PeerSoN: P2P Social Networking — Early Experiences and Insights

Sonja Buchegger, Doris Schiöberg
Deutsche Telekom Laboratories / TU Berlin
Berlin, Germany
firstname@net.t-labs.tu-berlin.de

Le-Hung Vu
EPFL
Lausanne, Switzerland
lehung.vu@epfl.ch

Anwitaman Datta
NTU Singapore
Singapore
anwitaman@ntu.edu.sg

ABSTRACT

To address privacy concerns over Online Social Networks (OSNs), we propose a distributed, peer-to-peer approach coupled with encryption. Extending the distributed approach by direct data exchange between user devices removes the strict connectivity requirements of web-based OSNs. In order to verify the feasibility of this approach, we designed a two-tiered architecture and protocols that recreate the core features of OSNs in a decentralized way. This paper focuses on the description of the prototype built for the P2P infrastructure for social networks, as a first step without the encryption part, and shares early experiences from the prototype and insights gained since first outlining the challenges and possibilities of decentralized alternatives to OSNs.

1. INTRODUCTION

Online social networks (OSNs) enable people to keep in touch with their friends, find people they lost contact with and even find new friends based on shared affinities such as groups, hobbies, interests or overlaps in friendship circles. More and more people use OSNs to exchange messages instead of e-mails and update information on what they are currently doing or other brief expressions of their identity, thereby leaving information about themselves and their friends on the system of the OSN provider. This information is usually private and intended for the eyes of a specific audience only. In OSNs like Facebook, MySpace, or Orkut, users can adjust privacy settings to protect their content and limit access by other users.

There is, however, no protection against access by the OSN providers themselves and little protection against third-party OSN application providers that gather information about their users' friends. Privacy concerns in OSNs have been raised repeatedly in the last few years.[1] For example, there has been an outcry over Facebook's Beacon program[2] that fed back users' information from third-party websites such as Amazon and displayed it on Facebook. There have also been questions of data ownership when users discovered that their profile information still existed on the OSN websites even after they had canceled their account. Even without such events that many people consider outright breaches of privacy, there are concerns over the centralized nature of user data repositories that can be used for data mining and targeted advertising by the service providers.

To provide users with privacy protection in OSN, we propose using a peer-to-peer (P2P) infrastructure and encryption for social networks. Previously, we outlined some of the challenges and preliminary solutions to make OSNs distributed and privacy-preserving [1]. We will summarize some high-level ideas from this prior work here when necessary for context. This paper extends this prior work and formally describes the prototype built for the P2P infrastructure for social networks, PeerSoN. We also share early experiences and insights gained from building the prototype.

In the remainder of the paper, we briefly describe our approach to privacy-preserving social networks, PeerSoN, in Section 2 and outline the typical features of OSNs and their implementation in PeerSoN. Section 3 describes the architecture and protocols for information exchange in the PeerSoN prototype. During the course of the prototype implementation, we have gained some experience and also encountered several more fundamental challenges that are not directly related to the prototype. Section 4 describes these insights and challenges. We relate PeerSoN to other approaches in Section 5 and conclude the paper in Section 6.

2. THE PEERSON SYSTEM

This section presents the PeerSoN system and describes its

[1] http://www.sophos.com/pressoffice/news/articles/2007/08/facebook.html
[2] http://arstechnica.com/tech-policy/news/2007/11/facebook-reevaluating-beacon-after-privacy-outcry-possible-ftc-complaint.ars

goal, the main design choices, and the set of OSN features it supports.

2.1 Goals

PeerSoN[3] aims at keeping the features of OSNs but overcoming two limitations: privacy issues and the requirement of Internet connectivity for all transactions. To address the privacy problem, we use encryption and access control coupled with a peer-to-peer approach to replace the centralized authority of classical OSNs. These two measures prevent privacy violation attempts by users, providers, or advertisers. Extending the decentralized approach, PeerSoN also enables direct exchange of data between devices to allow for opportunistic, delay-tolerant networking. This includes making use of ubiquitous storage to enable local services.

2.2 Design choices

The main properties of PeerSoN are encryption, decentralization, and direct data exchange. In a nutshell, encryption provides privacy for the users, and decentralization based on the use of a P2P infrastructure provides independence from OSN providers. Decentralization makes it easier to integrate direct data exchange between users' devices into the system. Direct exchange allows users to use the system without constant Internet connectivity, leveraging real-life social networking and locality.

