SMOB: The best of both worlds

Federated Social Web Europe Conference, June 3rd to 5th 2011, Berlin

Alexandre Passant [1], Julia Anaya [1], Owen Sacco [1], Pavan Kapanipathi [1, 2]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway, Galway, Ireland
[2] Kno.e.sis, Wright State University, Dayton, Ohio

Abstract

This paper presents the architecture of SMOB and the way it combines Semantic Web standards (RDF(S) / SPARQL) and new protocols such as PubSubHubbub to enable a Federated and Privacy-Aware Social Web.

Introduction

SMOB is an open-source microblogging framework that allows anyone to install her/his personal hub, whereas hubs consequently interact in a distributed manner on the Web. Started in 2008, it follows the vision of a federated Social Web, where people own their data and can openly share it without the constraints of a third-party provider. SMOB combines various ideas and technologies from the Semantic Web / Linked Data world to achieve this vision.

While several research work focused on the integration of the Semantic Web / Linked Data and the Social Web, SMOB is following this approach from an architecture and information systems engineering point of view. On the one hand, SMOB data is modelled and exchanged using Semantic Web standards (RDF(S) and SPARQL) and Linked Data principles. On the other hands, SMOB relied on protocols emerging from Social Web systems (RSS, PubSubHubbub, WebSockets). We believe that this combination can be a great benefit for the Federated Social Web, and can also help to tackle the important challenge of privacy.

In the remaining of this paper, we briefly discuss these different aspects that we hope to discuss at the conference. For the sake of keeping the paper focused on these aspects, we do not discuss related work but invite the reader to check the final report of the W3C Social Web Incubator Group [SWXG].

SMOB: Linked Data for the Social Web

Representing Social Data as Linked Data

In order to provide a structured representation of its content, SMOB uses an ontology stack (Figure 1) for the Social Web [SMOB], combining several ontologies such as FOAF, SIOC, MOAT / CommonTag or OPO. This provides a structured representation of every piece of content, in a more structured and interlinked way that simple RSS or Atom feeds.

For example, describing authors as instances of sioc:UserAccount allows to links to their profile on other social websites. Also, while this stack is currently used in the context of microblogging, it can be adapted to any Federated Social Web application, since (1) all models are application-agnostic, and (2) SIOC features extension for various applications (wikis, etc.).

Linking to the Outside World

Further that just representing structural information (authors, date, etc.), our focus is also to provide a meaningful representation of topics. SMOB provides an interlinking system, where `#hashtags` can be linked to URIs representing resources (in the Semantic Web sense, i.e. as online representation of real-world objects), from knowledge bases such as DBpedia or Freebase.

This allows not only to disambiguate tags (e.g. `#apple`), but also to reuse background knowledge from these bases to infer new information about microblog content (for instance, using the fact that rugby is a sport is Galway in ireland to know that "#rugby game tonight in #Galway" is about sport in Ireland). Yet, this interlinking from social data leads to challenge such as synchronisation of remote sources to the local SMOB hub, or distributed queries, that we not discuss here.

