A Meta Social Networking Approach Towards Decentralization

Pili Hu and Wing Cheong Lau

Information Engieering Department
Chinese University of Hong Kong

Email: {hupili,wclau}@ie.cuhk.edu.hk

Abstract

There are a lot of problems associated with centralized Social Networking Services (SNS), e.g. potential loss of accounts, poor privacy controls, etc. Many developers and researchers all over the world now agree that decentralization is the future of social networking. People from many communities have proposed different solutions to address the problem. Despite their attractive features, only a small portion of those solutions managed to jumpstart and only a few of them grow to near million level of user populations. Based on this observation, we envisioned that **migration** is actually the grand challenge of all. The loss of links (social connections with existing friends) is preventing former centralized SNS users from moving to decentralized services (either stand-alone distributed social services or federated social services). To address the problem of migration, one urgent task is to enable flexible and programmable **cross-platform socialization**. Towards this end, we build a lightweight Python-based middleware to unify the interfaces and data structures of heterogeneous SNS. Unlike most prior works, this middleware is user-oriented. With zero infrastructure support (no need of server setup, database setup, etc), users can readily manipulate their social channels programmably. After sharing our view on the future of social networking, we will briefly introduce the architecture of the middleware and demo two sample applications.

1 Social Networking Services and the Problems

Social Networking Services (SNS) like Facebook and Twitter are an essential part of our daily life now. For example, Facebook now have more than 1 billion monthly active users [4]. This is about 1/7 of the earth's population.

Despite the overwhelming success of the existing SNS, the centralized ownership and control of these services have led to serious concerns in user privacy, censorship and operational robustness. The operator of an SNS has full knowledge of the profiles, social relationships and communication activities of their users. It is a high-value target for not only the typical attackers but also many totalitarian regimes which constantly seek to monitor and control information dissemination among their people. The famous PRISM project [8] is an example. Apart from the technical issues, centralized control also leads to some non-user-friendly management styles. See the IndieWebCamp [3] for more examples.

All these observations lead us to pursue **decentralization** of social networking services. With decentralized services, users have more control over their system and data. This in turn gives more freedom for customization. People from many camps have envisioned that decentralization is the future of social networking services. This can be achived by either a stand-alone distributed system like Diaspora [1] or a federated protocol like OStatus [7].

2 Current and Future Landscape

There are many Decentralized Social Networks (DSN) implementations and proposals from both industrial developers and academic researchers. They address some of the problems associated with DSN, e.g. identity management, group management, communication confidentiality, etc. Although they may not fully address all of the problems, they have different attractive features. However, only a small portion of those solutions managed to jumpstart and only a few of them grow to near million level of user populations.

The current landscape of social networks is: 1) there are a few dominant centralized players; 2) there are many different decentralized solutions (systems or federation protocols). The decentralized world is very small compared with the centralized one. Based on the real world observations, we project them to the future:

- Majority of the users still live on centralized SNS. Firstly, most users do not have time, energy and knowledge to care about privacy or other limitations of centralized SNS. Even if they understand the differences, most users are willing to sacrifice security and privacy for functionality (see how people percept cloud services). Secondly, **link loss** makes the new social network less interesting. Even if a consensus understanding of the benefits of DSN is reached, users are still unwilling to move. For example, even after the PRISM scandal [8], people are still active on those sites.

- Minority of the users move to many different platforms due to privacy policy, richer functionality, etc. Those users are early adopters of new ideas even prior to the DSN age. Since different platforms provide different functions, there is no single dominant solution. Specialized SNS may emerge for certain group of users. Note that those heterogeneous platforms include both centralized and decentralized services.

Here are a few remarks regarding the future landscape we envisioned:

- Migration is the biggest challenge on the way towards decentralization.

- Cross-platform socialization will be the main stream in a very long time.

3 A Meta Social Network

As is discussed in the last section, it is unwise to design a DSN from scratch and try hard to attract users. Many excellent federation protocols also suffer the problem because large-scale centralized players do not bother to federate with them. Instead, we can form a network of all the social networks by stitching them together – a “meta social network”.

Nowadays, many users already actively communicate through more than one SNS. It is noteworthy that users often consciously or subconsciously “stitch” the heterogeneous platforms together by selectively relaying messages across different platforms after some manual filtering/ editing. For example, after reading a “juicy” gossip from a blog or subscribed email-list, you may forward it manually to your personal friends on Facebook. In fact, such cross-platform forwarding operations can be viewed as the formation of a meta social networks which overlays on top of the existing SNS.

