POSN: A Personal Online Social Network

Esra Erdin[1], Eric Klukovich[1], Gurhan Gunduz[2], and Mehmet Hadi Gunes[1(✉)]

[1] University of Nevada, Reno, USA
mgunes@unr.edu
[2] Pamukkale University, Denizli, Turkey

Abstract. A growing concern for end users of Online Social Networks (OSNs) is the privacy and control of user data due to the client-server architecture of the current ecosystems. In this paper, we introduce a privacy preserving decentralized OSN platform, which mimics real life social interactions. In particular, we decentralize the OSN platform and give direct control of the information to the user. The distributed platform removes central authorities from the OSN and users share their content only with intended peers through mobile devices. This decentralized system ensures that interaction happens between friends and third parties cannot access the user content or relationships. To be able to efficiently share objects and provide timely access in the POSN platform, we take advantage of free storage clouds to distribute encrypted user content. The combination of phone-to-phone applications with cloud infrastructure would address the availability limitation of peer-to-peer systems, while enjoying the benefits of peer-to-peer systems, such as no central authority and scalability.

Keywords: Decentralize · Phone-to-phone · Privacy · Social networks

1 Introduction

Online social interactions are an integral part of our daily activity. As indicated in Milgram's experiment, we live in a small-world [10,21]. Connecting this small-world in the digital world has been a challenge and has been addressed in various ways. OSNs enable frequent social interaction and expansion of knowledge or gossip. The emergence of OSNs sparked a major reform in information spread and how users interact with each other. From data to search to social interactions, users around the world are now more deeply connected to the Internet as user-generated content undergoes perpetual growth and expansion.

The client-server architecture of the current OSN ecosystems have raised privacy concerns. In general, users have to trust corporations (and governments) with their personal data when using OSNs. OSNs collect considerable amount of personal information about their clients and provide new services based on collected or derived information. For instance, a provider can filter advertisements based on user profile or user's circle (i.e., friends). Additionally, OSNs have predictive

© IFIP International Federation for Information Processing 2015
H. Federrath and D. Gollmann (Eds.): SEC 2015, IFIP AICT 455, pp. 51–66, 2015.
DOI: 10.1007/978-3-319-18467-8_4

capabilities about the users as they continuously gather user data. Researchers have tried to address privacy concerns by user-based key management [6,12,14, 34], peer-to-peer architectures [8,9,11,13,25,27,32,36], or decentralized platforms [1,3,20,30,31,33,35,38]. In general, peer-to-peer OSN architectures suffer from inefficiency due to high churn rate of users. Even though some of the proposed decentralized platforms utilize cloud resources, they rely on compute clouds which are typically fee-based and could analyze user data.

In a decentralized architecture, efficient sharing and timely access of objects play a vital role [22]. In two-way friendship OSNs, users typically access a small number of objects among vast number of posts, with many users accessing only recently posted objects. Additionally, updates by a user must be available to his/her friends in a timely manner regardless of whether the posting user has become offline. To address these challenges, we take advantage of *free storage clouds* to distribute *encrypted user content*. Even though there are OSNs that utilize cloud, they either use it as an execution platform with plaintext or as a backup store to central servers. To best of our knowledge, this is the first study to promote a decentralized OSN relying on mobile devices and storage clouds with no central server. While peer-to-peer systems require a peer to be online for data exchange, we utilize cloud to greatly enhance data availability. The free cloud storage services allows POSN to function through user clients such as smart phone or tablet apps with no infrastructure of its own. Moreover, smart phones and tablets have considerable computing capabilities that is utilized to provide direct interaction between users.

In this paper, to address privacy concerns, we introduce a Privacy Preserving Decentralized Online Social Network (POSN) which mimics real life social interactions. Figure 1 presents sample snapshots of Android app in development. The main contribution of this paper is *the combination of phone-to-phone applications with cloud infrastructure* to address the main limitation of peer-to-peer systems, i.e., *availability*, due to high churn rate while enjoying the benefits of peer-to-peer systems, i.e., *no central authority and scalability*. This approach can be expanded to provide other forms of social communications where data privacy is a concern.

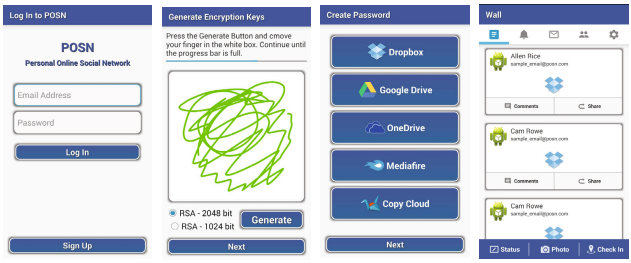

Fig. 1. Snapshots of the POSN prototype

2 Cloud as Storage

In proposed POSN, each user's data is kept in a separate location and the owner is in charge of granting access to his or her friends. The cloud can provide this

storage resource free of charge. Users upload encrypted data to their cloud and fetch data from the cloud's of their friends (see Section 5). Cloud providers could only observe encrypted data and share this encrypted data with users that have access to it as shown in Figure 2.

