repairs, the number of a user's trusted friends is normally too small to wait for several nodes to fail until triggering a repair. Actually, this condition might be never reached, leaving the system in a low-redundancy state for a long period of time and thus vulnerable to data loss. Also, proactive repairs are not directly applicable because users fear to store their data in arbitrary nodes in the system. Further, to put proactive repairs in place, it is necessary to compute summary statistics like the average failure rate in a per-user friendset basis. Due to the small sample of nodes, these statistics cannot be considered reliable enough to give a correct inference for proactive repairs to work as expected.

A final concern is that introducing more redundancy to the remaining friends is no guarantee to improve data availability in the presence of availability correlations [8]. Therefore, since repairs are expected to be rare, we advocate for placing the data block generated from the repair process in the cloud. The objective is to provide the same level of data availability the system enjoyed prior to the failure.

4.6. Data transfer

Once a file has been encoded, it is necessary to transfer the encoded blocks to the corresponding social ties. To minimize transfer time and fully utilize the upstream bandwidth, FriendBox uses the rented cloud storage service as a temporary repository to store the blocks for those social links that were offline when the transfer of their blocks was scheduled. In any case, the extra blocks pushed to the cloud are downloaded afterwards by the friends to whom they were initially allocated.

For downloading a file, FriendBox prioritizes the download of the corresponding blocks from friends to incur the minimal monetary costs due to the data transfers out of the cloud. Only in the case that there are less than k blocks, the remaining up to k are downloaded from the cloud storage service.

5. Empirical analysis of social cloud storage

In this section, we empirically study *the fundamental problems and challenges involved in the social cloud storage paradigm*. Indeed, what the incipient social cloud literature misses is a deep analysis of the implications that environmental factors such as user availability and topology have on the storage service. As a central contribution of this work, we identify and quantify the main underlying factors that influence the storage service provided by a social cloud.

5.1. Evaluation objectives and metrics

Through experimentation, we aim to shed some light on the following aspects that we believe capital to provide an adequate storage service in a social cloud:

[6] Reed–Solomon codes require to gather a subset of k blocks to generate a new redundant one. This may induce high bandwidth overhead in scenarios where failures are frequent. Making use of more sophisticated codes in FriendBox (e.g. regenerating codes) and analyze the overhead data repairs is object of future work.

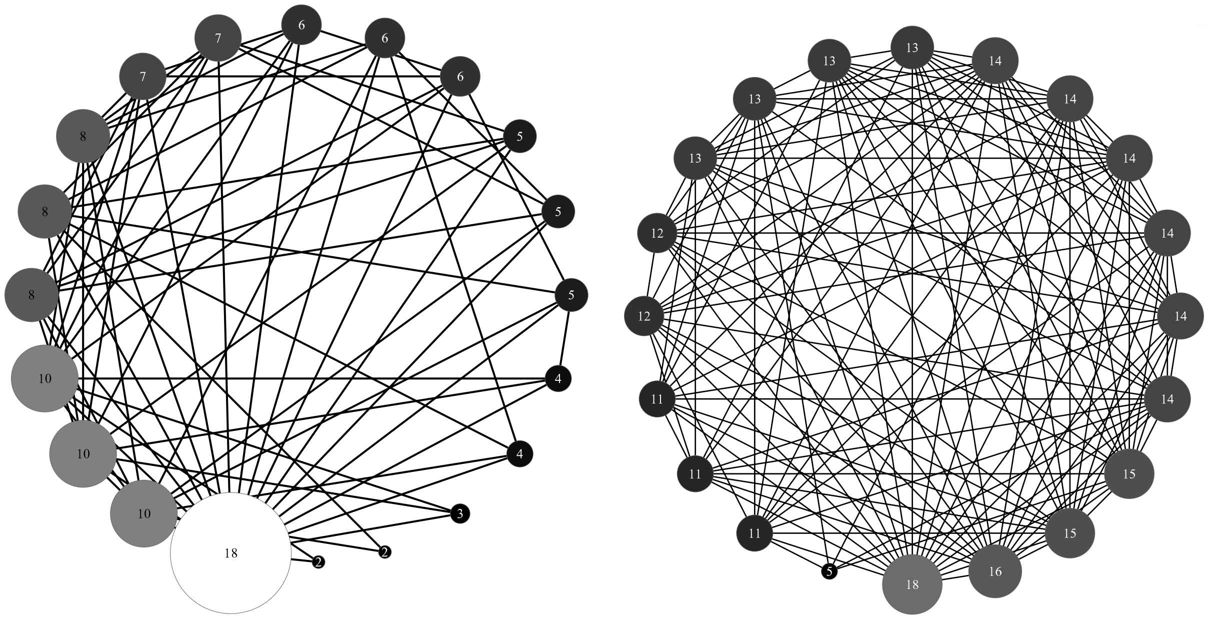


Fig. 4. Input social graphs for our experiments. The graph on the left exhibits a low average clustering coefficient of $CC = 0.3$, whereas the CC of the graph on the right is 0.7. Node labels correspond to their degree.

- *Daily data availability*: The probability to access a data object during the day, which depends on parameters such as the amount of redundancy $\frac{n}{k}$ and the fraction of the data allocated to Amazon S3. Of course, correlation in availabilities plays a key role on the achievable daily data availability.
- *Load balancing*: Load balancing is critical to the feasibility of a distributed storage system [8]. For this reason, we analyze the interplay of the social graph topology and user availability on the resulting storage load supported by users.

  We quantify load balancing in two ways. At the user level, we account for the number of storage operations processed by each user, i.e., data block PUTs and GETs. At the global level, we utilize the Gini coefficient and the Lorentz curve to examine the distribution of served storage operations in the social graph. Specifically, the Lorenz curve depicts the proportion of the total income of the population (y axis) that is cumulatively earned by the bottom x% of the population.[7] The diagonal line represents perfect equality of incomes. The Gini coefficient, denoted by G, is the ratio of the area that lies between the line of equality and the Lorenz curve (A) over the total area under the line of equality ($A + B$):

$$G = A/(A + B). \tag{2}$$

- *Transfer time*: An important performance metric for social cloud storage is data transfer speed. In particular, we study two aspects: the congestion caused by the topology of the social network and the impact of correlated user availabilities on the time to download a file from the system.
- *Fairness*: Typically, a social cloud adds regulatory protocols to enforce resource fairness. However, there is no analysis on the extent of the potential asymmetry that may arise in a social cloud along with what elements may originate it. As a simple measure of fairness, we utilize the ratio between the amount of resources contributed to the social cloud and those consumed by a user:

$$FR = \frac{R_p}{R_c}, \tag{3}$$

  where R_p represents the amount of resources a user provides to the system, and R_c the amount of resources that a user consumes from his social ties. A value of FR equals to 1 represents perfect equilibrium between resource consumption and contribution. $FR > 1$, however, means that a user is contributing more resources to the system than what is actually consuming. Finally, $FR < 1$ signals that a user may be abusing its social ties, because it consumes more than it donates.
- *Cloud contribution*: As we use cloud storage, i.e. Amazon S3, as a pivotal element to the feasibility of a storage service in a social cloud [12], its role in the system deserves special attention. Indeed, we measure the consumption of cloud resources that the members of the social cloud incur in their PUT and GET storage operations, depending on their availability and position in the social graph. We use the number of data blocks transferred in and out of the cloud because this simple metric can be immediately turned into monetary metrics like the ''dollars per storage operation''.

