Privacy-Aware and Scalable Content Dissemination in Distributed Social Networks*

Pavan Kapanipathi[1,2], Julia Anaya[1], Amit Sheth[2], Brett Slatkin[3], and Alexandre Passant[1]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway
{julia.anaya, alexandre.passant}@deri.org
[2] Kno.e.sis Center, CSE Department
Wright State University, Dayton, OH - USA
{pavan, amit}@knoesis.org
[3] Google, San Francisco, CA - USA
bslatkin@google.com

Abstract. Centralized social networking websites raise scalability issues — due to the growing number of participants — and policy concerns — such as control, privacy and ownership of users' data. Distributed Social Networks aim to solve those by enabling architectures where people own their data and share it whenever and to whomever they wish. However, the privacy and scalability challenges are still to be tackled. Here, we present a privacy-aware extension to Google's PubSubHubbub protocol, using Semantic Web technologies, solving both the scalability and the privacy issues in Distributed Social Networks. We enhanced the traditional features of PubSubHubbub in order to allow content publishers to decide whom they want to share their information with, using semantic and dynamic group-based definition. We also present the application of this extension to SMOB (our Semantic Microblogging framework). Yet, our proposal is application agnostic, and can be adopted by any system requiring scalable and privacy-aware content broadcasting.

Keywords: Semantic Web, Distributed Social Networks, Social Web, Privacy, FOAF, PubSubHubbub

1 Introduction

Centralized social networking websites, such as Twitter or Facebook, have raised, on the one hand, scalability issues [14] — due to the growing number of participants — and, on the other hand, policy concerns — such as control, privacy and ownership over the user's published data [2]. Distributed Social Networks, such

* This work is funded by Science Foundation Ireland — Grant No. SFI/08/CE/I1380 (Líon-2) — and by a Google Research Award. We would also like to thank Owen Sacco for his work on PPO and Fabrizio Orlandi for his feedback on SMOB.

as SMOB[4], StatusNet[5], Diaspora[6] or OneSocialWeb[7], aim to solve this issue by enabling architectures where people own their data and share it intentionally. While they use different stack, their goal is to allow users to setup their own "Social Space" — as people can do now by setting up a weblog —. Synchronisation between the different user spaces is performed with tools and protocols ranging from XMPP[8] to SPARQL 1.1 Update and its protocol[9] to OStatus[10] or to Activity Streams[11]. Yet, scalability, and most importantly privacy are still ongoing challenges. New techniques are needed to deal with information overload and to ensure content is directed only to intended recipient. This would enable, for instance, to keep a large list of followers/friends[12], and to limit the distribution of particular content to only a subset of people, on-demand (as opposed to generic policies such as "friends" or "family"). For example, limiting content about project X only to project members, this list being dynamically generated.

To achieve this goal, we have built an extension of Google's PubSubHubbub protocol [3] (aka PuSH, described in the next section), that improves both the scalability and the privacy issues in Distributed Social Networks. We enhanced its original broadcasting feature in order to allow publishers to decide whom they want to share their information with, among the users in their social network. Using our approach, content is delivered on-demand to a list of interested parties, as defined by the publisher, but using dynamic preferences. We do this by combining PuSH (including an RDF ontology to describe its core attributes), SPARQL 1.1 Update and a the Privacy Preference Ontology [17] — a lightweight vocabulary for modeling user privacy on the Social Web. Therefore, our approach aims at combining "the best of both worlds", re-using efficient and pragmatic Web 2.0 approaches (PuSH and RSS) with outcomes from the Semantic Web community (lightweight vocabularies and SPARQL). In the rest of this paper we first discuss some background information used in our work (Section 2). We then describe our motivation for, and how we extended the PuSH protocol (Section 3). Further we detail an implementation use-case (Section 4) and conclude with the related work (Section 5).

2 Background

2.1 Distributed Social Networks and PubSubHubbub

Centralized Social Networks (CSN) such as Facebook, Myspace and Twitter suffer drawbacks such as those mentioned in Section 1. For instance, the growing

[4] `http://smob.me`
[5] `http://status.net`
[6] `http://joindiaspora.com`
[7] `http://onesocialweb.org`
[8] `http://xmpp.org`
[9] `http://www.w3.org/TR/sparql11-http-rdf-update/`
[10] `http://ostatus.org`
[11] `http://activitystrea.ms/`
[12] I.e. people allowed to see your content and information

number of Twitter users has been a continuous concern for the performance of the service[13]. Issues related to the sharing of personal information with third party websites by Facebook[14] or Twitter retweet issues [11] have defeated the privacy mechanisms of these services. These lead to new approaches to engineer Online Social Networks (OSN), termed as Distributed Social Networks (DSN) [10]. While CSNs store users' data in their own servers and owns user's data as per Terms of Service, DSNs distribute the data across users, emphasizing on data portability and interoperability. In addition, they promote ownership of users' data, as data resides either on a trusted server or on a local computer.