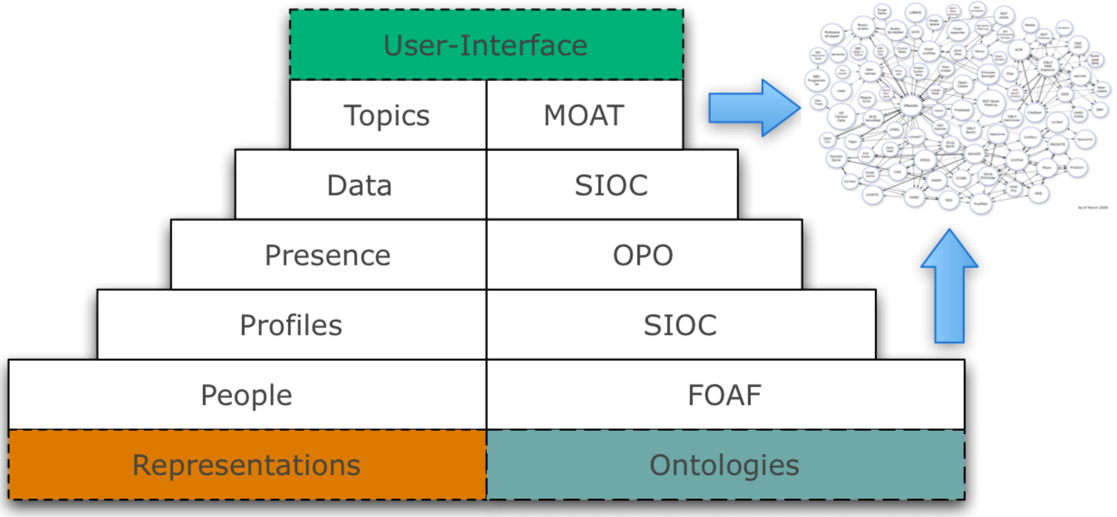

Figure 1: The SMOB Ontology stack

The SMOB Architecture

A Push Approach

A SMOB *subscriber* is a any SMOB hub (person) that is following another SMOB hub, and a SMOB *publisher* is any hub that is being followed (has a follower) by other one. The original SMOB version simply HTTP POST-ed any update from a publisher to all its subscribers. It avoid the inefficient polling of new content (used for instance by feed readers to get new feeds), yes is not scalable. If a SMOB publisher has one thousand subscribers (followers), its hub has to multicast the updates to one thousand subscribers, which can lead to broadband and bandwidth issues (especially if SMOB is installed on a mobile phone, which is one of our current research area).

Thus, SMOB now uses PubSubHubbub [PUSH] for pushing update between hubs. At a glance, PubSubHubBub (or PuSH) is "a simple, open, server-to-server web-hook-based pubsub (publish/subscribe) protocol as an extension to Atom and RSS". It relies on a dedicated server, called *hub* in the PubSubHubbub terminology, used to broadcast any update notifications to subscribers. In our case, this means that a SMOB hub do not need to broadcast information to all its subscribers, but simply ping the PuSH hub that broadcasts the update to all followers.

Another advantage is that subscribers do not need to be online to receive the new/changed content, as the hubbub retries to send the updates several times if they are not online.

Yet, while it supports several formats as Atom, RSS and JSON, RDF is not supported. Since SMOB provides rich content by representing its data in RDF, we extended it so that we encode basic graph patterns of SPARQL 1.1 Update queries in the RSS feeds (see Listing 1). We use the RSS <content:encoded> tag to embed this query. When new content is created, the published embeds the query in the feed, sends it to the hubbub, that sends the new/changed content to the subscriber. When data is received, the feed is parsed, the BGP is extracted and a INSERT query is created, in a particular named graphs that corresponds to the URI of the post (link element of the feed).

```
<item rdf:about="http://example.org/post/2011-04-27T14:26:46+02:00">
  <title>publisher112</title>
  <link>http://example.org/post/2011-04-27T14:26:46+02:00</link>
  <description>publisher112</description>
  <dc:date>2011-04-27T14:26:46+02:00</dc:date>
  <content:encoded><![CDATA[
    <http://example.org/> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://smob.me/ns#Hub> .
    <http://sws.geonames.org/2964180/>  <http://www.w3.org/2000/01
/rdf-schema#label> "Galwaycity, Ireland (seat of a second-order administrative
division)"^^<http://www.w3.org/2001/XMLSchema#string> .
    <http://example.org/post/2011-04-27T14:26:46+02:00> <http://www.w3.org
/1999/02/22-rdf-syntax-ns#type> <http://rdfs.org/sioc/types#MicroblogPost> .
    <http://example.org/post/2011-04-27T14:26:46+02:00> <http://rdfs.org
/sioc/ns#has_container> <http://example.org/> .
    <http://example.org/post/2011-04-27T14:26:46+02:00> <http://rdfs.org
/sioc/ns#has_creator> <http://example.org/me> .
    <http://example.org/post/2011-04-27T14:26:46+02:00> <http://xmlns.com
/foaf/0.1/maker> <http://example.org/me#id> .
    <http://example.org/post/2011-04-27T14:26:46+02:00> <http://purl.org/dc/terms
/created> "2011-04-27T14:26:46+02:00"^^<http://www.w3.org
/2001/XMLSchema#dateTime> .
    <http://example.org/post/2011-04-27T14:26:46+02:00> <http://purl.org/dc/terms
/title> "Update-2011-04-27T14:26:46+02:00"^^<http://www.w3.org
/2001/XMLSchema#string> .
    <http://example.org/post/2011-04-27T14:26:46+02:00> <http://rdfs.org
/sioc/ns#content> "publisher112"^^<http://www.w3.org/2001/XMLSchema#string> .
    <http://example.org/post/2011-04-27T14:26:46+02:00#presence>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://online-presence.net
/opo/ns#OnlinePresence> .
    <http://example.org/post/2011-04-27T14:26:46+02:00#presence> <http://online-
presence.net/opo/ns#declaredOn> <http://example.org/me> .
    <http://example.org/post/2011-04-27T14:26:46+02:00#presence> <http://online-
presence.net/opo/ns#declaredBy> <http://example.org/me#id> .
```

```
    <http://example.org/post/2011-04-27T14:26:46+02:00#presence> <http://online-
presence.net/opo/ns#StartTime> "2011-04-27T14:26:46+02:00"^^<http://www.w3.org
/2001/XMLSchema#dateTime> .
    <http://example.org/post/2011-04-27T14:26:46+02:00#presence> <http://online-
presence.net/opo/ns#customMessage> <http://example.org
/post/2011-04-27T14:26:46+02:00> .
    <http://example.org/post/2011-04-27T14:26:46+02:00#presence> <http://online-
presence.net/opo/ns#currentLocation> <http://sws.geonames.org/2964180/> .
  ]]> </content:encoded>
</item>
```