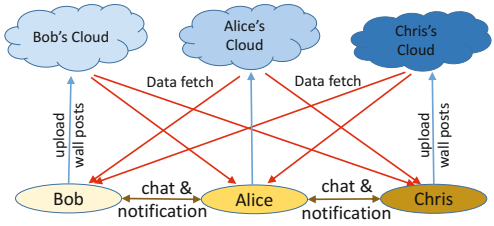

Fig. 2. Personal Online Social Network

In order to find online friends, we need to check their cloud location file where their IP address and Port number is stored. In order to access posted content of a friend, a user should download the content i.e. the wall post file. In general, walls containing posts will be automatically downloaded but the multimedia content and comments will be downloaded only when a user views it. Even though downloading a file is straightforward, checking hundreds of user's files across different clouds is not efficient. Knowing if your friends have a new content will speed up the process. Online friends can speed up the process as they can share their knowledge about common friends (see Section 8).

In order to see whether clouds can provide communication efficiency of OSNs, we compared the file upload/download timing of popular OSNs (i.e., Facebook and Google+) with popular clouds (i.e., Dropbox, Google Drive, Sky Drive, Mediafire, Copy Cloud). Figure 3 presents the average upload and download timing of 10 measurements using an Android tablet over WiFi at our campus. In these experiments to reduce bias due to background processes and radio status, we closed all unnecessary services, performed a small file transfer to ensure radio is on, and kept the measurement process in the foreground. As Facebook resizes and compresses pictures, we were not able to download a picture of about 1MB even though larger pictures were uploaded. Hence, 1MB and larger files are videos doubling in size whereas below 1MB are pictures resized by Facebook. We obtained comparable performances in our experiments at 10 different WiFi locations throughout the city as well (results not shown due to space limits).

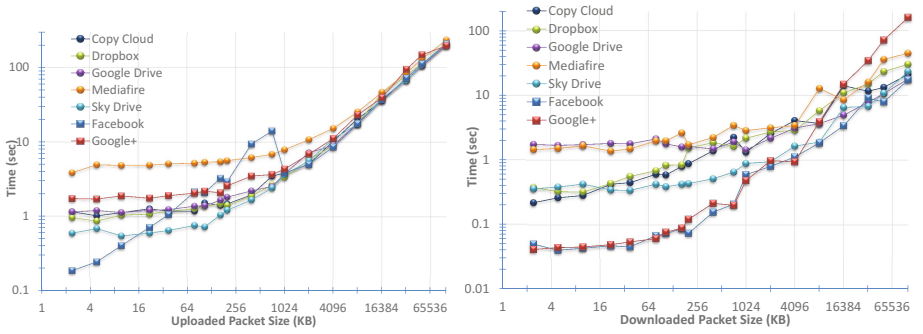

Fig. 3. Upload and Download times (log scale)

When uploading a photo, Facebook becomes progressively worse for pictures (likely due to re-rendering) but is comparable to the rest for videos. Google+ seems worse than all but one cloud provider. Our in depth analysis revealed that the Google+ API streams the videos, therefore the total download time is considerably increased. Overall, upload performances are similar as videos of 1MB and larger are uploaded. Download performance of OSNs are also better than clouds for files smaller than 1MB but are comparable for larger sizes. Even though this might impact interactive communications such as chat, performance difference of 1 sec or less would not be considerable for humans to affect adaptation of POSN. These experiments shows there is no significant performance difference when uploading/downloading a file to/from an OSN or a cloud provider.

Additionally, we performed measurements with 7 mobile phones over 3G in the Reno/SFO area. Figure 4 presents the ratio of Dropbox over Facebook performance for transfer of the same file by the phone around the same time. As each volunteer did not complete the same number of experiments, we present each point separately. Note that points above 1 indicate OSN is faster than cloud. Overall, we observe the OSN can be up to 10x faster or the cloud can be up to 9x faster, while in most cases OSN is faster. We believe such considerable performance variation in data transfers is due to the 3G traffic rate instability.

3 Friendship Establishment

Establishing friendship between interested people in a decentralized system is a challenge. Hence, in POSN, we need to rely on other mechanisms in establishing friendship and ensuring identities. POSN platform relies on users' existing contacts, i.e., phone numbers and email addresses, in order to establish friendship and ensure identities. POSN platform can exchange emails or SMS messages between users to establish certificates/tokens that will inform the cloud location and public keys of friends as in Figure 5. As the temporal identities need to be unique, POSN generates identities based on hash of email addresses or phone numbers.

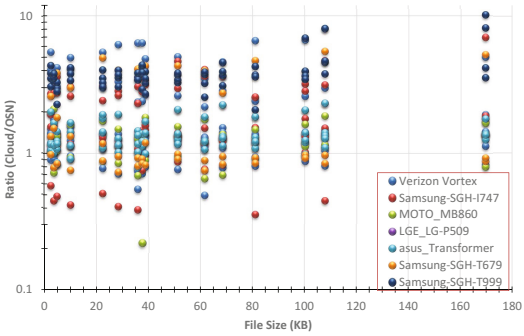

Fig. 4. 3G Performance Comparison (log scale)

When a person wants to establish friendship with a contact, s/he sends a URI containing her/his identity, public key, and profile location. The receiving person may then respond with her/his information if s/he wants to establish a link. Note that, both messages should be exchanged via e-mail or SMS so that real identities are ensured. This approach is also in line with our focus on personalized OSN as the links in our network should correspond to real life friendships between users.