Implementing DSN requires various layers, including one to transmit data between users' platform. A common way to do this is Google's PubSubHubbub (PuSH), a decentralized publish-subscribe protocol which extends Atom/RSS to enable real-time streams. It allows one to get near-instant notifications of the content (s)he is subscribed to, as PuSH immediately "pushes" new data from publisher to subscriber(s) where RSS readers must periodically "pull" new data. The PuSH ecosystem consist of a few *hubs*, many *publishers*, and a large number of *subscribers*. Hubs enable (1) publishers to offload the task of broadcasting new data to subscribers; and (2) subscribers to avoid constantly polling for new data, as the hub pushes the data updates to the subscribers. In addition, the protocol handles the communication process between publishers and subscribers:

1. A subscriber pulls a feed (Atom/RSS) from a publisher (a "topic" in the PuSH terminology). In its header, the feed refers to a hub where the subscriber must register to get future notifications about publisher's updates;
2. The subscriber registers to the feed at the hub's URL. This process is automatically done by the subscriber the first time the feed is accessed;
3. The publisher notifies the hub whenever new content is published — also, the hub can check for updated directly from the publisher by pulling its feed;
4. Finally, when new content is generated by the publisher, the hub sents updates to all its subscribers for this feed.

PuSH is a scalable protocol, and Google provides a public hub that people can use to broadcast their content[15]. This public hub delivers for approximately 40 million unique active feeds, with 117 million subscriptions. In two years, approximately 5.5 billion unique feeds have been delivered, fetching 200 to 400 feeds and delivering 400 to 600 of them per second. Its largest subscribers get between 20 and 120 updates per second from the hub.

2.2 Semantics in Distributed Social Networks

Within DSN, individuals mapped to each other with their social relationships form what is generally termed as a "social graph", that became popular with

[13] `http://mashable.com/2010/06/11/twitter-engineering-fail/`

[14] `http://www.adweek.com/news/advertising-branding/facebook-facing-more-privacy-issues-126296`

[15] `http://pubsubhubbub.appspot.com/`

OSNs such as Facebook. OSNs and other Social Web services take advantage of the relationships between individuals to provide better and more personalized online experience. In [8], Brad Fitzpatrick, founder of the LiveJournal blogging community[16], discussed his views on building a *decentralized social graph* and the aggregation of individual's friends across sites.

Lightweight semantics can play an important role in social graphs and DSN, allowing to share content between users whether or not they are on the same system. FOAF [5] — Friend of a Friend — is generally used to represent information about individuals (name, e-mail, interests, etc.) and their social relations in a machine readable format. Generally combined with FOAF, SIOC [4] — Semantically-Interlinked Online Communities — is a lightweight vocabulary used (in combination with several of its modules) to represent social data and user activities (blog posts, wiki edits, etc.) in RDF. To a larger extent, vocabularies such as the Open Graph Protocol[17], or schema.org[18] could be considered to model the objects being manipulated and shared by users (movies, photos, etc.) — especially as they may have a larger uptake than the previous ones at Web-scale.

2.3 WebID

To enable users privacy and secure communications in a DSN, an authentication protocol is required. WebID [19] is an decentralized authentication protocol that allows users to manage their own identities and data privacy. It uses X.509 certificates and SSL certificate exchange mechanisms to provide an encrypted communication channel and ensures that users are who they claim, represented by a WebID URI — generally identifying a `foaf:Person`. Hence, FOAF relation may be enhanced with trust descriptions so as to create a reputation network. Moreover, this trust network can be backed by the use of cryptographic keys and digital signatures, so as to form a secure *Web of Trust*[19].

It can also be used for authorization purposes in conjunction with other vocabularies and ontologies such as, Privacy Preference Ontology (PPO) [17] to provide a fine grained access control. In a nutshell, the protocol works as follows:

1. A client sends its X509 certificate (including his WebID URI) to a server;
2. The server extracts the public key and the URI entries from certificate;
3. The server dereferences the URI and extracts a public FOAF profile;
4. The server attempts to verify the public key information. If the public key in the certificate is part of the public keys associated with the URI, the server assumes that the client uses this public key to verify their ownership of the WebID URI;
5. The client is authenticated, authorization mechanism can be applied.

[16] `http://livejournal.com`
[17] `http://ogp.me`
[18] `http://schema.org`
[19] `http://xmlns.com/wot/0.1/`

2.4 PPO - The Privacy Preference Ontology

By itself, WebID does not determine what kind of access an authenticated user has on a resources. Yet, it can be combined with authorization mechanisms to provide such access control. The Privacy Preference Ontology [17] (PPO) is a lightweight vocabulary built on Web Access Control ontology [13] to provide fine-grained restrictions to access RDF data. It consists of a main class `PrivacyPreference` with properties defining (1) the resource to be restricted; (2) the conditions to create the privacy preferences; (3) the access privileges and; (4) the attribute patterns that must be satisfied by the requester — also known as access space. Access Spaces are SPARQL queries, checking FOAF profiles of the requesters to grant access (or not) to some data, so that FOAF plays a central role in the authorization process.