Listing 1: Example SMOB RSS with RDF triples inside the `<content:encoded>` tag

The current FLOSS hubbub implementations cannot detect when a post has been deleted, i.e. when a post that was published before is not anymore in the last feed update. To do so, we have implemented the following: when the publisher deletes a microblogging post, the post graph is deleted from the triple store (via an SPARQL DELETE query), but instead of deleting the entry from the RSS feed too, its <content:encoded> tag becomes empty and the new feed is broadcasted through the hubbub. Then, the subscribers run a SPARQL Update DELETE query.

PubSubHubBub + Web Sockets

The architecture described so far, will not allow a user running a SMOB hub from her/her laptop in a home broadband or mobile phone to subscribe to other hub (and to get updates from she/he) due to the firewalls and Network Address Translation (NAT). In the same way, this user will not be able to publish updates to hub At this point, we assume that users networks are using Internet Protocol version 4 (IPv4), as the scenario will be rather different using IPv6.

To tackle this issue, we are currently introducing Web Sockets [WS] technology in the SMOB architecture. Web Sockets enable Web applications to maintain bidirectional communications. To establish a Web Socket connection, the Web Socket client sends a handshake request and the server sends a handshake response. Once established the connection, messages can be sent back and forth between the client and the server in full-duplex mode. To say it in another words, instead of the traditional HTTP mechanism where a client send a request and the server responses and close the connection, the client request a Web Socket connection, and it is used to open a socket between the client and the server, in order to the server to communicate later with the client, even if it is behind a firewall or router.

We have already integrated Web Sockets in the PubSubHubBub subscription. In this case, the subscriber client Web Socket sends the subscription request to the hubbub Web Socket server that plays the role of the hubbub and then the hubbub Web Socket server sends the subscription request to the PuSH hub, playing the role of the subscriber. At the moment, we are currently implementing WebSockets for the publication process.

The privacy challenge

Finally, we are currently working on privacy-aware SMOB + PuSH architecture. On the one hand, our aim is to integrated WedID in PuSH, on the other hand, we defines an ontology-based privacy preference system, that will be interpreted in the hubbub so that information is forwarded only to relevant parties.

A Privacy Preference Ontology

Usually, microblogging services does not allow users to restrict some of their post to some persons. Current social networks provide minimum privacy settings such as granting privileges to a previously defined group or individual persons. SMOB would provide the privacy options to the publisher to restrict the broadcast of his/her content to specified set of subscribers.