Our objective is to provide tools/ systems that can help users to better perform cross-platform socialization. Those tools and systems can also **gradually detach** users from existing centralized services and allows a smooth transition to decentralized world.

4 SNSAPI: A Cross-Platform Middleware

We proposed, designed and open-sourced SNSAPI – a cross-platform middleware to unify interfaces and data structures of SNS. Some federation protocols have similar functions. Due to their clean-slate design, existing SNS do not bother to implement them. This makes them less likely a building block for a meta social network.

The design philosophy of most existing SNS (including many DSN proposals) is **service-centric**. That is, they consider from the network's point of view what data structure and interface to use. The SNSAPI is designed in a **user-centric** manner, i.e. accept the fact that the SNS world is highly heterogeneous and unify them for the users. It makes the cross-platform operation much easier for normal users, giving them the programmable flexibility. We envisioned three stages towards a meta social network:

1. Adapt to existing interfaces and try to unify the data structures. This is to allow seamless migration from existing SNS to SNSAPI. User will be detached from centralized services **without loss of existing social connections**.
2. Form a DSN directly among SNSAPI users. The multi-interfacing nature makes the use case of SNSAPI very flexible. In Section 4.4.2, we demonstrate a way to form an Ad Hoc DSN using SNSAPI based on RSS feeds. One can also leverage Emails and other forthcoming communication channels.
3. Get support from existing SNS. When the SNSAPI's way of socialization (i.e. user-centric) becomes well accepted, we may expect the support from existing service providers (e.g. special API). The gain of efficiency and functionality can benefit both system operators and end users.

4.1 A Unified View of Social Networking Services

A key function of SNS like Facebook, Twitter and SinaWeibo, is to support the dissemination of information among a group of users. From this viewpoint, many other conventional communication services (Email, RSS/Atom feeds, SMS, etc) can also be abstracted using the same primitives of SNS. We thus term all of these services as Social Networking Services and include them in the scope of SNSAPI. Table 1 summarizes four parameters of those services. Facebook type of services are *emphasized* as an example.

Parameters	Values
Links	*static*, dynamic
Direction	uni-directional, *bi-directional*
Accessibility	*read*, *write*
Verification	*authorization*, authentication

Table 1. Unified View of SNS

4.2 Two Design Principles

Firstly, we **focus on solving 80% problems**. As a start, we only abstract five methods for SNS, namely `auth`, `home_timeline`, `update`, `reply`, `forward`. The five methods can cover more than 80% daily operations on an Online Social Network (OSN) and are also universal across heterogeneous platforms. We refrain from abstracting platform specific functions.

Secondly, we **stay open with existing services**. We do not want to solve a full stack of problems. Many problems are already solved (probably in a degraded way). We encourage users to take advantage of existing services rather than reinventing the wheels. For example, I want to selectively forward messages from one OSN to my cellphone via SMS. Instead of developing a "SMS" plugin for SNSAPI, I would suggest the user to program the selection strategy using SNSAPI and use IFTTT [6] to perform a straightforward forwarding from RSS feeds to cellphone.

4.3 Architecture

SNSAPI adopts a modularized and layered design. Figure 1 depicts the architecture of SNSAPI, which consists of the following three layers:

- Interface Layer (IL). `SNSBase` is the base class for all kinds of SNS. One can derive the base class to implement real logic that interfaces with those platforms. In SNSAPI terminology, the modules containing derived classes are called "plugin"; the derived classes are called "platform"; the instance of the classes are called "channel". Message and message list types are also defined in this layer.
- Physical Layer (PL). There are many common operations when interfacing with different SNS. We implement them in PL so that plugin authors do not need to build all the logics from scratch. Examples are: HTTP request/response, OAuth, Error definition, etc. Many 3rd party modules are used in this layer. To keep flexibility, we provide wrapper class for most third party modules, so that one can substitute (part of) them with better ones in the future.
- Application Layer (AL). Application authors can directly use classes from IL. e.g. write an auto-reply script by using `RenrenStatus` (derived from `SNSBase`) and only a dozen of lines are needed. To reduce repeated works in batch operations, we developed a "Pocket" class. It is a container to hold multiple channels. For most applications, Pocket should be the Service Access Point (SAP) to SNSAPI. In this way, end users can enable new channels by simple configuration and no intervention from app developer is needed.