Note that, a malicious email provider might tamper with message and this is a research challenge we will focus on. To address this issue, public key certificates can be obtained or secure e-mails can be sent with PGP to assure validity of public keys against potential man-in-the-middle attackers. Distributed key agreement protocols can also be utilized to verify public keys among friends [5,28,29].

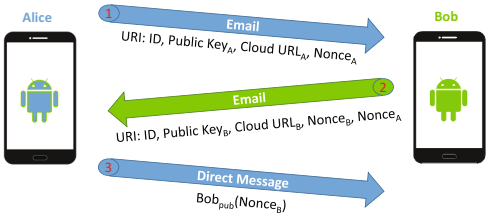

Fig. 5. Friendship establishment

4 Data Dissemination

POSN utilizes cloud storage system where users upload their content to a cloud and obtain wall posts of friends from their clouds. Even though one might think the overhead would be prohibitive (e.g., Facebook has 1.2 billion users), each user needs to manage content of their friends. On average, a Facebook user has 130 friends (while the number is 214 in the United States) [18] who are typically clustered into 12 groups [37]. We keep users in groups similar to most OSNs and have a wall post file for each group in addition to individual wall file. When the user makes a post it is appended to the appropriate group.

To better understand how data is being generated in OSNs, we monitored the activities of 16 Facebook users and their circles (i.e., posts shared by their friends) for 15 days with explicit permission from the users. On average; for analyzed users, number of friends is 220 (146 of whom used a smart device) who are divided into 9 groups. Similarly, per day, average number of text posts in the user's circle is 53, links is 37, pictures is 51, videos is 7, and chat messages is 85. The monitored users themselves uploaded just a total of 25 pictures and 2 videos in total over the same period.

Figure 6 shows how multimedia (i.e., pictures and videos) posts are generated by circles of each user. We observe majority of photos are less than 100KB while

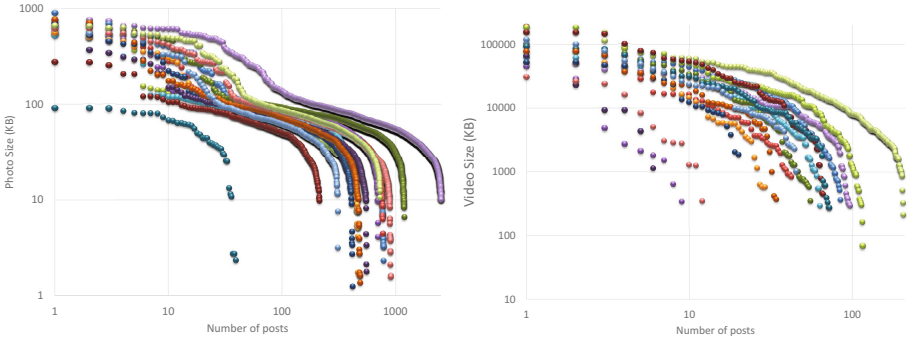

Fig. 6. Multimedia posting pattern of the circles of users (log-scale)

none is greater than 1MB. Likewise, majority of videos are less than 10MB but exceptionally there are videos larger than 100MB.

As multimedia can be very large, the multimedia should be distributed through the cloud where the user lets online friends know that there is a new post with a direct link to download data from the cloud. Otherwise, the user would send multiple copies of the file (i.e., as many as the number of online friends). Figure 7 shows the sender bandwidth overhead for a user if s/he was to post her/his friend's posts. It indicates that user overhead considerably increases if the user directly sends multimedia files to online friends. For sending average file of 93KB to 90 online friends, the user would need to send over 8MB data. Hence, rather than sending multiple copies of the file, the poster notifies online friends about the post that has been uploaded to the cloud.

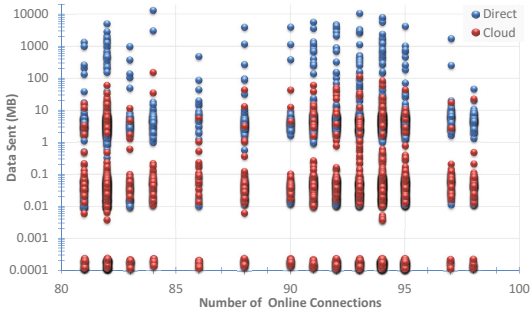

Fig. 7. Sender overhead for multimedia (log-scale)

Moreover, Figure 8 shows the frequency of non-multimedia posts (e.g., wall update, link, like, and comment) for the circle of a user. In the figure, points show the average number of posts when x number of friends were online and the line is the moving average. On average, during a session, a user posts 66 non-multimedia content while there are on average 81 online friends. Even though, cloud is utilized for efficient distribution of content, non-multimedia posts can directly be delivered to online friends.

