Scalability and Privacy Issues in Social Networks

Samir Hasan
Dept. of Computer Science and Software Engineering
Auburn University
Auburn, AL
Email: szh0064@auburn.edu

***Abstract*—Online Social Networks (OSN) are one of the largest and most dynamic systems on the web. They are fed with personal information and social activities by numerous users through internet, perceived by the users as a personal tool of communication and representation. Friends, families, employees and people all over the world can gather into one central network – the social network. Thus, inherent to these systems is the ability to scale. Without users, social networks are ineffective. On the other hand, the most important *social* attribute that OSNs should have is ensuring privacy. Users want to trust the systems of their personal information and activities – no other users, or the system itself, should use/misuse it without granted permissions. The paper explores whether architectural choices can impact these quality attributes. We have considered Facebook, a centralized social networking system, and the distributed Peer-to-Peer social network model employed by several applications, and evaluated the systems on their support for scalability and privacy. We discovered that the distributed peer-to-peer architectural model puts the user in better control of his data than Facebook, and is also scalable at a cheaper cost.**

I. Introduction

Online social networks provide a virtual social existence to a huge population of Internet users as of today. Social Networking Sites (SNS) are built around users with common interests, shared values, profession, location, etc. Using the Web as the technology medium, they can reach a broad variety of users, facilitating communication between friends within the network. These networks come with various flavors depending on the type of members. Nevertheless, all web-based social network services cater for three primary needs: an interface to create a public/restricted user profile; a means of discovering other users with common interests; and a way to browse through others' connections. These three are the core components of any social network, although individual sites may name these entities differently (e.g. friends $\rightarrow$ fans $\rightarrow$ followers).

Since their introduction, SNS such as Twitter and Facebook have attracted millions of users, and the numbers are growing by leaps and bounds. Many users actually spend most of their daily life interacting within these virtual societies. With usage, the network grows as more and more users join, based on attributes like common language or racial, sexual or nationality-based identities. Thus, one inherent quality of an online social network is its *scalability* – it should be able to grow to accommodate users as they increase in numbers. The application must scale adequately to serve a large and growing number of users; the data stores must also be designed to keep track of the social interactions of the users to enhance the user experience. This large amount of data must also be readily available, which is also an important quality attribute for the social network system.

With a large number of users tracking their social life online, one must be able to control disclosing personal (private) information to unintended recipients. Most of the information that we share online, even the secret ones, actually become lawful property of the SNS company [1]. The SNS actually stores an archive of personal information and activities of a user in a persistent storage [1], instead of replacing the old information. Many of us do not realize that privacy has already been jeopardized. We are brought down to entrusting the service company or agency with our information, willingly or not. Companies may use these data for variety of purposes, such as using social data to show better targeted ads as Facebook does it. Apart from this, even if SNS companies promise not intercept with our data in any way, other privacy issues still arise when using the application. How does the application make sure that the data shared by a user is not visible to any inappropriate individual? Can an employer observe the social activities of an employee? Can a user decide who sees his data and who does not? Apparently, the SNS may give a false impression of privacy when one has to register to the site with personal information such as email, phone number, school affiliation, etc. In this context privacy can be defined as "the right to be let alone" [2], and SNS need to have clean and strict policies to account for all privacy issues.

In this project, we try to identify key issues in scalability and privacy of online social networks by studying two different architecture implementations – a centralized architecture versus distributed P2P architectures for social networks. The architecture of Facebook is considered under the centralized architecture model, although internally Facebook uses distributed implementations of many of their servers. In an abstract view, the distribution of Facebook is an internal aspect of the system, which we do not observe from the outside. Overall, it uses a client-server architecture, so we considered it as an instance of a centralized architecture. Distributed p2p architecture implementations such as PeerSoN [3], Bitgroup [4] and Friendica [5] builds and maintains a social network with peer-to-peer mechanisms. This provides better privacy than centralized architectures and scalability at a lower cost.