2.2.1 Encryption

The main means for privacy protection in this context is to allow for users to encrypt their data and control access by appropriate key sharing and distribution.

Security considerations include bootstrapping, key distribution, and key revocation. A first, naive approach is to assume the availability of a public-key infrastructure (PKI) with the possibility of key revocation and to encrypt data with the public keys of the intended audience, i.e., the friends a user wants to grant access to files. For friends to exchange keys, first, extra-network contacts are used to establish and verify credentials, second, other friends are recommended and their credential authenticity vouched for by the former (trusted friends), and third, unrelated users are verified using the PKI.

Even in centralized OSNs such as Facebook, impersonation is possible by creating a profile using publicly available information and pictures. In PeerSoN, this should be prevented. Besides the steps outlined above, one can also use a challenge-response protocol to mitigate this, a general one to differentiate people from bots and a specific one to verify the possession of shared secrets/memories. The extra-network contact mentioned above can also be within PeerSoN by making use of physical meetings of friends and direct exchange between their devices running PeerSoN.

We are working on reducing the assumptions on PKI and developing a more efficient approach, using also symmetric keys, that will be integrated into PeerSoN. In the current prototype as described in this paper, however, the focus is on making OSNs distributed and not on the encryption part for privacy.

[3] `http://www.peerson.net`

2.2.2 Decentralized Architecture

To protect privacy, user data is encrypted and only accessible to those who have the right keys. The user determines who gets keys to what data. The main rationale behind decentralization is that no single entity can determine or change the terms of service, switch off the service, or have easy access to all data (albeit encrypted) to infer relationships or behavior. One could also argue that it is not necessary to have a decentralized approach, as one could store encrypted data centrally without loss of privacy. This is, however, not strictly true, since the central service provider might still be able to infer who is related to whom based on accesses and correlations, such as to IP addresses. In PeerSoN, crawling the peer-to-peer network can be prohibitive since individual peers can be offline and the network may be partitioned when used in a delay-tolerant network. Moveover, it becomes less commercially viable to provide a centralized system, because there is no commercial benefit from data mining and targeted advertising.

2.2.3 Direct Exchange

When the social networking service is decentralized and no longer web-server based, users need not be connected to the Internet for every single use of the system. They can exchange information directly with one another when they meet, thus making use of the real-life part of social networks. Users can carry data for each other and spread information through the physical social network or delay uploading data until someone has online connectivity. Moreover, parts of the system can be kept local, by limiting the scope of where information is stored and by making use of local devices, such as home access routers or ubiquitous computing devices that need not be connected to the Internet.

2.3 Features of Online Social Networks

Since the first goal of PeerSoN is to provide the advantages of OSNs, but without the drawback of loss of privacy, the desirable features of OSNs need to be defined. The various features provided by OSNs can broadly be classified into the categories of social link establishment, digital personal spaces, and means of communication. We now describe what we mean by these categories and report on how they are represented in PeerSoN.

Social Links. OSNs connect people by allowing them to list other users within the OSN that have a social relation, such as friends or colleagues. In addition to representing existing social ties, OSNs can also be used to re-establish lost social connections, if the people are also on the OSN. Social links can also be newly established, when users link to each other after meeting in virtual space, for instance in

common-interest groups. These groups form an overlay of social links, as affiliation networks do in real-life social networks [10]. The links to other users and the links representing group memberships make up social graphs and by letting information flow along these graphs, users have ambient knowledge of other users' presence and current activities. This way, information is pushed to rather than pulled by the user.

Digital Personal Space. In OSNs, users have their own digital personal space where others have access rights to read, write, or leave messages (text or other media). In this space, users express their identity typically by posting a profile, pictures, and a status of what they are doing or what they want to express at the moment. Users can post links or embed media. Users' actions can be reported to their friends in news feeds, contributing to the ambient social knowledge and complementing the information the users put in deliberately.