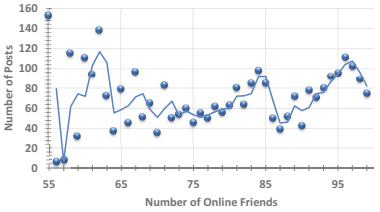

Fig. 8. Non-multimedia posts in a user's circle (log-scale)

5 Privacy Protection

In order to provide security and privacy in the POSN platform, data is encrypted before uploading into the cloud. Every user in the POSN would have a public key. This public key is exchanged at the time of friendship establishment and shared through the cloud. However, this would be very inefficient as a separate wall post would be made for each friend. Instead, POSN keeps a wall for each group as most users forms clusters of friends [17,24]. Therefore, each group needs a symmetric key for the group wall, and these keys need to be exchanged securely. In order to handle the access to the wall post that belongs to a specific group, the poster embeds a symmetric key into the file as shown in Figure 9. Using this mechanism, POSN ensures security without adding considerable overhead

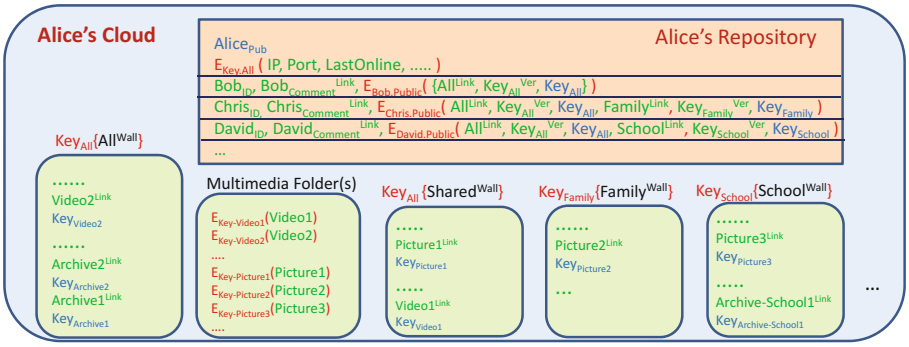

Fig. 9. Key Hierarchy to Protect User Content

to the system since the symmetric keys do not need to be exchanged but rather recovered through public keys. Furthermore, as multimedia files might be posted to different groups, we encrypt multimedia files with individual symmetric keys, which are shared in the wall post for the group.

Each user's repository provides group keys encrypted with each user's public key along with the comment file for that user (see Section 6 for commenting). In Figure 9, Bob just has access to the "All" group while Chris has access to "All" and "Family" groups. As the symmetric keys for groups can be updated, the Key^{ver} indicates the version number of the keys. Note that, each user has a different repository file and hence can not see other's by default.

In order to assess encryption overhead, we analyzed performance overhead of encryption (using standard Android AES library) using an Android tablet over WiFi at our campus. Figure 10 presents performance ratio of Dropbox over Facebook where blue points indicate transfer without encryption and red points indicate transfer with encryption for average of 10 measurements. Overall, we observe that encryption increases the ratio on an average from 1.40 to 1.71, an average of 22% increase in time with encryption. This results are encouraging as the extra encryption is not adding significant delay overhead.

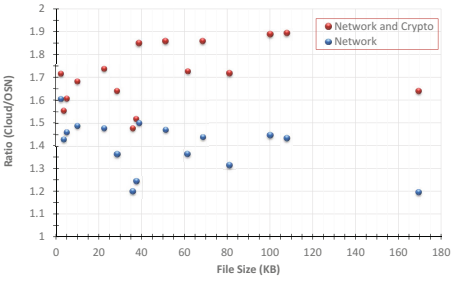

Fig. 10. Cryptography overhead

6 Commenting

Commenting is one of the important feature of OSNs since people typically care about their friends opinion and it should be incorporated into the framework. However, majority of the peer-to-peer and decentralized OSN platforms ignore

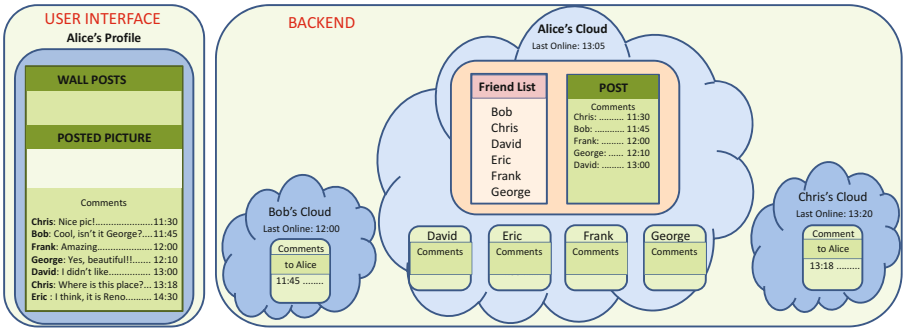

Fig. 11. Comment propagation

this feature as it is very challenging to provide in a decentralized structure (see Section 10). In POSN, each user posts her/his items into the cloud. In order for friends to post a comment, they should be granted with write permission to the wall. This might introduce security problems since friends might intentionally or accidentally mess up a shared file.