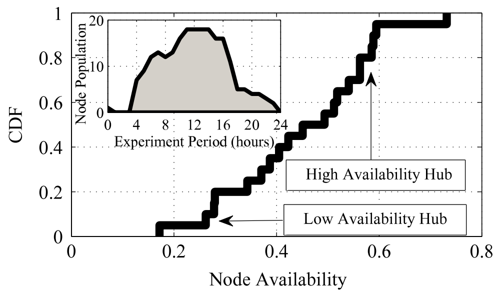
5.2. Scenario and setup

Once elaborated on the objectives of our evaluation, we are ready to describe the setup of our experiments.

Topology. We deployed a group of 20 FriendBox desktop clients in our university laboratories. The 20 FriendBox clients were organized according to two real graphs from Friendster [28] in order to assess the influence of the friendship topology. One topology shows a high clustering or local transitivity, i.e., if user a knows b and c, then b and c are likely to know each other, while the other is weakly clustered. To identify each topology, we will use the value of the clustering coefficient at the hub. Both topologies are illustrated in Fig. 4. Accordingly, their degree distributions are shown in Fig. 5 (right).

Availability. To incorporate availability correlations into our experiments, we instrumented the alternating ON–OFF behavior of users by means of an availability trace from Skype [29], which exhibits strong diurnal patterns and high heterogeneity in user availabilities. Both properties are clearly visible in Fig. 5 (left). The CDF of user availabilities ranges from 0.18 to 0.75, which evidences high heterogeneity. Furthermore, the time-series representation in

[7] For a technical description of Gini coefficient and Lorentz curve see http://en.wikipedia.org/wiki/Gini_coefficient.



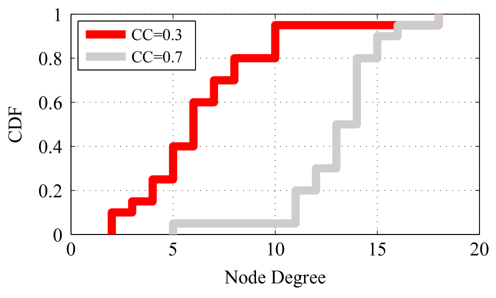
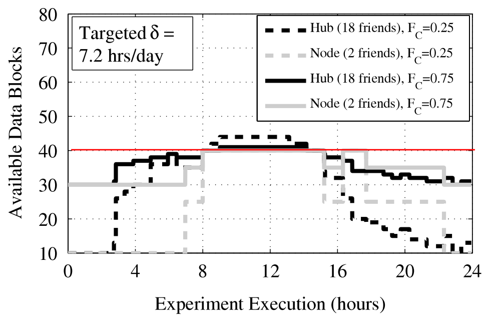


Fig. 5. Nodes present high availability heterogeneity and diurnal patterns (left). The node degree distribution varies significantly depending on the *CC* (right).



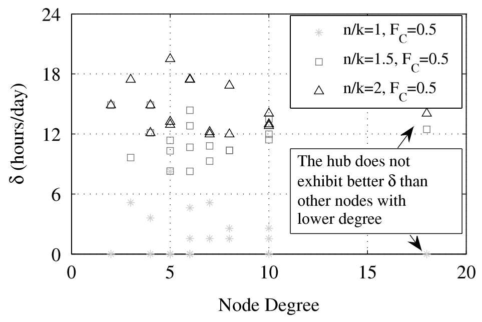


Fig. 6. Time series plot of the available blocks for the hub and the least connected node to achieve $\delta = 7.2$ hrs./day (left). Impact of increasing n/k on δ depending on the node degree (right).

Table 1
Parameter configuration in our experimental scenario.

Parameter description and values	
Nodes in the system	20
Experiment duration	24 h
Node storage capacity	40 GB
Parallel upload/download connections	2, 2
Erasure codes original file fragments (k)	40
Cloud file fraction (F_C)	0.5
Object size (β)	400 MB
Data redundancy (n/k)	2.0
Cloud back-end	Amazon S3

the inner plot illustrates that friends are mostly connected during the central part of the day and disconnected during night hours.

To study the impact of availability in the social hub, we assigned two different availabilities to the highest degree user in the social graph: a high availability of 0.594 and a low availability of 0.278.

We also conducted simulations where users were always online as baseline to understand the effect of availability correlations. We will refer to this scenario as "no churn" throughout the evaluation.

Workload. The workload model of our experiments is homogeneous. All nodes alternatively perform file downloads and uploads while being logged in. Hence, file transfers are concurrently executed throughout the experiment to capture the effects of network congestion. File transfers are randomly performed every period of [600–1200] seconds over synthetic files of size $\beta = 400$ MB. Unless otherwise stated, we fixed $F_C = 0.5$ and the redundancy ratio to $\frac{n}{k} = 2$.

Hardware. FriendBox clients were hosted in desktop computers (Intel Core2 Duo and AMD Athlon X2 processors) equipped with 4GB DDR2 RAM. The OS was Debian Linux.[8] The clients were connected via a 100 Mbps switched Ethernet links. For the collection of physical network information, we utilized *vnstat*, a tool that keeps a log of network traffic for a selected interface. The rest of information presented in this section was gathered by the FriendBox log system.

Other important parameters in this experimental scenario are depicted in Table 1.

5.3. Experimental results

Here we present the experimental results and describe the main insights that follow from our analysis of the social cloud storage.

5.3.1. Data availability

In this section, we study the factors that influence the daily data availability. For this reason, we fix the target daily data availability δ to 7.2 h and vary the fraction F_C of data to be allocated to the cloud. For clarity, we only report the results for the topology with small clustering. Also, we only consider the case where the social hub is highly available.

The effect that availability correlation induces on daily data availability can be clearly seen in Fig. 6, left. Surprisingly, the least connected user achieves the target 7.2 h of data availability by making use of less redundancy than the social hub. This can be easily inferred by tracking over time the number of data blocks available for each user. The cause of this counterintuitive behavior is availability correlation: the two friends of the least connected user are simultaneously online for ≈8.5 h. Because they cover by far the target 7.2 h of daily data availability, no extra redundancy is necessary. In general, however, it is difficult to have a sufficient number of online friends for δ hours, which requires the introduction of extra redundancy to meet the target level of data availability.

Further, Fig. 6, left, gives an interesting result: the allocation of a larger proportion of data to the cloud makes it possible to achieve the target 7.2 h of data availability with less redundancy. This is because a larger F_C reduces the number of data blocks to be given to friends. Since altogether friends exhibit poor availability compared with Amazon S3, the necessary redundancy to meet a certain δ may become smaller. This occurs to the social hub whose redundancy ratio $\frac{n}{k}$ decreases by a 14% when increasing F_C from 0.25 to 0.75. These savings become more significant for higher δs.