For instance, in a scenario when Alice requests to access to Bob's information (*e.g.* a microblog post), Bob's privacy preference for the corresponding resource are checked. If the access spaces for this preference matches Alice's description (from her FOAF profile), she will gain access to the requested data. A resource can have multiple access spaces, and access is granted if one's profile matches at least one of the access spaces.

3 Extending PubSubHubbub for Privacy-Aware Content Dissemination

3.1 Motivations for Extending PuSH

PuSH provides a distributed architecture and hence more scalability compared to a centralized architecture, but it still does not implement any privacy policies. In CSN such as Twitter, minimal privacy settings are provided to users. Users can either make their account *public* (by default, everyone can view their content) or *protected* (only approved followers can view their content). Yet, the lack of fine-grained privacy policies caused several incidents, such as people being fired because some content reached undesired people in their network[20].

Using PuSH in OSNs brings similar patterns where a publisher can either broadcast his data to all the subscribers or not. Although it would be possible to enable finer-grained access control in PuSH by creating one feed per subscriber; this is considered to be difficult at significant scale. Therefore, we extend PubSubHubbub to feature user-controlled data dissemination. This allows one user to dynamically create groups of people who will receive a private post that remains hidden to other users.

3.2 PuSH extension

The Publisher and the Hub are extended with respect to their counter parts in the original PubSubHubbub protocol, while the Subscriber functionality is kept

[20] `http://www.huffingtonpost.com/2010/07/15/fired-over-twitter-tweets\_n\_645884.html`

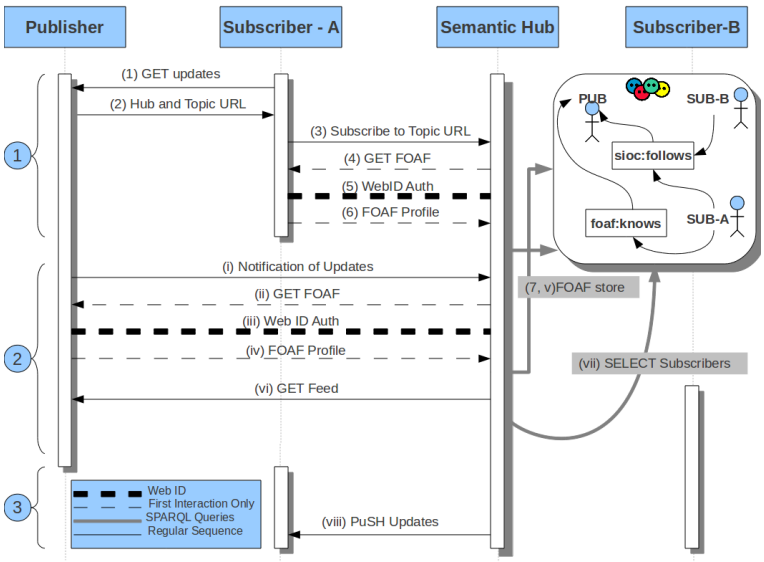

Fig. 1. Sequence of Interactions.

intact. Semantic Web technologies such as RDF, SPARQL and tools such as Triple-Stores are the primary modifications we brought. Following the original design principles of PuSH, the Hub manages most of the complexity of user content dissemination. Therefore, it is solely responsible for pushing feeds to the subscribers explicitly targeted by the publisher. We term this a "Semantic Hub" since it uses Semantic Web technologies and tools to perform this dynamic and private dissemination feature. This is detailed in Sections 3.3 and 3.5.

Fig. 1 illustrates the sequence of interactions between the main participants of the protocol. The ecosystem comprises of the Publisher, the Subscriber and the Semantic Hub. The sequence is divided into three parts (1) *Subscription process* by Subscriber-A (Sub-A) to the Publisher's feeds; (2) *Updates notifications* by the Publisher (Pub) to the Semantic Hub (SemHub); (3) *Updates pushes* to the Subscribers by the Semantic Hub.

The ***Subscription Process*** is independent of the other two whereas the *Updates notifications* and *Updates pushes* happens in sequence. The communication steps in a subscription process begins with Sub-A requesting Pub for its feeds (S-1)[21]. Pub answers with a feed that includes the Topic URL and SemHub URL (S-2). Sub-A then requests the Semantic Hub to subscribe to Pub's feeds Topic URL (S-3). The first communication between Sub-A or any Subscriber with the Semantic Hub leads to the access of Sub-A/Subscriber's FOAF profile by the Semantic Hub (S-4 to S-7). This is further explained in Section 3.3. The

[21] S-X refers to Step X in Fig. 1

interactions that take place only in the first communication are illustrated by dashed lines in the Fig. 1.