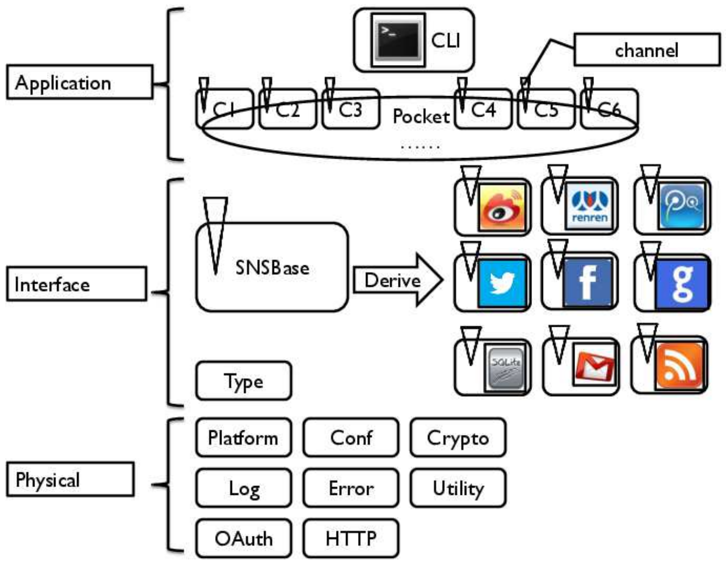

Figure 1. The SNSAPI Architecture

4.4 Example Applications

SNSAPI is very flexible to support many user-centric applications. A group has used it in production to automatically update statuses on multiple channels. We have also build PIXS [5] upon SNSAPI. Equiped with a lightweight ranking framework, PIXS users can be much more efficient in identifying and disseminating information in a cross-platform fashion.

In this section, we briefly discuss two example applications to give the reader a flavour.

4.4.1 An Auto Backup Tool Based on SNSAPI

Many SNS users find it hard to backup their data, especially when their data span multiple SNS. Such a backup tool can be easily written on SNSAPI in a few steps:

1. Register as developers on intended SNS and acquire app credentials.
2. Write a script to invoke `home_timeline` from those SNS and invoke `update` to a local SQLite platform.

This only takes a few lines of coding and some configuration. With more forthcoming plugins, one can easily backup data from many SNS. The major benefit roots in the unified data structure. The entries stored in SQLite DB are ready to support further data mining tasks, e.g. find my use pattern, extract the messages from my best friends, etc.

4.4.2 An Ad Hoc DSN Based on RSS Platform of SNSAPI

RSS platform of SNSAPI can be used to form an ad hoc DSN:

1. Configure a `RSS2RW` (read-/ write- able RSS2) platform.
2. Use SNSAPI (from CLI, GUI, or one's own scripts) to `update` status on it.
3. Users configure a `RSS2RW` for each friend's feeds. When `home_timeline` is invoked, one can get all the statuses of his/ her friends.

This is just a preliminary example. One can use PubSubHub [2] to extend the idea in this section to form a realtime ad hoc DSN. Of course, more infrastructure is required in this case. We have a few remarks regarding this prototype:

- Users who do not use SNSAPI can gracefully degrade. e.g. use feed readers.
- Using SNSAPI can benefit from the meta information we will add in the future.

Bibliography

[1] Diaspora. Available at `http://diasporaproject.org/`.

[2] Pubsubhubhub. https://code.google.com/p/pubsubhubbub/.

[3] Indiewebcamp. http://indiewebcamp.com/why, 2013.

[4] Facebook. Facebook fact sheet. http://newsroom.fb.com/download-media/4227, 2012.

[5] P. Hu, J. Li, and W. C. Lau. Pixs: Programmable intelligence for cross-platform socialization. In *HotPlanet*, 2013. co-located with Sigcomm'13.

[6] IFTTT. Ifttt. https://ifttt.com.

[7] W3C. Ostatus community group. http://www.w3.org/community/ostatus/, 2013.

[8] WashingtonPost. Nsa slides explain the prism data-collection program. http://www.washington-post.com/wp-srv/special/politics/prism-collection-documents/, June 2013.

1. *A longer version is at* `https://github.com/hupili/snsapi/wiki/Introduction`
2. *Project link*: `https://github.com/hupili/snsapi/`