The dispersion graph in Fig. 6, right, relates the number of social links (x axis) with the achievable δ (y axis) for different amounts

8 FriendBox works for other platforms such as Windows and Linux Ubuntu.

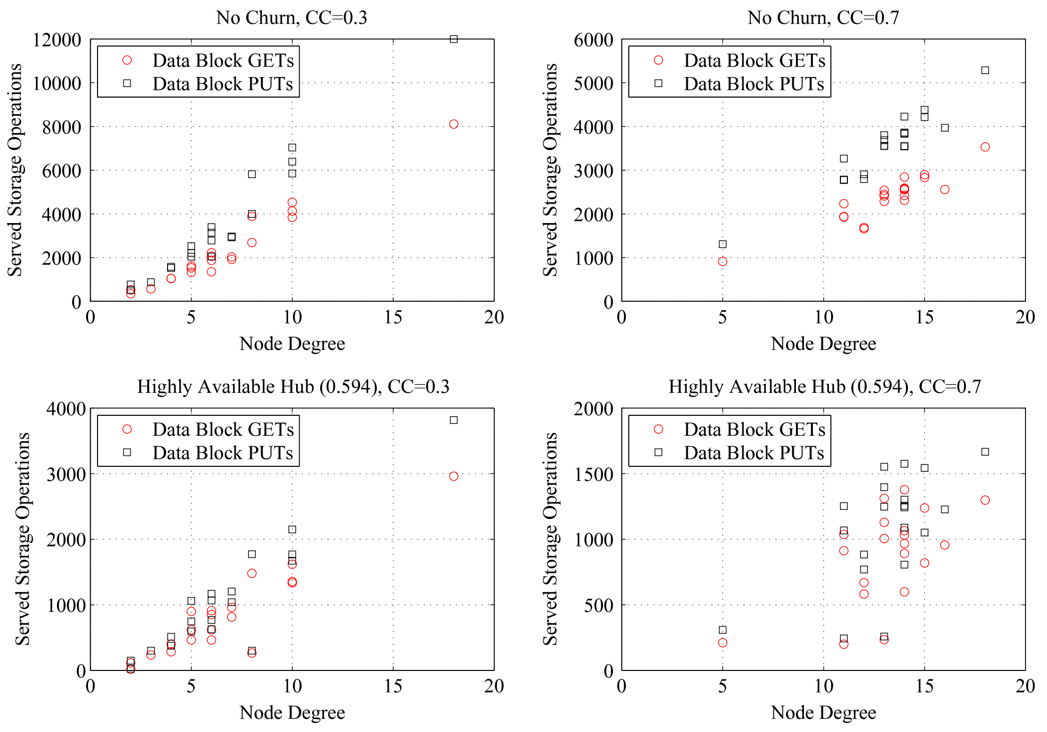


Fig. 7. Relationship between a node's degree and the storage load caused by its friends. We illustrate a churn scenario (high available hub) and a stable scenario.

of redundancy $\frac{n}{k}$. As expected, the higher the redundancy is, the higher the data availability is. However, the increase in data availability is not linear and may be abrupt or even zero for a higher $\frac{n}{k}$. Concretely, the final data availability depends more on the availability pattern of users than on the number of friendship links a user has. This is evidenced by the lack of correlation between the node degree and the achievable δ. In fact, some users with a smaller number of friends present a higher δ than those users with a larger friendset.

We can summarize the main findings of this section as follows:

Observation 1: *A larger number of friends help but do not necessarily improve daily data availability.*

Observation 2: *The degree of coincidence in the online periods of friends is crucial to understand the relationship between data availability and redundancy.*

Observation 3: *Storing a fraction of data in the cloud may reduce the overall redundancy of a social cloud system.*

5.3.2. Load as function of social graph topology

Here we examine the influence of the graph topology on the load experienced by users. In Fig. 7, we report the number of data blocks that a user stored (PUT) and served (GET) as a function of its degree. The figure contains four subplots, each of which corresponds to a distinct combination of topology and availability model. Interestingly, all four dispersion graphs show that the load of users varies significantly depending on the clustering of the social graph topology. For high clustering, load is more evenly spread across all users, irrespective of the availability model.

For low clustering topologies, however, the degree strongly determines the load of a user. This conclusion comes from the visible linear growth on the number of storage operations with increasing user degree. Such a behavior may compromise the scalability of a social cloud. Social hubs, which interact with most of their social links [4], may become eventually saturated, and socially-based incentives may be even insufficient to enforce cooperation in the social cloud. This may pose the need for more sophisticated trading and sharing strategies like auctions and formal SLAs.

To examine load balancing from a global view, we calculate the Gini coefficient to measure the inequality in serving GET operations. The corresponding Lorentz curves are shown in Fig. 8. As shown in this figure, the Gini coefficient is much smaller and the Lorenz curve much closer to the diagonal in the topology with high clustering, which indicates that a higher connectivity facilitates the balancing of load among the members of the social cloud. But more importantly, and contrary to conventional wisdom, there exists no correlation between the load and the user degree in the presence of availability correlations. This phenomenon can be easily seen in the lower right subplot of Fig. 7, where users of similar degree present very disparate load. We explore this issue in the next section.

We summarize the main results of this section as follows:

Observation 4: *For low clustering, the degree strongly determines the load of a user.*

Observation 5: *In general, a high clustering coefficient results in a better load balancing within the social cloud.*

5.3.3. Load as a function of user availability

Let us now consider the traffic load a user encounters as a function of its availability. The dispersion graphs in Fig. 9 relate these metrics for both stored and served blocks in a dynamic scenario with different clustering values.

The first main observation is that user availability does not positively correlate with storage load when the degree of clustering is low. This result is important because conventional wisdom assumes that high user availability is synonym of a higher burden. However, we observe that load in a social cloud system depends on other factors like the specific topology of the social graph. Concretely, we find that for low clustering, the number of friends that a user has is what determines its storage load.

On the contrary, when the social graph is highly interconnected, availability is what mainly determines the storage load experienced by users. This conclusion is evidenced by the linear increase in the number of data block transfers with increasing user availability. This result is not surprising. In the ideal case that all the members of the social cloud were fully connected, the burden

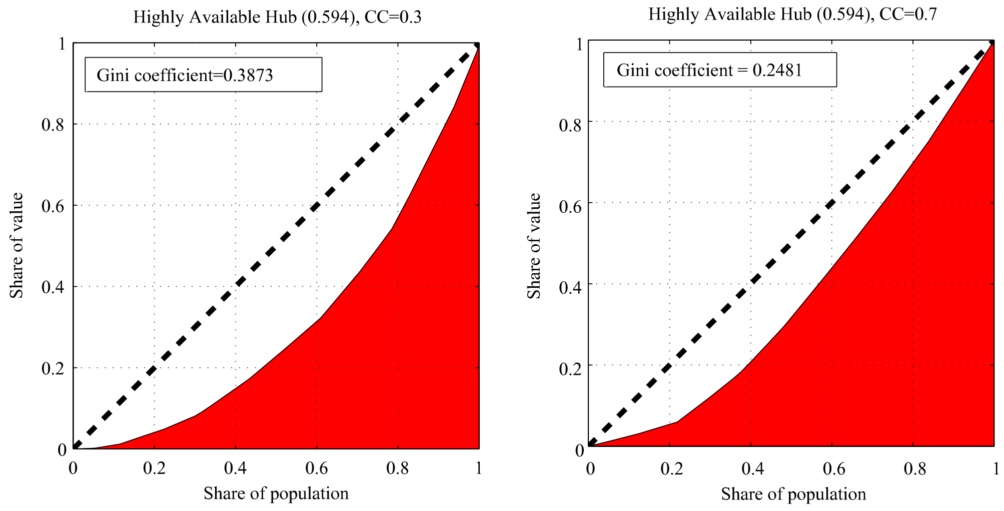


Fig. 8. Distribution of served download block requests (GETs) in a churn scenario depending on *CC*.