In the ***Updates Notification*** the flow starts with an *item* generated by Pub. Once a new *item* is created, Pub embeds its privacy preference for the *item* in the feed. We detail the generation of privacy preferences and how they are embed in the feed in Section 3.4. The preference is a set of SPARQL queries (also known as *access space*) and represents a subset of the semantic social graph hosted in SemHub (Section 3.3). Once the privacy preferences are embed, Pub notifies an update to SemHub (S-i). Similar to Sub-A's first interaction with SemHub, Pub must also provide its FOAF profile to the Hub in order to enable privacy-aware dissemination (S-ii to S-v). This interaction happens only once between a Publisher and a Semantic Hub and is illustrated in Fig. 1 using dotted lines. As soon as the Semantic Hub is notified with the update, SemHub fetches the feed from the Pub (S-vi). Each *access space* for an *item* is executed on the semantic social graph (S-vii). Only the matched Subscribers are eligible to receive the updated *item* from the Publisher.

Updates pushes sequence (Fig. 1) represents the privacy-aware dissemination of the updates only to Sub-A because of the privacy settings of the Pub (S-viii). On the other hand, Sub-B does not receive the updates even though it is subscribed to Pub's topic.

3.3 Distributed Social Graph

FOAF profiles play a crucial role in our architecture. They provide a means for authenticating (WebID) to a platform where users can dynamically create groups for a privacy-aware microblogging. The latter requires a Semantic Social Graph (SSG) where SPARQL queries represent a subset of the SSG — which in-turn forms the dynamic group of people. Generation of the SSG in our protocol consists of collecting and linking the FOAF profiles of people who communicate via the Semantic Hub.

Although collecting the FOAF profiles can be done with secure connections and authorizations, the linking of FOAF profiles in terms of PubSubHubbub protocol required a vocabulary. Since SIOC does not consider communication protocols to represent users' social activities, it is not enough for linking the FOAF profiles using PuSH protocols for further usage. Hence, we created a lightweight vocabulary for PubSubHubbub on top of SIOC [12]. The description of the vocabulary and its usage is explained in the use case (Section 4).

The Semantic Hub uses a triplestore with a SPARQL endpoint to store the RDF data such as the FOAF profiles. The detailed process of collecting, storing and linking the FOAF profiles to enable a Semantic Social Graph is as follows As illustrated in Fig. 1, the Semantic Hub gathers user's profiles during the registration for publishing/subscribing in the following sequence.

1. The user sends a requests for publishing/subscribing at the Semantic Hub.
2. Before acknowledging for publishing/subscription, the Semantic Hub requests the user's FOAF profile.

3. The user authenticates to the Semantic Hub using WebID, further providing a secure connection to the user's personal information stored in FOAF format. As it can be inferred from the sequence, the Semantic Hub has its own WebId URI and certificate for the users to authenticate.
4. The Semantic Hub stores the FOAF profiles with added necessary information about subscriber and publisher in the RDF store. The generation of necessary information in case of SMOB is presented in Section 4.

3.4 Generating Privacy Preference

Creating SSG helps to dynamically extract groups of subscribers from the publisher's social network who are eligible to receive the publisher's post. To do so, the publisher must create preferences to restrict which users can access the data. These preferences are defined using PPO based on (i) SPARQL queries defining *access spaces* to represent a subset of the SSG that can access the data (e.g. people interested in "Semantic Web" among my friends) and (ii) conditions to match *items* that must be delivered with their corresponding *access space*.

Our implementation provides a user-friendly interface to formulate SPARQL queries where no knowledge about SPARQL or Semantic Web is required (Section 4). Formulating *access spaces* for each *item* to be published is not practical. The privacy settings for *items* can have conditions to categorize an item and assign *access space* for the corresponding category. For example, the privacy preference in Fig. 2 restricts any document tagged with *Semantic Web* (categorizing by tags) from the publisher to only those users who share an interest in *Semantic Web*. Since the Privacy Preferences are represented in RDF, a triple store is necessary at the publisher to store and retrieve the privacy settings.

```
<http://example.org/privacy/3> a ppo:PrivacyPreference;
    ppo:appliesToResource
      <http://xmlns.com/foaf/0.1/Document>;
    ppo:hasCondition [
      ppo:hasProperty tag:Tag;
      ppo:resourceAsObject
        dbpedia:Semantic_Web
    ];
   ppo:assignAccess acl:Read;
   ppo:hasAccessSpace [
     ppo:hasAccessQuery "SELECT ?user WHERE {
      ?user foaf:topic_interest dbpedia:Semantic_Web }"
  ] .
```

Fig. 2. Example SMOB Privacy Preference Filtering.