Means of Communication. Since OSNs, for users, are for maintaining social relationships, communicating with others is the central feature. There are several possible channels of communication among users. One can leave public messages using the digital personal space, again not limited to text, or send private internal e-mail-like messages. For synchronous communication, most OSNs provide instant messaging. An attractive feature of OSNs is that they are open to third-party applications. While many of these applications provide alternative means of communicating or getting in touch asynchronously, there are also applications that allow users to engage in joint activities, such as real-time games.

In the PeerSoN [8] prototype, we have implemented some of the features as a proof-of-concept. We therefore abstracted the implementation problems posed by the features above to one representative instance of a feature in lieu of implementing all features, even if they are based on the same principle. As an example, public messages in the form of so-called wall entries represent the functionality of asynchronous messaging. It is straightforward to add private messaging, as in an Inbox, to a full-fledged implementation.

The current PeerSoN implementation replicates the following features of OSNs. In the category of social links, users (peers) can become friends and thus establish a social link between each other. Digital personal spaces are provided in that users can maintain their own profile and a wall, a space for items posted by themselves or their friends. Communications between users are directly peer-to-peer when both are online, and the implementation supports asynchronous messaging when this is not the case.

3. IMPLEMENTATIONS

This section describes the high-level architecture and protocols of PeerSoN.

3.1 Prototype Architecture

PeerSoN has a two-tiered architecture. Logically, one tier serves as a look-up service. The second tier consists of peers and contains the user data, such as user profiles. The look-up service stores the meta-data required to find users and the data they store, for example, the IP address, information about files, and notifications for users. A peer that wants to connect to another peer asks the look-up service directly to get all necessary information. The peers then connect directly. After having exchanged a message or file, the peers disconnect immediately, except when doing instant messaging. Figure 1 shows an abstraction of the whole architecture by way of a message-sending example.

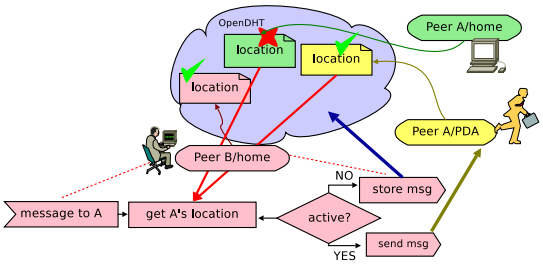

Figure 1: *B* sends a message to peer *A*.

PeerSoN currently uses a Distributed Hash Table (DHT) for the look-up service. The nodes of the DHT stay connected if the DHT protocol requires it, but the peers do not. The rationale for this divided architecture is to enable thin clients such as mobile phones or hand-helds to use the system, despite their resource constraints that prevent them from being reliable DHT nodes.

The focus here is not on building and deploying yet another DHT. To start with, we decided to use OpenDHT[4]. OpenDHT is a centrally managed deployment of the Bamboo DHT [7] on PlanetLab[5]. OpenDHT has been overloaded and unreliable recently, leading us to eventually resort to also using a single server process emulating the OpenDHT interface (put, get, and remove) for testing and evaluation. In the longer run, PeerSoN users should support such a look-up service in a self-contained manner. The question then becomes which machines would fit the needs of a DHT node to offer a reliable and scalable service. One idea could be to enable the usage of existing P2P networks, such as Kad or Gnutella as the look-up service. The architecture is conceptually not bound to a certain system. Any service that provides a key-value interface, as DHTs, can be used. That is possible, because the DHT stores only information, but does not process it. Thus there are smart 'edge' nodes instead of a smart look-up service.

The use of a DHT in itself, however, does not provide any security or privacy to the users. Ultimately, the security is derived from the use of encryption when storing objects, and more privacy is derived from the fact that no centralized entity stores or gathers all the interactions in the system, or

[4] `www.opendht.org`
[5] `http://www.planet-lab.org`

mines all the data stored in the system.

3.2 Prototype Protocols

This section describes how the architecture is implemented and what the different protocols look like. There are three basic protocols; login, file discovery, and notifications for asynchronous messages. As a DHT usually works with key-value pairs, we implemented all the protocols based on this. A key-value pair consists of a key that can be searched, the get method, and the values, which are the data items associated with the key. Any meta information, e.g., which user stores a file or the IP address of a user, is expressed as a key-value pair with a certain syntax. The details of this syntax can be found in [8]. The protocols rely on user IDs, which are discussed next.