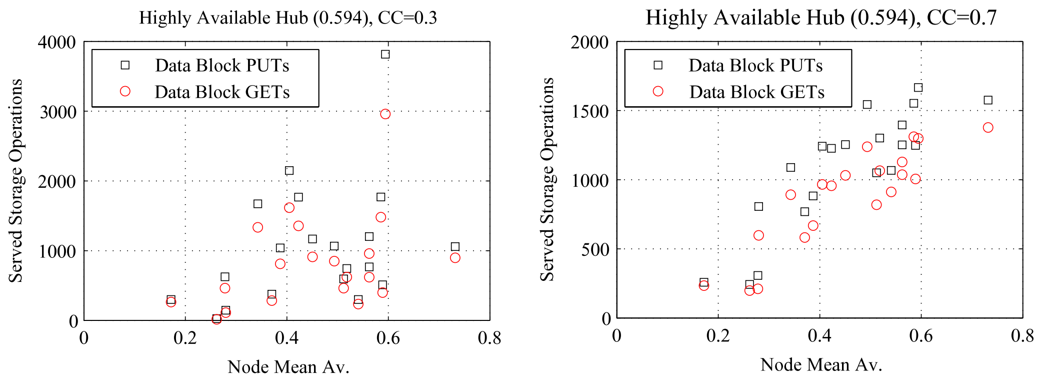


Fig. 9. Relationship between storage load and node availability depending on the clustering coefficient.

experienced by each individual would be proportional to its availability: the higher the availability the greater the odds of undertaking a storage PUT and GET operation.

We summarize the main insight of this section as follows:

Observation 6: *In a social cloud with high clustering, the availability of a user determines the load it will receive.*

5.3.4. Data transfer time

First, we assess transfer speed as a function of the social graph topology. To avoid any interference caused by availability correlations, Fig. 10 depicts the distribution of transfer time when all the users in the social cloud are online, i.e., when there is no churn. For clarity, we only plot the transfer time distribution for three users: the social hub who is linked to 18 friends, a user with the average network degree, and the least connected user in the social cloud.

For the low clustering topology, two observations are specially interesting. First, the least connected user achieves a lower transfer time than its higher degree friends, particularly for downloads. This is explained by the fact that for low clustering topologies, the users with many social links support a higher storage load and suffer from congestion. Second, the differences in the upload time are less significant. This is mainly due to two factors. First, local data block transfers among friends are much faster because of our Fast Ethernet LAN than accessing Amazon S3. Second, uploading in FriendBox involves the transfer of a fraction of the data to Amazon S3 while downloads retrieve as much as possible from friends and only access the cloud if there are not enough blocks available at friends.

For the high clustering topology, however, there are no important differences in file transfer times neither for uploads nor downloads. This means that a higher clustering coefficient introduces less congestion.

Now we study the effects of availability correlations on download times. For such a purpose, Fig. 11 plots the download time given as a time series for the social hub and one of the users whose degree coincides with the average degree of the social graph. For the social hub, Fig. 11 (left) reports that the download time is short when most of its friends are logged in. However, this time increases significantly during night hours. This is because the hub needs to resort to the cloud in order to complete the file download, which makes downloading to be slower in our campus scenario.

For the average-degree user, Fig. 11 (right) reports a larger download time than for the social hub, which indicates that the download time diminishes with the number of friends since blocks transfers from friends are faster than accessing the cloud. This is supported by the fact that for the same node, in most cases, a higher degree induces shorter download times.

Finally, it is worth mentioning that in some cases, specially at the end of the regular node execution, a few file downloads when that node has 13 friends are slower than when it has only 5 friends. As in the case of data availability, *a higher degree reduces download times if friends are simultaneously online at the moment of downloading the content*. Otherwise, a higher degree will have little or no positive effect for the storage service a node receives.

We summarize this section as follows:

Observation 7: *For low network clustering, users with high degrees exhibit larger transfer times due to network congestion. This can be critical for hubs.*

Observation 8: *A higher clustering coefficient inherently reduces congestion and improves transfer times.*

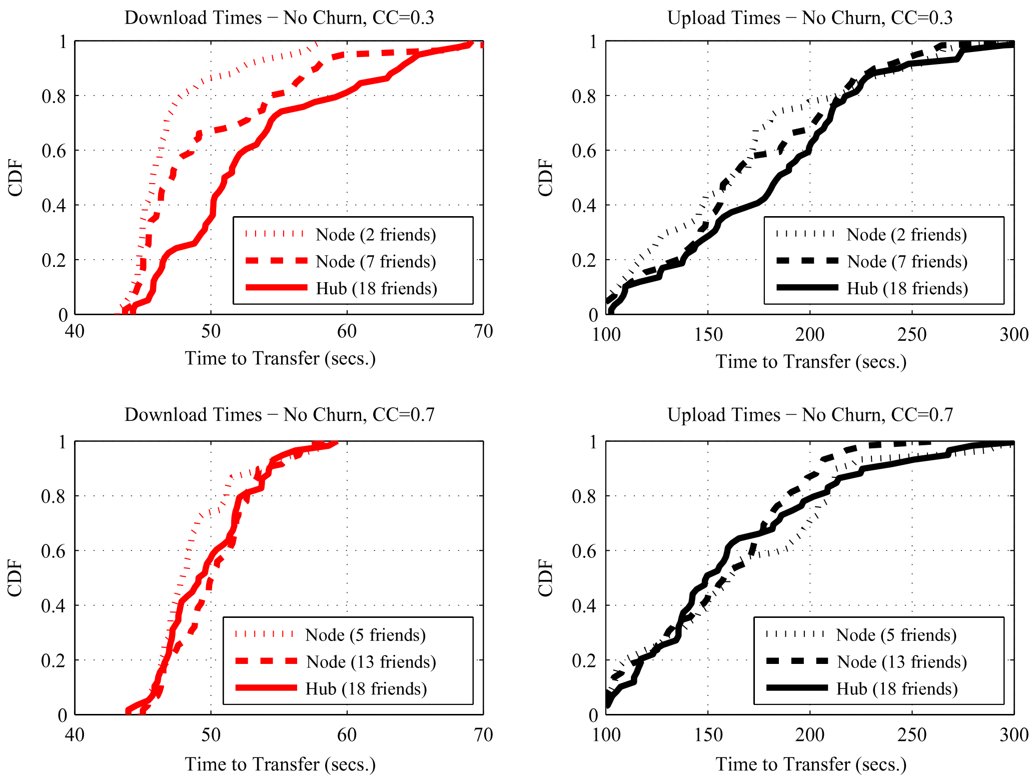


Fig. 10. Effects of *CC* on transfer times and congestion.