One way of incorporating commenting in this framework would be opening a file with a shared write permission for each friend (such as David, Eric, Frank, and George in Figure 11). If a friend wants to make a comment about a specific post, then this comment can be posted into a file that belongs to that specific friend. Associating the comments with the posts can be done by using the previously assigned IDs of wall posts. The next challenge is how to spread this comment to friends viewing the original post. So far, we assumed wall post file will be used to exchange data. Now, we would need to look into as many as the number of friends' files to see if they have a comment about posts. The owner can aggregate all the comments under friend's comment files to be included under the original post file and remove them from individual comment files. Unfortunately, this process can be accomplished only if the owner is online.

Moreover, users have different cloud providers and might not have an account with the poster's cloud (such as Bob and Chris in Figure 11). Such users will have comment files for their friends's post in their own cloud. To handle these distributed commenting files, the original poster needs to have a file that holds the friend's information where a link to each friends commenting location is provided (such as *Friend List* in Figure 11). In this case Alice will not be able to erase Bob's comment after populating it to her wall.

The last issue with the commenting is the efficiency. Gathering comments from each friends' cloud will introduce considerable overhead to the system. To minimize this overhead, we implement caching schemes as described in Section 8. When making a comment to a post, the user sends this comment to all (common) online friends who can further propagate to other common friends.

7 Search Optimization

Another feature of OSNs that is challenging in distributed platforms is the search for objects. In POSN, the cloud is used only for storage purposes and friends' encrypted content is scattered across several locations. In order to search for content of friends, the wall files from all friends should be downloaded to a client to be searched through. Such a scheme is very inefficient since the number of content belonging to a user's circle can be very high. In order to overcome this problem in POSN, an index structure is implemented as described in Figure 12. Whenever a post is made, its keywords or tags are inserted into the index file by the content creator and uploaded to the cloud along with the post.

Considering there are several groups for a user, one index file will not be enough to handle different groups. Because an owner might post a multimedia content to a specific group, inserting its tag information into a common index will hint other users of its existence. Hence, POSN keeps a different index file for each group encrypted with the symmetric key of the group that it belongs to as shown in Figure 9.

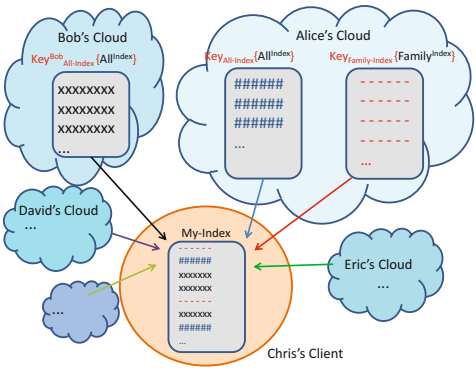

Fig. 12. Multi-level Indexing for Searching

Since an index file is needed by all of our friends, it needs to be disseminated to them. If a user wants to search for a multimedia content, s/he needs to download the index files of each friend. These index files are then searched for the desired content, which is not very efficient either. To improve search efficiency, POSN will preemptively process the index files of all friends and create a new index structure on the client whenever the user is online.

8 Data Distribution Optimization

The lack of a central server results in accessing several locations to gather the posts from friends. In the worst case, the user has to check all of friends' cloud locations, which may introduce significant overhead. In POSN framework clients establish direct connection with online friends' clients. When online, clients can exchange information about common friends that are not necessarily online. The number of connections that a client needs to aggregate the information from can

Fig. 13. Friends inter-connections

considerably be reduced as friends typically form clusters (i.e., communities). Figure 13 presents the network between 635 friends of a monitored user where they form several clusters among themselves. Online friends can exchange information about common friends' latest posts. As a result, the number of connections that a client needs to aggregate the information from will decrease.

In POSN, when a client (such as client **A** in Figure 14) becomes online, it will look for friends' cloud to see if they have a new post and learn their communication address (i.e., IP address and port number) if they are online. If the friend (such as client **D**) is online then the client can establish direct connection and ask the friend about common friends (clients **C**, **E**, **F**, and **G** in the example). The online friend then provides its knowledge about common friends. Once other online friends (such as **C**) are reported then the newcomer can recursively query these friends. This operation is carried out only when a client comes online. Thereafter, rather than periodically checking for new wall posts at every friend's cloud, post notifications are pushed to online friends. As users might become online simultaneously, they need to first update the online status in the cloud and then look up for friends to assure that they are aware of each other.

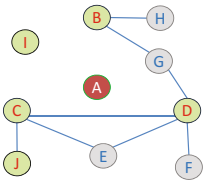

Fig. 14. Phone-to-Phone exchange

The order of lookup is important as finding online friends early on has considerable benefits and there can be several approaches. The first one is to rank friends by the *number of common friends*. The second method is to rank friends by their expected *online duration*. In order to implement these approaches, a user needs to keep and share the relevant information. The success rate of these methods is a research issue that we will try to optimize in our system.