Each *access space* is a subset of the SSG in the Semantic Hub, which in turn is the list of subscribers who are eligible to receive the post. Our implementation

of privacy settings for a SMOB user embeds a default *access space* for the micro-posts that do not have any predefined privacy settings. If there are more *access spaces* then all are embedded into the RSS. Embedding *access spaces* into RSS will include another element `<privacy>` for each item/post in the RSS/Atom feed. An example of RSS *item* including the privacy preferences is shown in Fig. 3. The `<privacy>` element comprises of each *access space* as a child element `<accessspace>`. Also, it allows us to simply reuse and extend RSS / Atom to pass the policies from the publisher to the Hub. The Semantic Hub then uses the embedded *access spaces* to filter the list of subscribers to only those who can receive the *item*.

```
<item>
  <title>Only Friends</title>
  <description>
    Send this to only people I know and
    interested in Semantic Web
  </description>
  <link>http://example.org/rss</link>
  <guid>123123123123</guid>
  <pubDate>March 06 2001</pubDate>
  <privacy>
    <accessspace>
      SELECT ?user WHERE {
      foaf:me foaf:knows ?user .
      ?user foaf:topic_interest dbpedia:Semantic_Web . }
    </accessspace>
    ...
  </privacy>
</item>
```

Fig. 3. Access space embedded in an RSS feed .

3.5 Semantic Dissemination of Content

The final step of the protocol is to disseminate the publisher's content based on the privacy settings for the content. Once the post and its privacy settings are updated in the feed by the publisher, the publisher notifies the Semantic Hub for updates. The Semantic Hub pulls the updated feed and parses it to fetch the updates and the corresponding *access spaces*.

Every updated item in the feed has its own set of *access spaces*. The process for multicasting each item is as follows:

1. Each *access space* for the item is queried on the SSG at the Semantic Hub's RDF store. The result is a subset of the SSG that matches the *access space* executed.

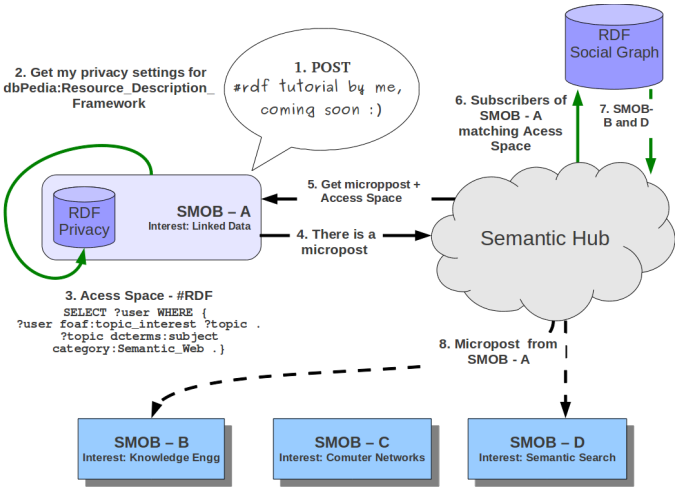

Fig. 4. Sequence of Interactions SMOB.

2. Union of all the subsets of the SSG retrieved by executing the *access spaces*, comprises of the subscribers who are eligible for receiving the *item*.
3. As illustrated in the Fig. 3 the RSS/Atom *items* in the feed also comprises of its own *access spaces* in the `<privacy>` element. However, the *item* with the *access spaces*, if broadcasted will let the subscribers be aware of the privacy settings of the publishers. Therefore, to maintain the privacy of the publishers, the `<privacy>` element is deleted from each *item* at the Semantic Hub.
4. The modified *item* is then sent to only the callback URLs of the filtered subscribers list from step 2.

4 Implementation and Use Case in SMOB

SMOB - `http://smob.me` is a open and distributed semantic microblogging application combining Semantic Web standards and Linked Data principles with State-of-the-art social networking protocols. In the microblogging context, it is common to use the *follower* and *followee* terms, where the *follower* is a user who follows another user's timeline (i.e. her/his microblog posts) and the *followee* is the user generating content and being followed. A user can be both a follower and a followee, and can have multiple followers and followees. Combining the PuSH terminology with this one, we have: a follower is a PuSH *Subcriber*, a followee is a PuSH *Publisher*, a PuSH feed/topic is the *User's Timeline* and each micropost is a PuSH *item* in the feed.

SMOB used the PuSH implementation (using Google's public hub[22]) to broadcast feeds. But privacy was still a concern. We now present the step by step implementation of our protocol in SMOB, enabling both privacy-awareness and scalability when distributing microblog posts.

There is an screencast of the full process available at `http://smob.me/video`

4.1 SMOB Hub-User Initial Interaction

The Semantic Hub creates a SSG of the users communicating via the Hub using the same process than the one explained in Section 3.3. We had to distinguish between the user's social connections on FOAF and the ones in SMOB. For example, in Fig. 4 the access query just mentions all users interested in Semantic Web, whereas we need only (in this subset) those who are followers of SMOB-A. Hence, we modeled the social activities over the Semantic Hub.

We created a lightweight vocabulary for PuSH and SMOB. Fig. 5 depicts our vocabulary, where classes and properties are mapped to the terminology used in core specification of PubSubHubbub [3]. The full specification is available at `http://vocab.deri.ie/push`. As per specification, the PuSH Topic represents the feed of the Publisher. In SMOB, there is a one to one relationship between the Publisher and his Feed. We are thus using `sioc:UserAccount` to each user communicating via the SemanticHub. Instances of `sioc:UserAccount` are linked to PuSH Topics with appropriate properties. We introduced:

- `push:has_hub` to express the relationship between the PuSH Topic and the Semantic Hub. Semantic Hub is considered to be a FOAF Agent.
- `push:has_callback` to express the relationship between Subscriber and its callback URL which is of type `rdfs:Resource`.