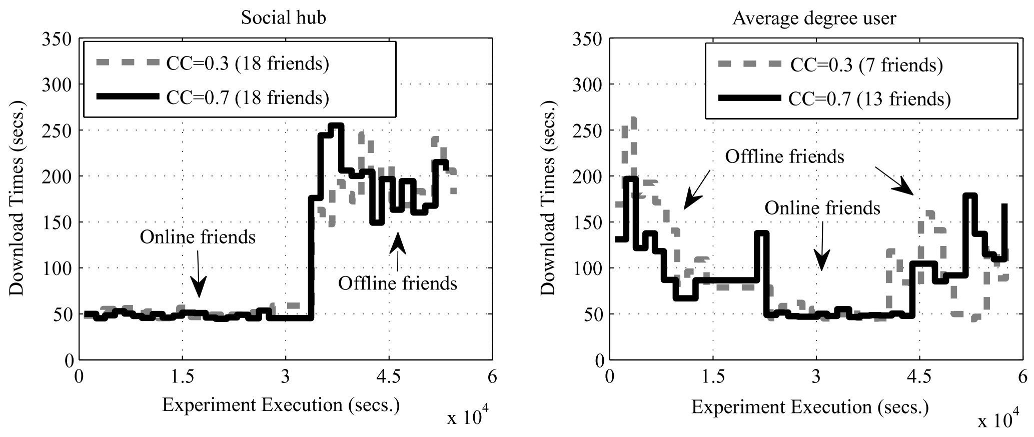


Fig. 11. Time series analysis of download times of two different nodes. We clearly observe the consequences of availability correlations on download times.

Observation 9: *Although having more friends may in general improve download times, the actual number of online friends when the download occurs is fundamental.*

5.3.5. Fairness

Now we study the resource fairness among members of the social cloud. We use the fairness ratio (*FR*) as defined in (3) to measure the asymmetry in resource contribution. To start with, we focus on the fairness in bandwidth contribution. As a boxplot allows to assess the dispersion of a given distribution, Fig. 12 shows the boxplots of the distribution of fairness ratio when the resource under consideration is the upstream and downstream bandwidth.[9] As can be seen in the figure, a high clustering is crucial to promote fairness. For the topology with small clustering, around 70% of the users consume more resources than they contribute. This forces the remaining 30% to correct this deficit and contribute the missing resources for little or no personal gain. Some users even present a *FR* superior to 2, which may be a powerful disincentive for many users to remain in the system.

For the topology with high clustering, however, the boxplots resemble a normal distribution centered at the equilibrium point of $FR = 1$. This is very positive for the system, as it means that most users consume an amount of resources that is equal to their individual contribution.

Next, we investigate the influence of user degree on the fairness ratio. More specifically, Fig. 13 correlates the fairness ratio with user degree by means of several dispersion graphs. As before, this figure contains four subplots, each corresponding to a single combination of topology and availability model.

As can be seen in the figure, and contrarily to our prior observations, the user degree is the dominant factor controlling local

[9] In our experiments, the application workload is homogeneous, which means that asymmetry arises as a result of topological variations.

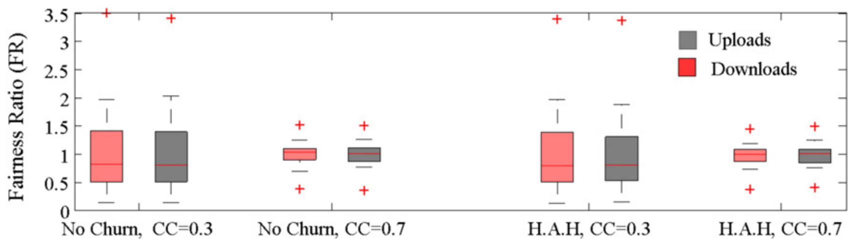


Fig. 12. Fairness ratios of up./down. transfers depending on the network's *CC* for stable/churn scenarios.

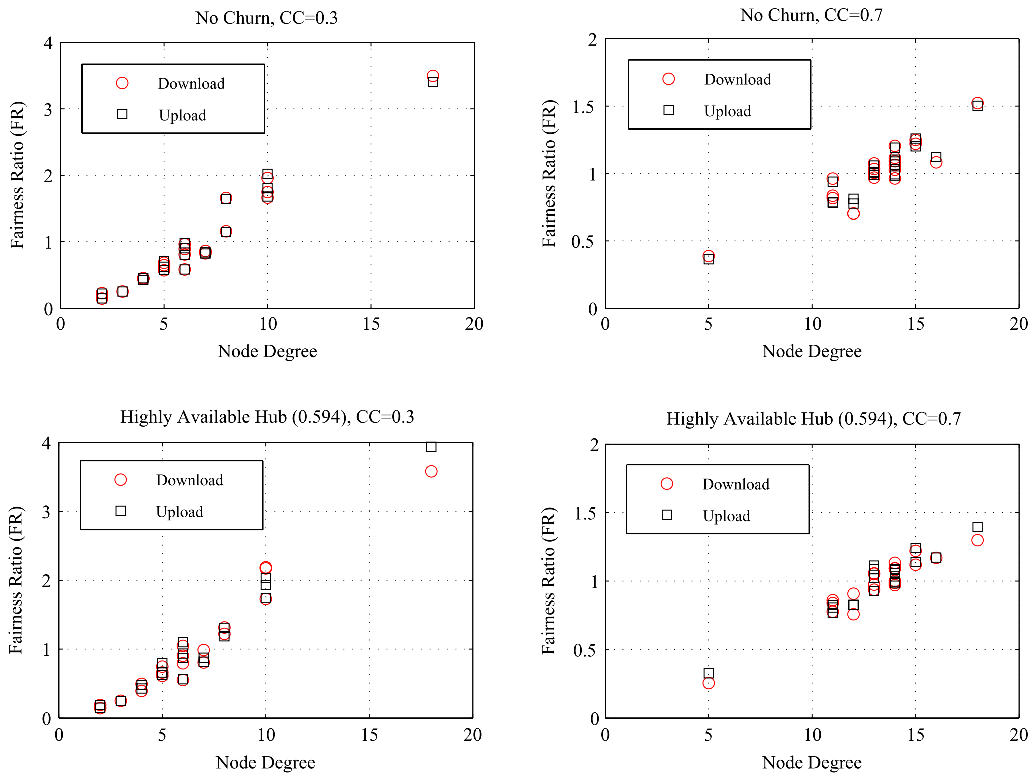


Fig. 13. Relationship between up./down. fairness ratios and node degree for churn/stable scenarios.

fairness: the higher the degree is, the higher the asymmetry is, because the number of storage operations is proportional to the number of friends. Interestingly, perfect fairness is only achieved for those users whose degree is close to the average degree of the social graph, which is 6.7 and 13.1 for the low and high clustered graphs, respectively. This gives a clue about the intricate relationship between topology and fairness, whose analytical study is the object of future work.

Furthermore, the availability of friends does not affect fairness, which can be verified by comparing the subplots of Fig. 13 where users are always logged in, labeled "no churn", with those subplots where users join and disconnect from the social cloud. The main reason is that while a user is offline, no data block can be stored in the hard disk of a friend, and vice versa.