3.2.1 Globally Unique IDs (GUID)

User identities must be unique, and the system should be resistant to impersonation attacks. The advantage of server-based systems is that a user can sign on with an ID. The server can verify it and manage the state of a user. This is more difficult in a distributed system. As a first approach for the prototype, we assume that everyone today has an e-mail address that is unique. There is no further registration process necessary. To prevent a malicious DHT-node from collecting e-mail addresses, a hash of each e-mail address is computed and used as an ID. The hash is one-way, preventing computation of the e-mail address. Another option for a unique ID is to use the hash of the public key of a user. To address impersonation concerns, challenge/response protocols can be used, but are currently not implemented in the prototype.

3.2.2 The Log-in Procedure

A log-in to the system is the announcement that a certain user is now online along with the meta-data necessary to connect to this user and a list of files that the peer stores. This announcement is sent to OpenDHT in the format of a key-value pair. The following example serves to illustrate the login procedure in more detail.

Assuming a user *A* wants to join PeerSoN from her PDA. *A* uses the get-method of OpenDHT with the user's own GUID as key. The results returned by the DHT are all values stored for *A*. These are the different values for each of *A*'s locations, for example her home PC. The value for each location contains also the state for the location, such as *offline* or *online*. These states need to be changed every time *A* changes her location. To do that, all the key-value pairs are substituted by those containing the currently correct state. To log out, *A* substitutes the key-value pair with one containing the state value *offline*. If a friend of *A* wants to communicate with her, the friend retrieves all values for *A's* key and learns that *A* is currently reachable on her PDA.

This protocol allows users to track the state of their friends. Similarly to server-based architectures, users can log in from different machines, which are called locations. Their friends can tell them apart by the different log-in values. It is possible to have more than one value associated with a key in OpenDHT. In that case, OpenDHT returns all these values to a get-request.

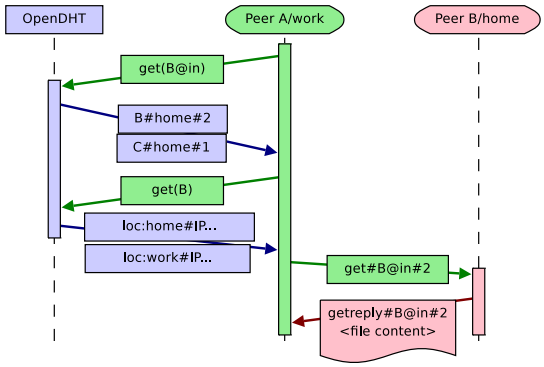

Figure 2: Peer *A* gets the file *B*@in. (Pseudo code)

3.2.3 Getting A File

We now describe how file discovery is handled in PeerSoN. For example, a user *A* logs in and wants to know if there is something new from her friend *B*. If a user wants to receive a file she first needs to know who has the file in the latest version – key-value pair for files – and then find out how to establish a connection to that peer – key-value pair for user information. To learn all news of *B*, it is enough for *A* to get *B*'s index file. This type of file contains all file names related to *B*'s profile. *A* sends a request for this key, the file name, to the DHT. From the returned values, *A* learns who has the file and in which version. Receiving an updated (or the latest) version of *B*'s index file allows *A* to compare it with the local older version. This way, *A* can find out about updates and new files. If the index file has a newer version, *A* retrieves the contact information of the user *A* wants to get the file from. This does not need to be *B*. Figure 2 shows a detailed example with all exchanged messages. To get all missing or outdated files listed in *B*'s index file, *A* just needs to repeat this procedure.

All files, including index files are protected with appropriate access control mechanisms in the design of PeerSoN. This is, however, not yet implemented in the current prototype.

3.2.4 Asynchronous Messages

PeerSoN allows for asynchronous messages to enable services similar to wll messages in Facebook or the inbox, even when the receiver is offline. In the current prototype implementation, this is solved by storing such messages within the DHT until the receiver logs in again. Assuming a user *A* wants to send a message to her friend *B* but learns that *B* is offline. *A* sends the message together with the sender GUID as a value to the DHT. When *B* logs in again he can retrieve

the messages by using a special key-type *B*@no, which is reserved for notifications. This request for notifications is always done as the last step of the log-in procedure.