In Figure 15, we compare four methods namely, ideal, most online duration, most common friends, and random for our measured data. In the figure, the x-axis indicates the percentage of online friends when a user becomes online and the y-axis indicates percentage of connections that could be eliminated. One issue is to determine when to take a snapshot of online activity. To mimic real user activity, we assumed the user came online one minute before one of her/his friends actually became online. Hence, we obtain a much

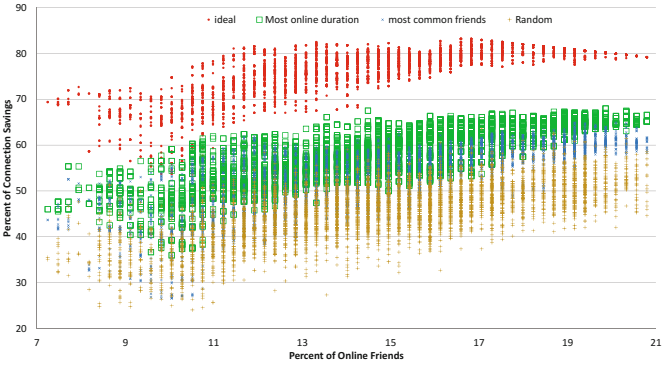

Fig. 15. Help from online friends

larger number of data points than an individuals online pattern. The marks indicate one instance in our measurements for the user whose circle is shown in Figure 13. On average, 14.6% of friends were online while ideally one could save 76.5% of connections on average. Heuristics to first lookup at friends that are most online saves 57.9% of connections while prioritizing most common friends saves 53.2% of connections on average. Even, random ordering saves 45.7% of connections on average. Our evaluations with the user with 435 friends yield very close results (difference of less than 1% in each average).

Figure 16 presents the amount of multimedia data a user would need to download when s/he became online based on the measurement of user in Figure 13. Similar to Figure 15, we assume the friends' online pattern as the user's online duration. As the time between logins increase, the amount of data the user might download from her/his friends increases. In extreme cases, we observe there is about 1GB of data when the user has not logged in for about a day and friends have posted large videos. Hence, rather than auto-downloading all multimedia content, POSN would parse most recent posts and wait for the user to fetch subsequent content as s/he is scrolling through the posts (especially for large videos).

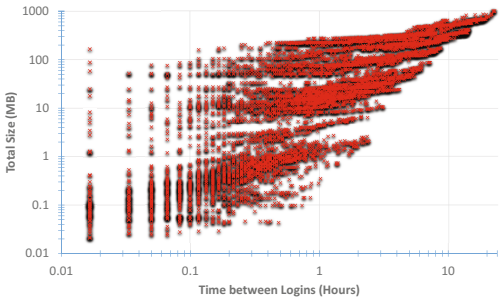

Fig. 16. Multimedia data to download when a user becomes online (log-scale in both axis)

9 Discussion

9.1 Social Challenges

Friendships might not last forever or the level of friendship might change, as a result the content that friends exchange will change. The POSN framework should reflect changes in the friendship status. To reduce the level of information exchange with a friend, the group or groups that friend belongs to could be changed. As we might not want the person to be aware of unfriending, we move all other friends to new group(s) with new key(s). Hence, the unfriended person would think that there are no new posts even if they analyzed the raw data.

Another challenge in real life is *stalking*. In order to stalk someone in our system, s/he has to know the victim in real life and be a friend in the POSN platform. If the stalker is a friend of victims friend, s/he can only see the comments of the victim. There is no way that the stalker can see the shared content of the victim through POSN if they are not friends.

To allow third party application support without harming user privacy, each user can keep an App File for each authorized application in their cloud. This App file is encrypted with a key as any other group wall. Applications will

write the relevant data into the files without revealing it to the app developer. Moreover, friends who want to partner in specific app can exchange their App file location and keys so that they can collaborate/compete. Online applications can also exchange information over the socket connection that is created by the online clients.

9.2 Security Challenges

Stolen devices pose a security risk. The owner of the stolen device should send notifications immediately. This notification could be send either by email or SMS or through some automated mechanisms to minimize the damage. Once a notification is received every key that is paired with that user would be renewed. On the other side, the mechanism should be carefully deployed so that this is not employed for denial of service attacks. Hence, we will investigate a solution that provides immediate notification mechanism while it can not be used for DoS.

Finally, a cloud provider might track who is accessing individual files and gain knowledge about interactions of a user [15,23]. Note that, to download content, a friend just uses direct link to the file without logging into the cloud. Also, online users do not need to connect to the cloud to exchange non-multimedia content. To prevent even IP based analysis by cloud provider, the user might utilize multiple clouds for different groups s/he is managing. Moreover, friends can utilize proxies and anonymizer technologies in her/his communications with the cloud [16,19]. Likewise, a user's online friends (who are not necessarily friend with the person whose encrypted data is downloaded) can be utilized in accessing clouds so that the cloud is not certain of who is accessing the content.

10 Related Work

We have analyzed some of the decentralized OSNs that we could identify, and found several decentralized OSNs currently in use: Diaspora [3], Friendica [1], and RetroShare [2]. Diaspora has tens of thousands of active users. Diaspora is a set of pods (currently about 130 are publicly available) that decentralize the system and is accessible through browsers. Even though an individual can deploy their private pod, the main issue with Diaspora is that each pod is a central server of many users. Similarly, Friendica builds a social web platform using set of decentralized servers. RetroShare is a peer-to-peer PC application that provides decentralized OSN support.