II. Evaluation Criteria

A. Scalability

Online social networks are bound to grow. An adequate architecture therefore should address scalability for SNS. A system may grow (or need to grow) both in its capability of managing more data as well as processing an increasing number of client requests. We have tried to generalize the requirement for achieving data-level scalability, and the common approaches to solve such issues. We have looked at the following aspects:

- Data storage (databases) mechanisms
- Distributed file systems
- Availability

B. Privacy

1) Privacy from Unwanted Visitors: We have explored ways through which the social network sites ensures that information is shared only to the intended recipients, or that there is a way to do so if the user wants it. We wanted to see how the systems incorporate features such as:

- User profiles made public or private

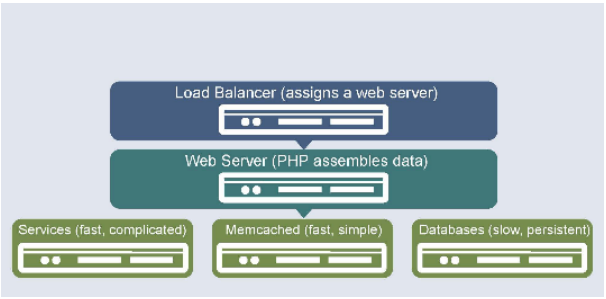

Fig. 1. Simplified Facebook Architecture

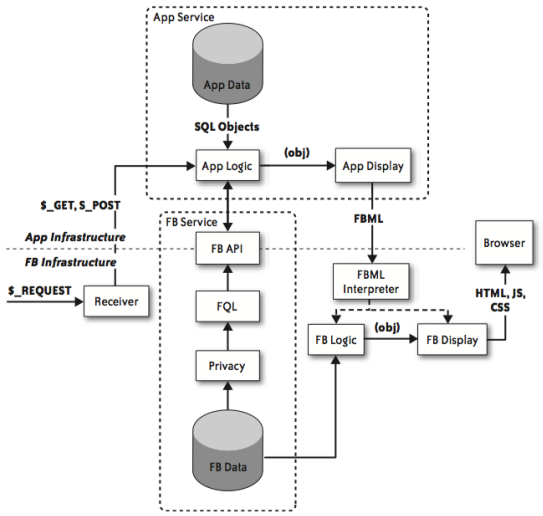

Fig. 2. Internal Architecture of Facebook

- Sharing posts among friends only
- Set different visibility rules for different class of friends
- Stopping/reporting unwanted visitors

2) Privacy from System: So, what is Facebook doing with our private data? User-centric advertising uses our personal information to improve the ad experience. But does it compromise our privacy? We wanted to explore this issue and study how Facebook and other systems allow/disallow such practices. We tried to see if

- Users can control the amount of data system may use for other purposes
- Users can completely remove his information from the system
- Users can have all the data the system stores for him

III. Overview of the Facebook Architecture

As of July 2010, Facebook owned more than 60,000 servers [6] with 300 TB of data stored in Memcached processes [7]. The backend clusters for Hadoop and Hive were made up of 3000 servers having a total of 24k cores, 96 TB RAM and 36 PB disk storage [8]. The traffic records about 100 billion hits per day, 50 billion photos, 3 trillion objects cached, 130 TB of logs per day [8].

Figures 1 and 2 provides an external and somewhat internal view of the Facebook architecture. A short outline of the architectural elements is as follows:

1) The web front-end is written in PHP. Since PHP is slow, Facebook developed the HipHop Compiler [9] that converts its PHP code to C++ and compiles it using g++, thus providing a high performance templating and Web logic execution layer.
2) Later, Facebook developed their own HipHop Interpreter [10] as well as a HipHop Virtual Machine, that translate PHP code to HipHop ByteCode [11].
3) Business logic is exposed as services using Thrift [12]. The implementation of these services can come about form a variety of languages, such as C++ or Java, depending on service requirements.
4) The Java based services do not use any enterprise application server for Java, but rather use a custom server made by Facebook engineers themselves. This is much more lightweight than Tomcat or even Jetty, but works well since much of the service handling is taken care of by Thrift.
5) Persistence is done using MySQL, Memcached [7], Hadoop's HBase [13]. Memcached is used as a cache for MySQL as well as a general purpose cache.
6) Offline processing is done using Hadoop and Hive [14].
7) Data such as logging, clicks and feeds transit using Scribe [15] and are aggregating and stored in HDFS using Scribe-HDFS [16], thus allowing extended analysis using MapReduce
8) BigPipe [17] is their custom technology to accelerate page rendering using a pipelining logic
9) Varnish Cache [18] is used for HTTP proxying. They've preferred it for its high performance and efficiency [19].
10) The storage of the billions of photos posted by the users is handled by Haystack, an ad-hoc storage solution developed by Facebook which brings low level optimizations and append-only writes [20].
11) Facebook Messages is itself an interesting implementation of its own architecture, which is notably based on infrastructure sharding and dynamic cluster management. Business logic and persistence is encapsulated in so-called 'Cell'. Each Cell handles a part of users; new Cells can be added as popularity grows [21]. Persistence is achieved using HBase [22].
12) The typeahead search uses a custom storage and retrieval logic [23]
13) Chat is based on an Epoll server developed in Erlang and accessed using Thrift [24].