Due to limitations of OpenDHT, these messages currently contain a maximum of 800 characters plus the header information and remain in the system for only a week.

3.2.5 Sending Files Unasked

What if a peer opens a connection to another peer and then sends a large amount of data to a mobile phone? For getting a file, a handshake protocol is already implemented. This kind of protocol can be extended by a parameter on file size. The receiving peer then can decide if she wants the file to be sent or she can reject the handshake. Moreover, if the file turns out to be larger than announced, the connection can be reset as soon as this fact is noticed. In addition, gray- or black-listing could help to directly reject connection requests from untrusted peers.

A similar problem with large files can occur when nodes log in and get their information and receive a large number of notifications (even with the size restriction of OpenDHT, but more so without) or when nodes download other people's profiles to their phone and the data is large. Handling size and the number of files is thus needed for both synchronous and asynchronous data transfer.

In addition to content that is inopportune to download at the moment due to resource constraints of a device, content can be simply unwelcome and be considered spam. The question of spam is twofold, (a) random peers uploading lots of data into the system that is then replicated by other nodes and (b) peers connecting to friends that they either do not know, or that turn out to be untrustworthy, or virus ridden, and those friends send unwanted content.

4. EXPERIENCES, INSIGHTS, AND OPEN PROBLEMS

At this early stage, even without access-control related security considerations, implementing and deploying PeerSoN at relatively small scales have provided us with insights and highlighted challenges that cannot be directly met with the current understandings of peer-to-peer systems.

4.1 Lessons from Using OpenDHT

The following challenges occur with OpenDHT in particular and, to a large extent, DHTs in general. They are logical centralization, a third-party provider nature, storage timing limitations, security vulnerabilities, performance, and availability issues. We discuss these in turn in this section.

When a user goes online, her peer client can collect her personal messages from OpenDHT, as explained in Section 3.2.4. The peer can then delete these messages from OpenDHT as a means of garbage collection. Since an item is deleted from OpenDHT by sending a delete instruction for a key-value pair, anyone can delete such a message. This means that malicious nodes can delete offline messages before these messages are read by their legitimate destination. While not currently implemented in the prototype, such malicious behavior is easy to thwart while using OpenDHT. This is because OpenDHT allows users to specify a secret key that is necessary in order to carry out a deletion. Thus, the message sender can generate a random secret and store it along with the rest of the encrypted message, which can then only be read or deleted by the legitimate recipient.

Besides OpenDHT being logically a separate and central entity, which does not augur so well with the spirit of complete decentralization envisioned as an ultimate objective of PeerSoN, OpenDHT has some practical limitations. In particular, OpenDHT is a third-party service with its own constraints such as a maximum time period of seven days of storage. Furthermore, OpenDHT is logically centralized and has in recent times had quite some down time, which could potentially become a bottleneck and single point of failure.

One case of when this matters is asynchronous messaging. When a user is online, accessing her digital personal space or sending her messages directly is straightforward. However if a user is not online, then mechanisms are necessary to guarantee availability any time for either accessing her digital personal space, or for any kind of communication with her. In the current, proof-of-concept, PeerSoN implementation, any off-line short message can be stored in OpenDHT, as described ins Section3.2, encrypted using the target peer's public key. It uses OpenDHT as a centrally-managed third-party service that ensures that participating nodes in the OpenDHT infrastructure adhere to the behavior necessary to run OpenDHT. This assumption cannot be sustained realistically. Just to take one example, nodes can drown OpenDHT with messages [7]. Along with the performance concerns, this makes using OpenDHT an unlikely choice going forward.

Thus, in the longer run, PeerSoN peers will run a self-contained DHT, taking into account resource heterogeneity. But in doing so, PeerSoN will lose the assurance that the participating peers actually enforce the DHT protocols correctly. Thus securing the DHT itself against all sorts of malicious and selfish behaviors, which has been a long-standing and yet not fully addressed research problem, becomes crucial. Since the applications are social in nature, we plan to utilize social trust — which is not present in most traditional DHTs — to make the DHT reliable.