Our observations may have important implications on the behavior of users in a social cloud. For instance, given that users with a low degree tend to abuse the system, their friends may, in turn, reject to transact with them until they increase their degree. This could lead to a cold-start situation, where newcomers cannot easily be part of the social cloud. Therefore, further research is needed to guarantee resource fairness in a social cloud by taking into account the underlying system characteristics.

The main insights of this section can be summarized as follows:

Observation 10: *A high clustering coefficient is critical to maintain fairness in the system.*

Observation 11: *The degree of a user greatly determines the fairness it establishes with the system.*

5.3.6. Cloud usage and monetary costs

Finally, we study the use of storage cloud resources by FriendBox clients. Concretely, we illustrate the total number of data block transfers in and out of the cloud when the social "hub" is highly available, abbreviated H.A.H, and low available, abbreviated L.A.H, for our two topologies with clustering coefficients of 0.3 and 0.7, respectively. To avoid biasing the results, the data blocks transferred by the hub were excluded from the final count. The reason was that a highly available hub conducts more block transfers than a hub with a lower availability, which may seriously bias results towards the H.A.H configuration. Results are shown in Fig. 14.

For the same degree of clustering, this figure shows that overall the users resort to the online cloud storage service substantially more times when the availability of the hub is low, effect that is more significant for file downloads. This behavior is aggravated for social graphs for which the clustering is small. For instance, for the topology with $CC = 0.3$, the number of data transfers out of the cloud increases a 26.5% when the availability of the social hub decreases from 0.594 to 0.278. The reason is that for social graphs with small clustering, users have fewer chances of downloading data blocks from their friends, thus making the system more dependent on the availability of the hub.

Table 2
Costs estimation of FriendBox compared with Amazon S3 for the experiment workload.

		Storage ($/month)	Down. Traffic ($)	Storage Buffering ($/month, only 1st month)	Down. Buffering Traffic ($)	FriendBox vs. Cloud (1st month)	FriendBox vs. Cloud (permanent)
H.A.H.	$CC = 0.3$	9.234	3.891	8.571	10.826	−21.19%	−68.19%
	$CC = 0.7$	9.234	2.941	8.227	10.392	−25.37%	−68.83%
L.A.H.	$CC = 0.3$	9.234	5.294	11.417	14.421	−2.18%	−64.79%
	$CC = 0.7$	9.234	3.459	9.373	11.839	−17.84%	−69.24%
Amazon S3		18.465	22.8	-.	-	-	-

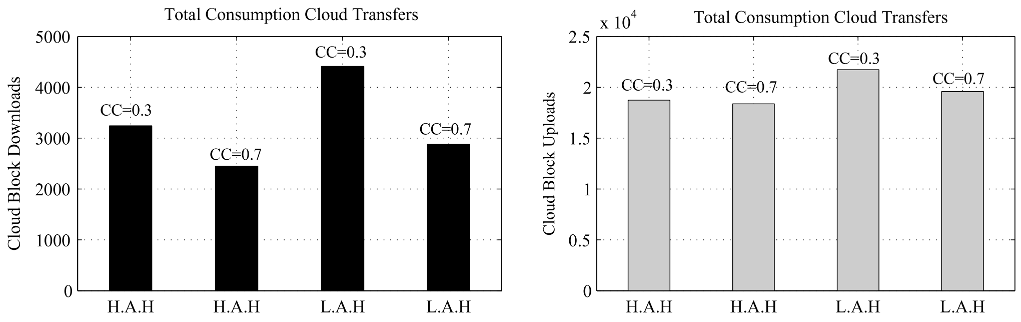


Fig. 14. Cloud block transfers depending on the hub's availability and the clustering coefficient.

For the same hub availability, a higher CC reduces significantly the number of data block transfers out of the cloud. To give some numbers: if the hub has low availability, the number of transfers is comparatively a 34.6% smaller in the high clustering graph than for the social topology with small clustering. This can be explained by the fact that a higher clustering degree is accompanied by a greater number of links between users, which in general increases the number of data blocks retrievable from friends at any time [8,5]. This reduces the number of accesses to the cloud.

Further, we observe that uploads consume a higher amount of cloud resources than downloads. This is because FriendBox minimizes the number of cloud transfers by giving priority to friends in the download schedule, only accessing the cloud in those situations where available friends cannot supply the necessary blocks to complete the file retrieval. However, uploads always require transferring a fraction of the data to the cloud, which increases its overall usage. It should be noted that alternatively uploading and downloading distinct files makes it difficult for offline nodes to download buffered blocks and serve download requests when they become online again. This means that less aggressive workloads would greatly reduce the number of cloud downloaded blocks, since there would be enough blocks available at friends.

Therefore, we see that a higher CC alleviates the consumption of cloud resources when the social hub is poorly available. This implies that when the hub is disconnected, Amazon S3 is used to temporarily buffer a smaller number of blocks per file storage operation than when the degree of clustering is low.

The previous observations are reflected in the economic cost of the FriendBox service as visible in Table 2.[10] At first glance, we observe that as the lower network CC and hub availability, the higher economic expenses in cloud resources. This particularly impacts on the number of extra blocks buffered in the cloud due to the unavailability of friends at the moment of storing a file. However, we should note that FriendBox greatly reduces the long term cloud costs. For example, configuring FriendBox with $F_C = 0.5$ and $n/k = 2$, users save up 50% of permanent storage costs and 87%–77% of download traffic costs compared with Amazon S3. Thus, we conclude that FriendBox is feasible in economic terms.

These economic savings opens up the possibility to develop storage services at low cost. Very often, SMEs and Startups cannot afford the steep infrastructure costs to support their business models, making the cloud an attractive hub for them. While cloud computing may save money in IT costs, the costs of renting a storage facility can be still very expensive for small businesses and Startups. Internet is full of successful IT stories of services, like `Spotify` or `Wuala`, which combined data centers with end-user resources to significantly lower the investment in servers and bandwidth, which is a pretty big deal for Startups. For instance, this combination allowed `Spotify` to scale up quickly without having to invest heavily in cloud resources, saving the company millions of dollars every year. In this sense, it is not hard to imagine businesses that take advantage of the "social cloud" model to reduce costs and flourish. Online social networks like `Facebook` launched development platforms that blossomed into entire ecosystems. And hence, it is not unwise to say that a similar approach could be followed by a social cloud to craft a blossoming ecosystem of value-added service providers. This issue is, however, outside the scope of this paper.

We summarize this section as follows:

Observation 12: *The availability of social hubs plays an important role in the consumption of cloud resources, specially for low clustering topologies.*

Observation 13: *In general, a high clustering degree reduces the overall amount of consumed cloud resources.*

Observation 14: FriendBox *provides an attractive trade-off between storage service and economic cost.*

6. Discussion on future directions

An important conclusion drawn from our evaluation is that *the degree of clustering plays a critical role on how storage resources are exchanged among social links*. More concretely, we have seen that resource fairness, simply understood as a cost–benefit ratio, can exhibit a large imbalance when the cluster coefficient is low.