Added relations stored with followee's FOAF are (1) newly created unique `sioc:UserAccount`; (2) relation to the PuSH Topic (Followee's Timeline) she/he is creating; and (3) PuSH Topic related to the Semantic Hub. The follower also has similar properties, i.e. (1) newly created unique `sioc:UserAccount`; (2) relation to the PuSH Topic (Follower's timeline) he/she is subscribing to; and (3) Callback URL of the Subscriber. When a follower wants to unfollow a followee, the follower's relation with the PuSH Topic (Followee's timeline) is simply removed.

4.2 SMOB Followee - Publishing

Since SMOB is a semantic microblogging service, it already includes an Arc2 triple store during installation. Therefore the requirement for the Publishers to include a triple store in our protocol was fulfilled. The same RDF store is used to store and retrieve the privacy preferences of the corresponding user.

Further, we frequently refer to Fig. 4 to explain the use case. SMOB-A is the Followee, SMOB-B SMOB-C and SMOB-D are followers of SMOB-A.

[22] `http://pubsubhubbub.appspot.com/`

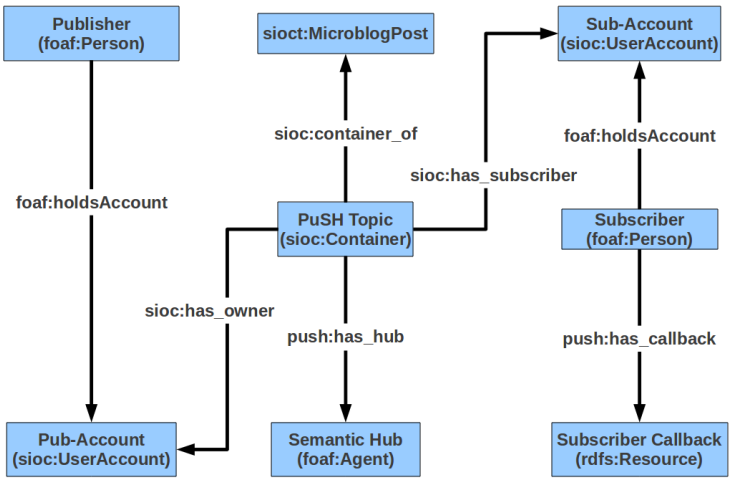

Fig. 5. Vocabulary to represent PuSH information in RDF.

A SMOB user (SMOB-A) has to generate privacy preferences (Section 3.4) for controlling his content distribution. In SMOB, the categorization of microposts is done using "semantic hashtags" mentioned in the micropost (i.e. tags linked to resources from the LOD cloud using models such as MOAT[23] or CommonTag[24]). Hence, we map these tags to the privacy preferences to enable the privacy in content distribution. As shown in the Fig. 6, we have build a simple user-interface to let users create such a setting without prior knowledge of SPARQL or the Semantic Web. Both the hashtag and the interest are looked-up in DBPedia[25] and Sindice[26] and the concepts are suggested to the user. Once the user selects the intended concepts, the resulting Privacy Preference is stored in the local triple store. For example, Fig. 6 shows the interface to create the privacy preferences where the microposts tagged with #rdf, should be sent only to those followers who are interested in Semantic Web. Also, we offer the ability to restrict based on the relation that people share together, using the RELATIONSHIP vocabulary. The main advantage compared to pre-defined groups is that the settings are automatically updated based on the user's attribute, in almost real-time since the Semantic Hub stores users' profiles[27] Once again, these

[23] `http://moat-project.org`
[24] `http://commontag.org`
[25] `http://dbpedia.org/`
[26] `http://sindice.com/`
[27] As said earlier, so far, we assume that the information provided in FOAF profile is correct, or at least should be trusted - which is legally the case with WebID as it requires a certificate which implies some real trust settings.