4.2 Dealing with Storage in P2P OSNs.

Traditionally the study of P2P storage systems has focused on guaranteeing availability and long-term durability of content, considering archival and back-up applications, where large files are typically accessed infrequently. Such systems can use coding techniques for storage space efficiency. However, in OSNs, many small objects need to be stored, which are then accessed frequently and each object is accessed by many users. The storage layer of PeerSoN must

take into account these peculiarities of OSN usage characteristics, which will imply the usage of replication techniques rather than coding, which in turn has much higher storage overheads.

Another issue not explicitly studied in current P2P storage literature is the total available storage capacity of the system. Particularly given that PeerSoN will store small objects which will be accessed frequently and hence replication instead of encoding is more suitable, the storage overhead for reliability will be considerably higher.

In traditional file-sharing systems, people store only files they are personally interested in. Therefore, total storage capacity contributed to the system is not an issue, but availability of any specific file is not guaranteed. Users typically trade bandwidth incurring temporary load, and the system is sustainable as long as each peer contributes as much upload volume as they download.

In storage applications, each user needs to not only contribute bandwidth, but also contribute storage space, and given the need to maintain redundancy, users need to provide much more storage space than they themselves consume. Such asymmetry will make it difficult to sustain a P2P storage system based purely on altruism.

Thus, for a limited total storage capacity of the system, even under legitimate usage scenarios, the system can get saturated. Spam like denial of service attacks storing huge files can further aggravate the problem to saturating the system's storage capacity. Some of these concerns are addressed in Section 3.2.

Traditional P2P storage systems have focused on placement of redundant objects at random peers or peers determined by DHT name-space. In recent backup systems like Friendstore [9], users use their friends' storage space to store data. Back-up systems however focus on durability, and availability is not so critical. Thus none of these approaches by themselves can guarantee a 24/7 availability, while a OSN will require exactly that. Exploiting geographic diversity along with typically diurnal behavior of users, besides using friends' storage space, are alternate strategies which all need to be taken into account in providing a reliable storage system, while reducing the storage overhead.

Finally, even if and when a technically well-engineered and mature P2P OSN is developed, the asymmetric resource demands on individual users can be a disincentive to participate, and without a critical mass of users in the system, such a system cannot be sustained. Similar to DHTs, a quick glance of literature in P2P storage systems will make it look like mature, and yet, when deploying a P2P OSN, the current state of the art turns out to be inadequate for the specific requirements, as evident from the above discussions and open questions.

To summarize, despite many years of research on DHTs as well as on P2P storage systems, the following issues need further investigation.

Asymmetry of resource demand versus supply: Storage intensive applications require much more resources to provide the necessary redundancy and resilience than what each individual users perceive to consume.

Geographic and temporal diversity: Existing redundancy placement strategies in P2P storage systems are almost all heuristics - including placement at random, DHT determined placements as well as at friends. Exploiting temporal and geographic diversity of nodes systematically can reduce the level of churn for each object, and thus in turn reduce the storage and maintenance overheads.

4.3 Next Steps

We have tested the PeerSoN prototype implementation and verified its functionality on a small scale. The next step for evaluation is a larger-scale set of experiments on PlanetLab and simulations targeting metrics such as response-time, scalability, and availability. For the metric of response-time, one question to answer is the effect that distribution and encryption have as compared to traditional web-service OSNs that, although complemented by content distribution systems, are more centralized. User modeling of time behavior and geographic distribution will inform the experiments on availability. In turn, availability results can deliver input for design decisions such as replication policies.

To determine robust as well as sustainable storage mechanisms for PeerSoN, we are currently looking at user and friends' geographic diversity, as well as exploiting other social characteristics like using friend of friends. While these considerations will take into account time-of-day effects of different time zones, the time properties of content can also inform design decisions. Some of the interesting directions include exploiting geographical diversity in the user base for finding popular content and developing a data replication strategy using a ranked-based friendship model, suggesting those users who are likely to be friends [6].

5. RELATED WORK

ePOST [5] is a P2P-based email system, where users provide storage to be a part of a distributed mail server. The messages are stored within the DHT until they are delivered. PeerSoN, however, is developed to provide additional OSN services, namely user profile management, and enabling communication among users via reading or writing to their shared space. The DHT is mainly used for lookups, not storage.

Cutillo et al [2] propose using a P2P substrate to solve privacy issues in current OSNs. The proposed solution relies on three component of their architecture: a trusted identification service, a P2P substrate, and a matryoshka (rings of trust for storage). In contrast, PeerSoN does not assume any kind of trust relationship between peers but provides access control by encryption and key management.