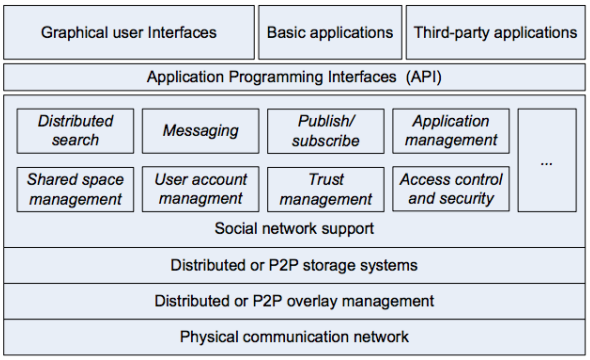

Fig. 3. The general architecture of a distributed online social network

IV. Overview of Distributed P2P Architecture

Figure 3 illustrates an overview of a distributed peer-to-peer architecture for online social networks based. This can serve as a general basis for the host of P2P applications discussed in this paper.

The lowest layer of the decentralized architecture for online social networks denotes the medium for communication, which can be accomplished through Internet or any other networking mechanisms. On top of the physical layer is the distributed P2P overlay management services, which is responsible for managing a host of trusted distributed servers or peers for the communication to take place. This layer ensures reliability of message transfers, and provides lookup and routing mechanisms for resources on the network. The decentralized data management layer provides database-like capabilities for resource query, insertion and update to simplify persistent resource management.

Analogous to the Application layer for the OSI model, the social networking layer provides the core functionalities for our application, namely Distributed Searching, User account, Shared Space Management, Trust Management, Access Control, Security, and finally, extensibility towards third-party applications (Application management) through a well-defined API. The Presentation layer equivalent of this model contains the user interface and applications developed on top of the social network API. The distributed nature of the application is opaque to the users in that the users feel they are using a centralized system.

V. Scalability

Facebook, although a centralized system from the outside, scales very well for its data and application. This is because, behind the single server point of entry, Facebook employs a number of distributed mechanisms to store and query huge amount of data. It uses Hadoop, an open-source distributed file system that stores huge amount of data and provides very high bandwidth for retrieval. For large scale data processing and query, Hadoop MapReduce framework is employed. This is good for unstructured data storage, such as images, videos and files. Facebook also uses HBase, a distributed non-relational database modeled after Google's Bigtable. HBase supports random, realtime read/write access to very large tables with billions of rows and millions of columns.

What makes a decentralized architecture suitable for online social networks? The peer-to-peer architecture is inherently scalable [25] with the cheapest means, which is a great advantage for social networks that requires a growing infrastructure. The advantage with this model is similar to that for deploying in the cloud – the network (hence the infrastructure) can grow (or fall) with popularity. There is surely a cost advantage that decentralized architecture brings for social network applications.

As for the data storage, most of the P2P OSNs work with the users storing all their social data locally [26]. A wall or news feed as that in Facebook can be realized in the distributed OSN as broadcasting profile updates to friends through a topic-based publish-subscribe system. Users can create topics on which their friends can subscribe and receive any update the user made thereof. A peer-to-peer infrastructure supports the publish-subscribe system to maintain the information flow among friends.

With regards to availability, an important constraint for peer-to-peer systems is that there can be a significant number of users who are offline in the network. This means that if users are storing their own information with themselves, they must remain online simultaneously to share information such as profile updates. The peer-to-peer overlay resolves the high-connectivity requirement by ensuring that in each social overlay, of which there can be many, there is at least one replica of a user's profile information. One strategy is to replicate profile information in friends' or friends of friends' nodes. These nodes keep each other updated with regular pings. When one node fails, the oldest node creates a new replica and effectively restores the social overlay. There can be many variations on who to choose for the new node, but the idea is central. To ensure security, the shared information must be encrypted.

VI. Privacy

What is the problem of sharing of private information on social networking sites? Reports have identified several such issues arising from unintentional sharing of private information through social networks [27]. Teenagers are prone to sharing too much personal information online [28]–[30], some of which lead to serious issues such as stalking, bullying and rape [31]. Some companies crawl social networks to gather personal information for creating well-targeted ads [32].