We envisage two different strategies to address this situation:

- Apply a different placement policy to balance the contributed resources by each user; and
- Increase the cluster coefficient through incentives.

Regarding the first solution, the idea is to replace the round robin allocation policy used in FriendBox by a fairer policy. For

[10] According to Amazon's S3 at December 2013 we assume 0.12 per GB of outgoing traffic and 0.095 per GB/month of storage. Incoming traffic is free of charge.

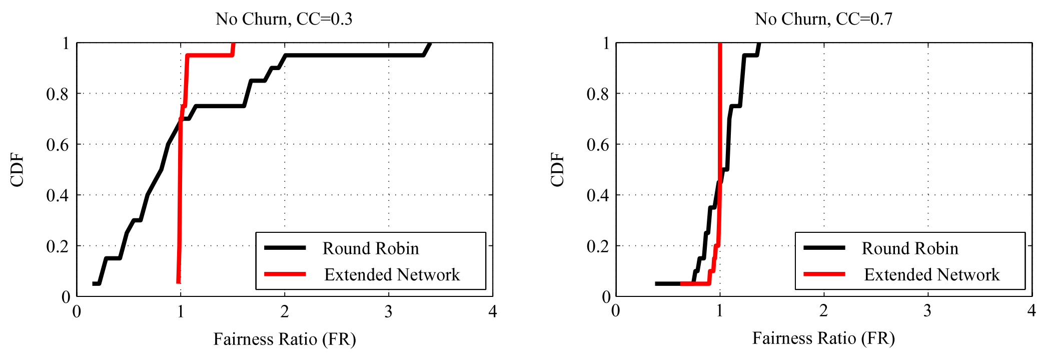


Fig. 15. Fairness ratio for different *CC* using round robin placement or widening storage links to the extended network.

instance, a better policy would be to allocate much more data to the members with a small number of social ties, because, in general, those users are prone to consume more resources than they contribute. This would free the hubs from donating too much resources to the social cloud.

However, this policy might introduce undesirable effects. For example, a large number of blocks might be allocated to a single friend in an attempt to reduce contribution asymmetry. Once this friend went offline, data availability could be highly affected because the data owner might unable to retrieve a sufficient number of redundant blocks from the remaining set of logged-in friends and the cloud.

Regarding the second solution, we have seen that the best fairness ratio is achieved in the social graphs with high clustering, mainly because the data is better spread among friends without overloading hubs. This leads to the question of which type of incentive mechanism would be appropriate to increase the clustering coefficient and improve the overall fairness.

An appealing way of regulating sharing, providing incentives to users and mitigating the risk of an unfair distribution of resources in a social context is the use of market metaphors as shown in [2]. Although using market-based mechanisms is not a new idea to solve the resource allocation problem in computer systems (see, for example, [30,31]), leveraging digitized social relationships provides benefits in terms of increased trust and lowers the barrier to share spare resources. The key idea would be to provide incentives for users to create new social interactions to increase the cluster coefficient up to the necessary level upon which a fair distribution of work among the whole social network could be achieved.

More technically, given a user v of the social cloud, let us consider the ratio of fairness at v, namely FR_v and calculated by (3), as the objective metric we want to equalize among all participants. Let us now denote by $d(v_1, v_2)$ the shortest distance between node v_1 and v_2 in the social graph, and by

$$X_v = \left\{v_i : d(v, v_i) \geq 1 \wedge FR_{v_i} < 1\right\}$$

the extended network of v that includes the friends and friends of friends that have a fairness metric less than 1 (contribute less than consume). If we consider the excess of contribution $\mathcal{M} = R_p - R_c$ as a currency in the social market, participants with a $FR_{v_i} > 1$ and $\mathcal{M} > 0$ could be allowed to use its extended network X_{v_i} to discover new social contacts where store new content, increase their FR_{v_j} while decreasing its own FR_{v_i}.

To give a sense of the efficacy of this solution, an initial simulation was run on the topologies of Fig. 4 using round robin scheduling for exactly 10 rounds of simulation. In each simulation round, storage requests were repeatedly made by all the members of the social cloud using the same setup as in the experiments of the preceding section. Results are depicted in Fig. 15, where it is easy to appreciate the high imbalance when the social graph is sparse ($CC = 0.3$) compared when it is highly connected ($CC = 0.7$).

If we turn our attention to the new mechanism, the members of the social network with an initial excess of contribution ($FR > 1$) after the first round of storage requests are allowed to use the extended network on subsequent rounds as explained above until they run out of storage currency. Contrary to the simple round robin policy, Fig. 15 clearly verifies how the fairness index at each member is close to the target value of 1: $R_p = R_c \Rightarrow FR = R_p/R_c = 1$, thanks to the use of the extended network and very importantly, irrespective of the clustering degree. The extended network serves to artificially increase the degree of clustering by creating new social links (transient social ties), thereby leading to a better balanced system. As a side effect, we are also improving data availability by adding more social contacts to the ego-centric graph of each user. The new social acquaintances might even belong to other time zones which would alleviate the effects of availability correlations.

Although the use of market metaphors in the social cloud is a promising line of work, their final adoption is yet uncertain as it remains to be studied how factors like the topology, availability correlations, etc., shape the form of utility functions. Regarding the implications on trust and privacy of adding new social ties, this decision potentially could be made using reputation measures to leverage the level of trust among direct links and the extended network, addressing to some degree the trust and privacy concerns of users.

7. Conclusions

Recently, the notion of "social cloud computing" has gained momentum for its amalgamation of social and cloud computing. Following this new trend, we have shown how to leverage social relationships to form a dynamic social cloud for storage. Although this model builds upon the unique environment in which users are motivated by social incentives, we have seen that there exist some difficulties and subtleties that prevent the realization of this concept in the real world, such as the availability correlation between social contacts and the asymmetry in contribution levels. Through a real deployment of a social cloud application in our campus, we have studied to what extent these factors affect the feasibility of socially oriented storage. Our analysis has revealed new insights on how to design a social storage cloud, in particular, when the storage resources contributed by each member are augmented with an external storage service like Amazon S3.

Acknowledgments

This research work was partly sponsored by the Spanish government through projects DELFIN (TIN-2010-20140-C03-03) and RealCloud (IPT-2011-1232-430000), and by the European Commission FP7 through project CloudSpaces (FP7-317555). We thank to the anonymous reviewers for their valuable comments and corrections.