Yeung et al [4] develops a framework to export user's FOAF[6] profiles and store them on dedicated trusted servers.

[6] `http://www.foaf-project.org/`

Users query and manage profiles through Web-based protocols, e.g., WebDAV[7] or SPARQL/Update[8]. Using a set of dedicated servers for storing user's data, however, might have further security and privacy issues.

NEPOMUK[9] is an on-going EU project with the goal to develop a middleware for sharing users' desktops with friends for online collaborations and sharing of knowledge by exploiting Semantic Web technologies. In Social VPNs [3], users can query different social networks to discover friends to build a Virtual Private Network (VPN) with them.

Commercial initiatives to develop decentralized infrastructures for OSNs are available. Tribler[10] uses a P2P infrastructure for video-on-demand application among friends. Wuala[11] is a partially decentralized encrypted file system, allowing users to share files freely and securely over the Internet. Diki[12] is a social bookmarking service that allows users to encrypt and share bookmarks with friends via the XMPP real-time communication protocol[13]. The storage of data however relies mostly on XMPP servers. 2peer[14] enables users to connect to their friends through a private Internet via a Firefox extension or a downloadable software.

6. CONCLUSIONS

In an attempt to fortify OSNs with privacy for the users, we presented the PeerSoN system which combines a P2P infrastructure with encryption and direct exchange between users' devices. This paper focuses on the description of how peer-to-peer networking can provide OSN features. The prototype architecture is a two-tier system, consisting of peers communicating with each other and a separate look-up service. We developed protocols for information exchange between peers and the lookup service as well as among peers directly. These protocols span the range of sessions supported in a typical online social network, such as logging in and out, finding friends or other peers and up-to-date content, and adding and removing content. The next step is to integrate our solutions for security, primarily encryption and access control, into the architecture, protocols, and the implementation itself. A more thorough analysis of the overhead and robustness of a security and privacy preserving mechanism in PeerSoN is an essential issue, and is a work in progress. The results will be combined with the presented prototype.

Acknowledgements

We would like to thank the five anonymous reviewers and our shepherd Meeyoung Cha for their useful inputs and very pertinent questions, some of which remain open issues which we unfortunately are yet to be in a position to answer, while the others that we could address have helped to improve the paper significantly.

[7] http://www.webdav.org/
[8] http://jena.hpl.hp.com/~afs/SPARQL-Update.html
[9] http://www.nepomuk.org
[10] http://www.tribler.org
[11] http://www.wuala.com
[12] http://www.pace-project.org/
[13] http://xmpp.org
[14] http://2peer.com

7. REFERENCES

[1] Sonja Buchegger and Anwitaman Datta. A case for P2P infrastructure for social networks - opportunities and challenges. In *WONS 2009, 6th International Conference on Wireless On-demand Network Systems and Services, Snowbird, Utah, USA*, February 2009.

[2] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Privacy preserving social networking through decentralization. In *WONS 2009, 6th International Conference on Wireless On-demand Network Systems and Services, Snowbird, Utah, USA*, Feb 2009.

[3] Renato J. Figueiredo, Oscar P. Boykin, Pierre St. Juste, and David Wolinsky. Social VPNs: Integrating overlay and social networks for seamless P2P networking. In *17th IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises*, June 2008.

[4] Ching man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, 2009.

[5] Alan Mislove, Ansley Post, Andreas Haeberlen, and Peter Druschel. Experiences in building and operating epost, a reliable peer-to-peer application. *SIGOPS Oper. Syst. Rev.*, 40(4):147–159, 2006.

[6] Liben D. Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–11628, 2005.

[7] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. OpenDHT: a public DHT service and its uses. In *SIGCOMM '05*, August 2005.

[8] Doris Schiöberg. A peer-to-peer infrastructure for social networks. Diplom thesis, TU Berlin, Berlin, Germany, December 17, 2008.

[9] Dinh Nguyen Tran, Frank Chiang, and Jinyang Li. Friendstore: cooperative online backup using trusted nodes. In *SocialNets '08: 1st Workshop on Social Network Systems, Glasgow, Scotland*, April 2008.

[10] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.