Users in the social networking sites such as Facebook and MySpace can create their own content. The personalization results in increased attachments to the web-life, and thus teenagers are more involved in sharing their thoughts and experiences through the media. As a consequence, teenagers often leak out personal information, mostly unknowingly. This may contain their or their parents' real names, home address, location and birth date, which are some crucial pieces of information for confirming an identity.

Online marketers can use this publicly available private information from the social networks, and create specialized ads. While this may help the marketers a great deal, this may very well be considered as an invasion of privacy. Already companies like Apple and Coke are using social networks to promote their products, and may other organizations advertise through their Facebook page.

This breaching of privacy can have serious impact. Schools can discover the students activities through the social networks they use, and investigate on matters that lie well outside school affairs. Employers may track the social life of their workers, which may impact the durability of their jobs.

All these highlight some privacy related issues that social networking sites raise. It is important to look into how Facebook and other distributed peer-to-peer systems handle privacy issues, as it directly affects the goodness of OSNs.

A. Privacy From Unwanted Visitors

Facebook allows users to control who can find him in the network. For unwanted visitors, a user can Block him, which means the unwanted visitor can not see the user on Facebook, and thereby post anything on his wall or message. The user can also choose, instead of blocking, to remain invisible to public searches. This is an application-level implementation of Facebook. For distributed peer-to-peer systems, we have identified several layers that may deal with this issue (Figure 3): Access control and security layer and Application management layer. An implementation of this architecture can easily incorporate such logic in the application layer.

1) Public and Private Profiles: Can users control their (profile) visibility to others? This is a general requirement that goes in-line with the previous. Some users in the social network may choose to remain invisible to everyone, or may want more private space. Both Facebook and the peer-to-peer architecture supports this through the application layer interface.

2) Control Visibility of Activities: Who gets to know when a user updates his profile? Some users may be fine with all his friends knowing about the update, while others may want to control who sees what. Facebook allows users to share posts to a specific user, or a group of users, or to the general public, thus providing multiple privacy levels to choose. The basic distributed peer-to-peer model

denotes that information is shared among friends only. So, if a user wants the share something publicly so that all users on the network may see irrespective of the friendship relation, the architecture may not have a direct support. However, the Distributed Search and Publish/Subscribe layer can be incorporated in the application layer to implement such a feature.

B. Privacy From System

This is the section where centralized systems like Facebook differs greatly from decentralized peer-to-peer social networks. One major reason is the ownership of personal data. We ask the following questions.

1) How much control does the user have on his personal information stored by the system? Are systems using this data for other purposes that users have no control upon?
2) Can users wipe away some his data that he does not want the system to keep?
3) Can users see what information the system stores for them?

Until recently, Facebook allows users to download a copy of their personal data they have stored in Facebook. However, this is not necessarily the same data that Facebook keeps for the users. Facebook, among all other services, provides customized ads to users – anyone living in Auburn, Alabama is more likely to see ads on restaurants near Auburn than anyone living in New York. The location information is exposed to the ad repository to produce better targeted ads. Some take this as invasion of privacy, since the user can not opt out from this feature. There is no way a user can declare his location (or any other personal information that Facebook tracks) to Facebook and keep it completely private. This lack of control is the reason why distributed peer-to-peer systems have come up – they solve the privacy issue raised by the system itself.

Distributed peer-to-peer OSNs do not store a user's personal data on any central repository. Rather, each node carries its own information, along with information on some other nodes. However, the foreign information is encrypted, so privacy breach is impossible. Users know where his data is located, and has complete control on who sees it and uses it. This also means he can wipe away his data completely from the system, while systems like Facebook may keep a copy of his data even when he deletes his account.

VII. Discussion

We have looked at both centralized and decentralized architectures for online social networks. Facebook serves an extremely powerful centralized system, or more accurately, a hybrid. From an external view, it is centralized – there is a single point of entry for requests on Facebook, and Facebook solely owns the data it houses. For the decentralized peer-to-peer architecture, we have seen data to be treated very personally, where each user carries his own information, so no one else really owns it.