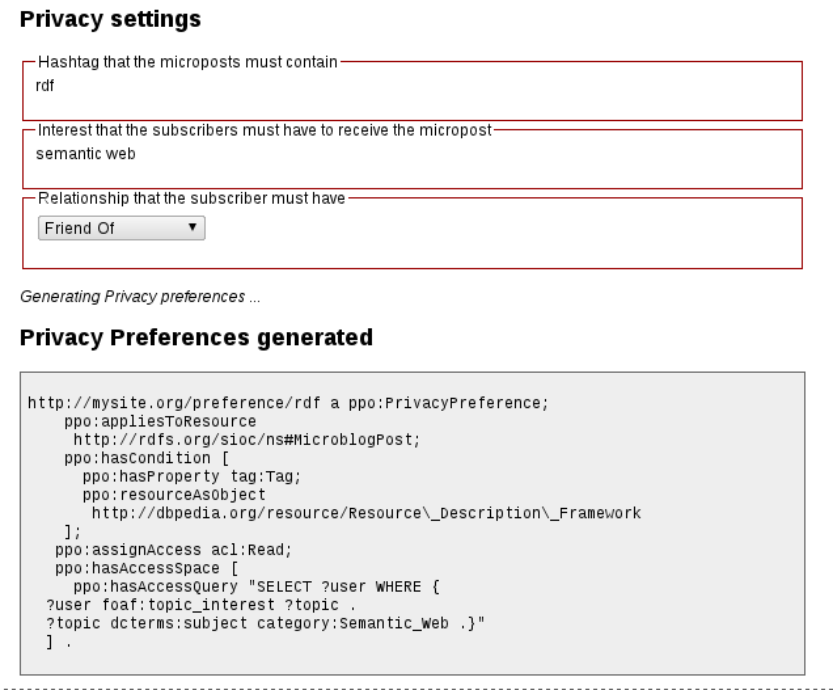

Fig. 6. Privacy Settings interface in SMOB

two use-cases are just an example, and the privacy settings could be further enhanced.

When SMOB-A is creating a new micropost and tags it with the #rdf hashtag (Step 1 of Fig. 4) , the SMOB interface suggests links to the resources with that label from DBPedia and Sindice in a similar way as for the privacy settings. Once the micropost is posted, SMOB-A queries the local triple store for *access spaces* matching the URI representing the hashtag #rdf (Step 2, 3 of Fig. 4). The *access space* is embeded into the RSS feed of SMOB-A.

The SMOB Followee then notifies the Semantic Hub about the update (Step 4 in Fig. 4).

4.3 SMOB Semantic Hub - Distribution

Querying data to and from the Semantic Hub triple store is performed using the Python SPARQLWrapper[28]. The wrapper is entirely HTTP-based, thus gives the flexibility to deploy the RDF store in any environment and access it remotely. To maintain privacy of the profiles, the SPARQL endpoint is accessible only by the Semantic Hub.

[28] `http://sparql-wrapper.sourceforge.net/`

After SMOB-A notifies the Semantic Hub about the update, the Semantic Hub fetches the feed and, for the newly created microposts extracts the *access space*. Before the access query is executed, more conditions are added based on the corresponding relations added during FOAF profiles storage (Section 4.1). In this use-case conditions are added to retrieve the callback URLs of the followers, who holds a sioc account subscribed to SMOB-A's topic. The *access space* is executed against the SSG of SMOB-A in conjunction with the above conditions (Step 5 in Fig. 4) and the list of users (SMOB-B and SMOB-D) matching the SPARQL query are returned (Step 6 in Fig. 4). SMOB-B and SMOB-D matched because their interests are in the category of "Semantic Web" where as SMOB-C has interest "Computer Networks" which does not fall into the category restricted by SMOB-A.

5 Related Work

Related work can be considered from two different perspectives (1) Distributed Social Networks and Semantics, and (2) Privacy in Social Networks. This combination should lead to a distributed and privacy aware Social Web, as envisioned in the final report of the W3C Social Web Incubator Group "Standards-based, Open and Privacy-aware Social Web" [1].

Research on privacy in OSNs are prominently about misuse of private information and therefore providing more security [6] [9]. Very few have done research where broadcasting of user content is taken into consideration. Work by B. Meeder et al. [11] analyses the privacy breach for protected tweets by retweets. Our work more or less falls in the category of awareness in distribution of content. Next, gathering information about users from online social networks to form social graphs [22] [16] has been really useful to leverage the powers of social graphs. Work by Matthew Rowe shows the creation of Social Graphs by exporting information from online social networks into semantic form using FOAF vocabulary [16]. Further to this, few of the areas where the power of such Semantic Social Graphs is leveraged are in finding friends in federated social networks [7] and to disambiguate identity of users online [15].

Efforts such as P3P or more recently policy systems such as Protune also enable rule-based access to content on the Web. However, they are not necessary suited to Web-based environment. Cuckoo [21] is a decentralized P2P socio-aware microblogging system. It focuses on providing a reliable and scalable system to increasing the performance with respect to traditional centralized microblogging systems. Yet, they rely on a more complex architecture which is not only HTTP based and makes it difficult to deploy in practical Web-based environments. FETHR [18] is another open and distributed microblogging system, that emphasizes about the scalability, privacy and security. While using a publish-subscribe approach, information is sent from the provider to all its subscriber (one of the reason while PuSH was build). More recently, [20] is at a close proximity to our work but on mobile platforms. They provide a semantic social network for mobile platforms, but do not tackle the privacy aspect. In

addition, larger open-source projects such as StatusNet or diaspora also enable distributed Social Networks using similar stacks (notably including PuSH and the Salmon protocol[29] for pinging-back comments), but neither directly focus on on-demand and dynamic privacy settings.