Additionally, there are several decentralized OSNs proposed in academia, but none of them seems to be actively used. Cachet [27], PeerSon [8], Safebook [9], and LifeSocial [13] build a peer-to-peer (p2p) system where peers are not necessarily trusted. Polaris [36], My3 [25] and Vegas [11] build a p2p network among friends to replicate data. Confidant [20], Tent [4], and Vis-a-Vis [31] build a distributed platform by building a protocol among servers that process user data. Persona [6] proposes to use attribute based encryption to store user data in

untrusted storage servers with web based OSN platform. PrPl [30] allows a user to use personal or 3rd party server to store their encrypted content.

Finally, there are privacy preserving social networks that are not traditional OSNs. Pythia attempts to present a privacy aware p2p network for social search [26]. Gossple is an anonymous social network that helps users build new relationships with others based on mutual interests [7]. Priv.io [38] builds a decentralized web based platform to securely share data. Contrail is a communication platform that provides content filtering in decentralized social networks [33]. NOYB [14] and Lockr [34] can be integrated into current social networks in order to guarantee the users privacy in OSNs.

Friendship Establishment: The decentralized networks that provide security as a feature tend to be more careful with revealing the very existence of certain users, thus finding friends becomes a challenge. Others are more cavalier with this information so making more friends is easier. There is a tradeoff of how privacy protecting the system is and how difficult it is for users to find and add their friends. As Diaspora and Friendica has member directories, friends can can easily be found. In Cachet, LifeSocial, and Vis-a-Vis users can locate friends through a DHT. PeerSon, Persona, Polaris, RetroShare, and Vegas exchange certificate files to establish friendship. In proposed POSN, we rely on existing identities, i.e., email addresses, to exchange credentials and establish friendship.

Data Storage: In a decentralized OSN, the data is, by definition, not stored in a single location. Some applications store the data in a distributed manner as a security precaution, some do it in order to promote inter-connectivity between users, and others do it for a combination of reasons. Diaspora and Friendica store data in a set of servers, which can be private. P2p systems store encrypted data amongst all peers (i.e., Cachet, LifeSocial, and PeerSon) or only in trusted peers (i.e., My3, Safebook, and Vegas). Confidant, Tent, and Vis-a-Vis decentralize the data among the users selected servers (potentially cloud providers), but the servers actually process the user data. Persona stores data in servers but only stores encrypted data. Some store data only in the user's PC (i.e., RetroShare) or smart phone (i.e., Polaris). In the proposed POSN, we build a p2p platform between mobile devices and utilize the cloud for encrypted data storage.

Support for Content Search: Among analyzed systems only Confidant, Diaspora, and Friendica users can search for content in their own server (or other servers in case of Diaspora), and Vegas users can perform controlled friend flooding to search for content. Different from p2p platforms that typically do not allow search function, POSN provides search of encrypted content by preemptively building search indexes.

Support for Commenting: Among analyzed systems only Confidant, Diaspora, and Friendica allow comments to posts as content is stored on a set of servers. Different from p2p platforms that do not allow commenting function, POSN provides commenting by propagating comments through the system.

11 Conclusions

OSNs have gained a great importance in our daily life. People prefer to communicate and interact with their friends through OSNs. In this paper, we propose a privacy preserving decentralized OSN so that users have direct control of their information. POSN can be easily deployed by installing an application and using a free-of-charge cloud storage provider. The POSN platform focuses on the community of individuals and tries to optimize the system through encrypted cloud storage. The combination of phone-to-phone applications with cloud infrastructure addresses the main limitation of peer-to-peer systems, i.e., availability, while enjoying the benefits of peer-to-peer systems, i.e., no central authority and scalability. POSN ensures that interactions happen between friends and third parties cannot access the user content or relationships.

Acknowledgments. We would like to thank James Bridegum for initial App development and Davut Ucar for collecting public WiFi measurements.

This material is based upon work supported by the National Science Foundation under grant number EPS- IIA-1301726.

References

1. Friendica: The internet is our social network. http://friendica.com
2. Retroshare: secure communications with friends. http://retroshare.sourceforge.net
3. Diaspora: community-run, distributed social-network. https://joindiaspora.com
4. Tent: All your data in one place. https://tent.io
5. Ateniese, G., Steiner, M., Tsudik, G.: New multiparty authentication services and key agreement protocols. IEEE Journal on Selected Areas in Communications **18**(4), 628–639 (2000)
6. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: an online social network with user-defined privacy. In: ACM SIGCOMM 2009
7. Bertier, M., Frey, D., Guerraoui, R., Kermarrec, A.-M., Leroy, V.: The gossple anonymous social network. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 191–211. Springer, Heidelberg (2010)
8. Buchegger, S., Schiöberg, D., Vu, L.-H., Datta, A.: Peerson: P2p social networking: early experiences and insights. In: SNS 2009 (2009)
9. Cutillo, L., Molva, R., Strufe, T.: Safebook: feasibility of transitive cooperation for privacy on a decentralized social network. In: WOWMOM 2009 (2009)
10. Dodds, P.S., Muhamad, R., Watts, D.J.: An experimental study of search in global social networks. Science (2003)
11. Durr, M., Maier, M., Dorfmeister, F.: Vegas - a secure and privacy-preserving peer-to-peer online social network. In: SocialCom/PASSAT (2012)
12. Feldman, A.J., Blankstein, A., Freedman, M.J., Felten, E.W.: Social networking with frientegrity: privacy and integrity with an untrusted provider. In: USENIX Security 2012
13. Graffi, K., Gross, C., Mukherjee, P., Kovacevic, A., Steinmetz, R.: Lifesocial.kom: a p2p-based platform for secure online social networks. In: P2P 2010