With respect to scalability, peer-to-peer architectures are inherently scalable, both for data and application. Each additional user in the network acts both as a computational unit and a data storage unit. Replication of data in the social overlay ensures availability. The shared data is encrypted to prevent security problems from arising. Facebook on the other hand has employed several distributed architectures behind their centralized interface – it uses distributed file systems and dynamic scaling of application servers for the messaging app where the number of server nodes increase when there is a heavy load. But a centralized architecture is not inherently scalable. Thus, for a software system of the scale of a social network, designing it on top of a distributed peer-to-peer design would ensure scalability form the first place. This may also be a advantage for small organizations that can not afford to pre-allocate huge amount of resources.

Distributed peer-to-peer systems beat centralized architectures like Facebook in its handling of privacy. Although the recent version of Facebook has enhanced its privacy support, the biggest question that arises from letting a company handle our private data is whether we trust them, and whether they are transparent in what they do with our data. Facebook allows more privacy controls now, but still uses our data for targeted ads, for example, that we can not control. For the peer-to-peer system, data is completely owned by the users – each user controls what to do with his data, who to share it with, and whether to wipe it clean from the system. With respect to privacy, distributed peer-to-peer systems clearly has an upper hand over centralized systems like Facebook.

VIII. Conclusion

We have seen examples of two different architectures focusing on scalability and privacy issues, two quality attributes that are inherent in online social networks. Distributed peer-to-peer systems are better at ensuring complete privacy, while for scalability, both the architectures work well. As a future work, it would be interesting to investigate the performance of the systems empirically under heavy usage and a large user-base.

References

[1] Susan B Barnes. A privacy paradox: Social networking in the united states. *First Monday*, 11(9), 2006.

[2] Samuel D Warren and Louis D Brandeis. The right to privacy. *Harvard law review*, pages 193–220, 1890.

[3] PeerSoN: Privacy-Preserving P2P Social Networks. http://www.peerson.net/.

[4] Bitgroup. http://www.organicdesign.co.nz/Bitgroup.

[5] Frendica: The internet is our social network. http://friendica.com/.

[6] Who has the most Web Servers? http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/.

[7] Facebook's architecture presentation at Devoxx 2010. http://www.devoxx.com/.

[8] Scaling Facebook to 500 millions users and beyond. http://www.facebook.com/note.php?note_id=409881258919.

[9] HipHop for PHP. http://developers.facebook.com/blog/post/358.

[10] Making HPHPi Faster. http://www.facebook.com/note.php?note_id=10150336948348920.

[11] The HipHop Virtual Machine. http://www.facebook.com/note.php?note_id=10150415177928920.

[12] Thrift. http://thrift.apache.org/.

[13] HBase. http://hbase.apache.org/.

[14] Hive. http://hive.apache.org/.

[15] Scribe. https://github.com/facebook/scribe.

[16] Scribe-HDFS. http://hadoopblog.blogspot.com/2009/06/hdfs-scribe-integration.html.

[17] BigPipe. http://www.facebook.com/notes/facebook-engineering/bigpipe-pipelining-web-pages-for-high-performance/389414033919.

[18] Varnish Cache. http://www.varnish-cache.org/.

[19] Facebook goes for Varnish. http://www.varnish-software.com/customers/facebook.

[20] Needle in a haystack: efficient storage of billions of photos. http://www.facebook.com/note.php?note_id=76191543919.

[21] Scaling the Messages Application Back End. http://www.facebook.com/note.php?note_id=10150148835363920.

[22] The Underlying Technology of Messages. https://www.facebook.com/note.php?note_id=454991608919.

[23] Facebook's typeahead search architecture. http://www.facebook.com/video/video.php?v=432864835468.

[24] Facebook Chat. http://www.facebook.com/note.php?note_id=14218138919.

[25] Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. Decentralized online social networks. In *Handbook of Social Network Technologies and Applications*, pages 349–378. Springer, 2010.

[26] Alexandra Olteanu and Guillaume Pierre. Towards robust and scalable peer-to-peer social networks. In *Proceedings of the Fifth Workshop on Social Network Systems*, page 10. ACM, 2012.
[27] Susan B. Barnes. A privacy paradox: Social networking in the united states. *First Monday*, 11(9), 2006.
[28] T. Bahrampour and L. Aratani. Teens' bold blogs alarm area schools. *Washington Post*, 2006.
[29] S. Downes. Teens who tell too much. *New York Times*, 2006.
[30] J. Kornblum. Teens wear their hearts on their blog. *USA Today*, 2005.
[31] R. Antone. Another isle man allegedly baits teen victim on myspace. *Honolulu Star Bulletin*, 2006.
[32] J Hempel and P. Lehman. The myspace generation. *BusinessWeek*, 2005.