6 Conclusion

In this paper, we described how we extended Google's PubSubHubbub protocol to cope with privacy-by-design in Distributed Social Network. We showed how we defined a way to broadcast content only to a subset of one's social graph, without having to hardcode user-group or specify static policies. Hence, our system can cope with dynamic groups and organizations without putting any burden on the user. Moreover, we have shown how we implemented this into SMOB, a Semantic and Distributed microblogging platform. Overall, our approach also shows how to combine pragmatic Social Web protocols and formats (PuSH and RSS/ Atom) with Semantic Web standards and vocabularies (SPARQL Update and PPO).

In the future, we will focus on enabling our architecture for mobile devices. Here, challenges will be to send information from and to devices that can be off-line from time to time, but still need to be notified. Also, we are considering to apply this architecture to Sensor Data, in order to deal with the management of sensitive information such as geolocation.

References

1. A Standards-based Open and Privacy-aware Social web. Technical report. W3c incubator group report 6th december 2010, World Wide Web Consortium, 2010. `http://www.w3.org/2005/Incubator/socialweb/XGR-socialweb-20101206/`.
2. Danah Boyd and Eszter Hargittai. Facebook privacy settings: Who cares? *First Monday*, 15(8), 2010.
3. Brad Fitzpatrick, Brett Slatkin, Martin Atkins. PubSubHubbub Core 0.3 – Working Draft. `http://pubsubhubbub.googlecode.com/svn/trunk/pubsubhubbub-core-0.3.html`.
4. John Breslin, Uldis Bojars, Alexandre Passant, Sergio Fernàndez, and Stefan Decker. SIOC: Content Exchange and Semantic Interoperability Between Social Networks. In *W3C Workshop on the Future of Social Networking*, January 2009.
5. Dan Brickley and Libby Miller. FOAF Vocabulary Specification 0.98. Namespace Document 9 August 2010 - Marco Polo Edition. `http://xmlns.com/foaf/spec/`, 2010.
6. Leucio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communications Magazine*, 47(12):94–101, 2009.
7. Ruturaj Dhekane and Brion Vibber. Talash : Friend Finding In Federated Social Networks. In *Proceedings of the Fourth Workshop on Linked Data on the Web (LDOW2011) at WWW2011*, 2011.

[29] `http://www.salmon-protocol.org/`

8. Brad Fitzpatrick and David Recordon. Thoughts on the Social Graph. `http://bradfitz.com/social-graph-problem`, August 2007.
9. Balachander Krishnamurthy and Craig E. Wills. Characterizing privacy in online social networks. In *Proceedings of the first workshop on Online social networks - WOSP 08*, 2008.
10. 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.
11. Brendan Meeder, Jennifer Tam, Patrick Gage Kelley, and Lorrie Faith Cranor. Rt@iwantprivacy: Widespread violation of privacy settings in the twitter social network. 2010.
12. Alexandre Passant, John Breslin, and Stefan Decker. Rethinking Microblogging: Open, Distributed, Semantic. In *Proceedings of the 10th International Conference on Web Engineering, ICWE'10*, 2010.
13. Eric Prud'hommeaux. W3C ACL System. `http://www.w3.org/2001/04/20-ACLs.html`.
14. Jeff Rothschild. High Performance at Massive Scale Lessons learned at Facebook.
15. Matthew Rowe. Applying semantic social graphs to disambiguate identity references. In *The Semantic Web: Research and Applications*, volume 5554 of *Lecture Notes in Computer Science*, pages 461–475. Springer Berlin / Heidelberg, 2009.
16. Matthew Rowe. Interlinking distributed social graphs. In *In Linked Data on the Web Workshop, WWW2009,April2009*, 2009.
17. Owen Sacco and Alexandre Passant. A Privacy Preference Ontology (PPO) for Linked Data. In *Proceedings of the Linked Data on the Web Workshop, LDOW2011*, 2011.
18. D. R. Sandler and D. S. Wallach. Birds of a fethr: Open, decentralized micropublishing. In *In IPTPS'09: Proc. of the 8th International Workshop on Peer-to-Peer Systems, Boston, MA, USA, April 2009*, 2009.
19. Henry Story, Bruno Harbulot, Ian Jacobi, and Mike Jones. FOAF+SSL: RESTful Authentication for the Social Web . In *European Semantic Web Conference, Workshop: SPOT2009.*, 2009.
20. Sebastian Tramp, Philipp Frischmuth, Natanael Arndt, Timofey Ermilov, and Sren Auer. Weaving a distributed, semantic social network for mobile users. In *8th Extended Semantic Web Conference (ESWC2011)*, June 2011.
21. Tianyin Xu, Yang Chen, Xiaoming Fu, and Pan Hui. Twittering by cuckoo: decentralized and socio-aware online microblogging services. In *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, SIGCOMM '10, pages 473–474, New York, NY, USA, 2010. ACM.
22. Shaozhi Ye, Juan Lang, and Felix Wu. Crawling online social graphs. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, pages 236 –242, april 2010.