14. Guha, S., Tang, K., Francis, P.: Noyb: privacy in online social networks. In: WOSN 2008
15. Gunes, M.H., Evrenosoglu, C.: Blind processing: securing data against system administrators. In: IEEE/IFIP NOMS 2010
16. Karaoglu, H.T., Akgun, M.B., Gunes, M.H., Yuksel, M.: Multi path considerations for anonymized routing: challenges and opportunities. In: IFIP NTMS 2012
17. Kardes, H., Sevincer, A., Gunes, M.H., Yuksel, M.: Six degrees of separation among US researchers. In: IEEE/ACM ASONAM 2012
18. Kirschner, P.A., Karpinski, A.C.: Facebook and academic performance. Computers in Human Behavior (2010)
19. Li, B., Erdin, E., Gunes, M.H., Bebis, G., Shipley, T.: An Overview of Anonymity Technology Usage. Computer Communications **36**(12), 1269–1283 (2013)
20. Liu, D., Shakimov, A., Cáceres, R., Varshavsky, A., Cox, L.P.: Confidant: protecting osn data without locking it up. In: Kon, F., Kermarrec, A.-M. (eds.) Middleware 2011. LNCS, vol. 7049, pp. 61–80. Springer, Heidelberg (2011)
21. Milgram, S.: The small world problem. Psychology Today (1967)
22. Narayanan, H.A.J., Gunes, M.H.: Ensuring access control in cloud provisioned healthcare systems. In: IEEE CCNC 2011
23. Naruchitparames, J., Gunes, M.H.: Enhancing data privacy and integrity in the cloud. In: HPCS 2011
24. Naruchitparames, J., Gunes, M.H., Louis, S.J.: Friend recommendations in social networks using genetic algorithms and network topology. In: IEEE CEC 2011
25. Narendula, R., Papaioannou, T., Aberer, K.: My3: a highly-available p2p-based online social network. In: Peer-to-Peer Computing (2011)
26. Nilizadeh, S., Alam, N., Husted, N., Kapadia, A.: Pythia: a privacy aware, peer-to-peer network for social search. In: WPES 2011
27. Nilizadeh, S., Jahid, S., Mittal, P., Borisov, N., Kapadia, A.: Cachet: a decentralized architecture for privacy preserving social networking with caching. In: CoNEXT 2012
28. Perrig, A.: Efficient collaborative key management protocols for secure autonomous group communication. In: CrypTEC 1999
29. Rafaeli, S., Hutchison, D.: A survey of key management for secure group communication. ACM Comput. Surv. **35**(3), 309–329 (2003)
30. Seong, S.-W., Seo, J., Nasielski, M., Sengupta, D., Hangal, S., Keat, S., Chu, T.R., Dodson, B., Lam, M.S.: Prpl: a decentralized social networking infrastructure. In: MCS 2010
31. Shakimov, A., Lim, H., Caceres, R., Cox, L., Li, K., Liu, D., Varshavsky, A.: Vis-a-vis: privacy-preserving online social networking via virtual individual servers. In: COMSNETS 2011
32. Sharma, R., Datta, A.: Supernova: super-peers based architecture for decentralized online social networks. In: COMSNETS 2012
33. Stuedi, P., Mohomed, I., Balakrishnan, M., Mao, Z.M., Ramasubramanian, V., Terry, D., Wobber, T.: Contrail: enabling decentralized social networks on smartphones. In: Kon, F., Kermarrec, A.-M. (eds.) Middleware 2011. LNCS, vol. 7049, pp. 41–60. Springer, Heidelberg (2011)
34. Tootoonchian, A., Saroiu, S., Ganjali, Y., Wolman, A.: Lockr: better privacy for social networks. In: CoNEXT 2009

35. Yeung, C.-M.A., Liccardi, I., Lu, K., Seneviratne, O., Berners-lee, T.: Decentralization: the future of online social networking. In: W3C Workshop on the Future of Social Networking Position Papers (2009)
36. Wilson, C., Steinbauer, T., Wang, G., Sala, A., Zheng, H., Zhao, B.Y.: Privacy, availability and economics in the polaris mobile social network. In: HotMobile 2011
37. Wilson, R.E., Gosling, S.D., Graham, L.T.: A review of Facebook research in the social sciences. Perspectives on Psychological Science, May 2012
38. Zhang, L., Mislove, A.: Building confederated web-based services with priv.io. In: COSN 2013