References

[1] K. Chard, S. Caton, O. Rana, K. Bubendorfer, Social cloud: cloud computing in social networks, in: IEEE CLOUD'10, 2010, pp. 99–106.
[2] K. Chard, K. Bubendorfer, S. Caton, O. Rana, Social cloud computing: a vision for socially motivated resource sharing, IEEE Trans. Serv. Comp. (4) (2012) 551–563.
[3] S. Pearson, Taking account of privacy when designing cloud computing services, in: Software Engineering Challenges of Cloud Computing, 2009, pp. 44–52.
[4] C. Wilson, B. Boe, A. Sala, K.P. Puttaswamy, B.Y. Zhao, User interactions in social networks and their implications, in: EuroSys'09, 2009, pp. 205–218.
[5] R. Sharma, A. Datta, M. Dell'Amico, P. Michiardi, An empirical study of availability in friend-to-friend storage systems, in: IEEE P2P'11, 2011, pp. 348–351.
[6] S.A. Golder, D.M. Wilkinson, B.A. Huberman, Rhythms of social interaction: messaging within a massive online network, in: Communities and Technologies, 2007, pp. 41–66.
[7] A. Nazir, S. Raza, C.-N. Chuah, Unveiling facebook: a measurement study of social network based applications, in: IMC'08, 2008, pp. 43–56.
[8] R. Gracia-Tinedo, M. Sánchez-Artigas, P. García-López, Analysis of data availability in f2f storage systems: when correlations matter, in: IEEE P2P'12, 2012, pp. 225–236.
[9] R.J. McEliece, M. Kac, The theory of information and coding : a mathematical framework for communication, in: Encyclopedia of Mathematics and Its Applications, Addison-Wesley Pub. Co., Reading, Massachusetts, 1977.
[10] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.
[11] R.J. McEliece, D.V. Sarwate, On sharing secrets and reed-solomon codes, Commun. ACM 24 (9) (1981) 583–584.
[12] R. Gracia-Tinedo, M. Sánchez-Artigas, A. Moreno-Martínez, P. García-López, Friendbox: a hybrid f2f personal storage application, in: IEEE CLOUD'12, 2012, pp. 131–138.
[13] R. Curry, C. Kiddle, N. Markatchev, R. Simmonds, T. Tan, M. Arlitt, B. Walker, Facebook meets the virtualized enterprise, in: IEEE EDOC'08, 2008, pp. 286–292.
[14] Z. Guo, R. Singh, M. Pierce, Building the polargrid portal using web 2.0 and opensocial, in: GCE'09, 2009, pp. 5:1–5:8.
[15] K. John, K. Bubendorfer, K. Chard, A social cloud for public eresearch, in: IEEE 7th Intl. Conf. on E-Science (e-Science'11), 2011, pp. 363–370.
[16] A. Thaufeeg, K. Bubendorfer, K. Chard, Collaborative eresearch in a social cloud, in: IEEE 7th Intl. Conf. on E-Science (e-Science'11), 2011, pp. 224–231.
[17] L.P. Cox, B.D. Noble, Samsara: honor among thieves in peer-to-peer storage, SIGOPS Oper. Syst. Rev. 37 (5) (2003) 120–132.
[18] P. Druschel, A. Rowstron, Past: a large-scale, persistent peer-to-peer storage utility, in: HotOS'01, 2001, pp. 75–80.
[19] L. Toka, M. Dell'Amico, P. Michiardi, Online data backup: a peer-assisted approach, in: IEEE P2P'10, 2010, pp. 1–10.
[20] T. Mager, E. Biersack, P. Michiardi, A measurement study of the wuala on-line storage service, in: IEEE P2P'12, 2012, pp. 237–248.
[21] Z. Yang, B. Y. Zhao, Y. Xing, S. Ding, F. Xiao, Y. Dai, Amazingstore: available, low-cost online storage service using cloudlets, in: IPTPS'10, 2010.
[22] R. Gracia-Tinedo, M. Sánchez-Artigas, P. García-López, F2box: cloudifying f2f storage systems with high availability correlation, in: IEEE CLOUD'12, 2012, pp. 123–130.
[23] N. Agrawal, W.J. Bolosky, J.R. Douceur, J.R. Lorch, A five-year study of file-system metadata, ACM Trans. Storage, 3, (3).
[24] W.K. Lin, D.M. Chiu, Y.B. Lee, Erasure code replication revisited, in: IEEE P2P'04, 2004, pp. 90–97.
[25] B. Leiba, Oauth web authorization protocol, IEEE Internet Computing 16 (1) (2012) 74–77.
[26] J.K. Resch, J.S. Plank, AONT-RS: blending security and performance in dispersed storage systems, in: Usenix FAST'11, 2011, pp. 191–202.
[27] L. Pàmies-Juárez, On the design and optimization of heterogeneous distributed storage systems, (Ph.D. thesis), Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, 2011.
[28] R. Zafarani, H. Liu, Social computing data repository at ASU (2009), URL http://socialcomputing.asu.edu.
[29] S. Guha, N. Daswani, R. Jain, An experimental study of the skype peer-to-peer voip system, in: Peer-to-Peer Systems (IPTPS), 2006.
[30] I. Sutherland, A futures market in computer time, Commun. ACM 11 (6) (1968) 449–451.
[31] M. Feldman, K. Lai, L. Zhang, The proportional-share allocation market for computational resources, IEEE Trans. Parallel Distrib. Syst. 20 (8) (2009) 1075–1088.

Raúl Gracia-Tinedo is a Ph.D. candidate at the University Rovira i Virgili (URV), Tarragona, Spain, in the Architectures and Telematic Services (AST) research group. His research interests are related to distributed storage systems, peer-to-peer networks and cloud computing. Contact him at raul.gracia@urv.cat.

Marc Sánchez-Artigas obtained his Ph.D. degree in 2009 with European Mention at UPF (Barcelona, Spain). During 2008, he worked at École Polytechnique Fédérale de Lausanne (Switzerland). He received the Best Paper Award from IEEE LCN'07. He also served as a Guest Editor for Computer Networks Journal. He organized the 12th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'12). He is actively participating in the coordination of the EC FP7 project CloudSpaces on personal cloud storage. Contact him at marc.sanchez@urv.cat.

Aleix Ramírez is a computer engineer at the University Rovira i Virgili (URV), Tarragona, Spain, in the Architectures and Telematic Services (AST) research group. He is currently participating in the RealCloud project. His research interests are related to security and privacy in cloud computing. Contact him at aleix.ramirez@urv.cat.

Adrián Moreno-Martínez is a computer engineer at the University Rovira i Virgili (URV), Tarragona, Spain, in the Architectures and Telematic Services (AST) research group. He obtained his M.Sc. degree in Computer Science from the URV, in 2012. He is currently participating in the FP7-CloudSpaces project. His research interests are related to distributed storage and cloud computing. Contact him at adrian.moreno@urv.cat.

Xavier León is a Post-doctoral fellow in the Department of Computer Engineering and Maths at the Universitat Rovira i Virgili (Spain). He obtained his Ph.D. in 2013 from the Universitat Politécnica de Catalunya (UPC). His research interests include computer networks, complex systems, and self-organization of distributed and decentralized peer-to-peer systems through economics-based mechanisms. Contact him at xavier.leon@urv.cat.

Pedro García-López is a professor at the Computer Engineering and Mathematics Department at the Universitat Rovira i Virgili (Spain) and IEEE Member. His research topics are distributed systems, peer-to-peer, cloud storage, software architectures and middleware and collaborative environments. He currently leads in Tarragona the Architectures and Telematic Services (AST) research group and coordinates the FP7 European project CloudSpaces. He is an active member of the P2P research community: Steering Committee Member of IEEE P2P, General Chair of IEEE P2P'12, PC Member of IEEE P2P (2008–2011), chair of the IEEE Collaborative Peer-to-Peer Systems Workshop (COPS 2009, 2009, 2010). Contact him at pedro.garcia@urv.cat.