The publisher uses the lightweight vocabulary called Privacy Preference Ontology [PPO] to store his/her privacy settings. This vocabulary provide the ability (1) to restrict access to resources, statements and named graphs; (2) conditions to refine the restrictions; (3) to specify the users that can access the data depending on whether they satisfy a certain SPARQL query; and (4) and to specify access control privileges granted to users that can access the data.

Listing 2 shows how to create privacy settings for a publisher using PPO. This example restricts a microblog post to users that share an interest similar to the concept used to tag the post.

```
  <http://example.org/privacy/3> a ppo:PrivacyPreference;
    ppo:appliesToResource <http://rdfs.org/sioc/ns#MicroblogPost>;
    ppo:hasCondition [
      ppo:hasProperty tag:Tag;
      ppo:hasLiteral < http://dbpedia.org/resource/Linked_Data>
    ];
   ppo:assignAccess acl:Read;
   ppo:hasAccessSpace [
     ppo:hasAccessQuery "ASK { ?x foaf:topic_interest <http://dbpedia.org
/resource/Linked_Data> }"
  ] .
```

Listing 2: Example SMOB Privacy Preference Filtering

PPO+WebId-enabled PubSubHubBub

First, we can improve security between pub, sub and hub by including an authentication protocol called WebId [WedID] without too much modifications at PuSH. Next, as mentioned in the previous section the privacy of the publishers content can be enhanced by semantic web technologies where the FOAF profiles of the users play a crucial role.

During the user's registration for publishing/subscribing, the authentication is done using the WebID protocol, further providing a secure connection to the user's personal information stored in FOAF format. As it can be inferred from the sequence, we plan to provide to the hubbub with its own WebId URI and certificate, since users have to authenticate the hub to share their profiles. The hub then stores these FOAF profiles into a RDF store which eventually develops as the social graph (FOAF profiles) of the publishers and the subscribers.

Further, the sequence of interactions is similar to the PuSH protocol with some modifications. The publisher once ready to post a message, queries its privacy settings (defined using the previous PPO ontology) to retrieve access space SPARQL queries(one or more) based on the hashtags mentioned in the posts.

The access space SPARQL queries are used by the hub to retrieve the subscribers who should receive the post. These queries are appended to the content of RSS/Atom feed to be sent to the hubbub. The hub is then notified about the feed. Once the hubbub pulls the feed from the publisher, retrieves the queries for the post and selects the subset of the social graph defined by these queries. The post is broadcasted to the subscribers in the subset.

Conclusion

In this paper, we expressed some of our views regarding a Federate Social Web, discussing how a combination of Semantic Web standard and pragmatic Social Web protocols could be beneficial to achieve this vision. We showed that these RDF(S) and SPARQL technologies can be efficiently used in a pragmatic context, and how we currently implement privacy-by-design in SMOB, combining ontologies and Semantic broadcasting. n

References

- [SWXG]. A Standards-based, Open and Privacy-aware Social Web. Editors: Harry Halpin, University of Edinburgh/W3C and Mischa Tuffield, Garlik. W3C Incubator Group Report 6th December 2010. http://www.w3.org/2005/Incubator/socialweb/XGR-socialweb/.
- [SMOB]. Rethinking Microblogging: Open Distributed Semanti. A. Passant, J.G. Breslin, S. Decker. ICWE 2010.
- [PUSH]. PubSubHubbub Core 0.3 -- Working Draft. B. Fitzpatrick, B. Slatkin, M. Atkins. http://pubsubhubbub.googlecode.com/svn/trunk/pubsubhubbub-core-0.3.html.
- [WS]. The WebSockets API. Editor's Draft 10 May 2011. I. Hickson. http://dev.w3.org/html5/websockets/
- [PPO] A Privacy Preference Ontology (PPO) for Linked Data. O. Sacco and A. Passant. In Proceedings of the Linked Data on the Web Workshop, LDOW2011, 2011. http://events.linkeddata.org/ldow2011/papers/ldow2011-paper01-sacco.pdf