FOSP: Towards a Federated Object Sharing Protocol that Unifies Operations on Social Content

Felix Maurer and Sebastian Labitzke

Karlsruhe Institute of Technology (KIT)
Steinbuch Centre for Computing (SCC) & Institute of Telematics
Zirkel 2, 76131 Karlsruhe, Germany
felix.maurer@student.kit.edu, sebastian.labitzke@kit.edu

Abstract: Years ago, the World Wide Web (WWW) began as a system for publishing interlinked hypertext documents. While the protocols on top of which the WWW is built are almost still the same, the usage, as well as the content have changed significantly. Simple delivery of hypertext documents has been expanded by operations, such as uploading, sharing, and commenting on pieces of content. Online Social Networks (OSNs) and other IT services provide aggregated views on these pieces of content. However, the services are often implemented as vendor specific applications on top of common web technologies, such as HTTP, HTML, JavaScript and CSS. Moreover, users are locked into these applications of dedicated providers, which prevents sharing of content across applications and limits the control users have over their data. Most existing approaches that overcome these issues focus on defining a common HTTP API or prefer solutions based on peer-to-peer networks. In this paper, we start by discussing related work and identifying essential requirements for an appropriate solution. Furthermore, we outline the concept and implementation of a Federated Object Sharing Protocol (FOSP), i.e., a different approach to support today's common operations on social content already on a protocol level. We show that services built on top of this protocol can be federated by default, i.e., users registered with different providers can easily interact with each other. Finally, we provide an evaluation and discussion on the proposed approach.

1 Introduction

The World Wide Web started as a system to publish simple, linked documents and the Hypertext Transfer Protocol (HTTP) was created to transfer these documents. In contrast, today's websites are rather complex front ends of powerful applications and documents are dynamically generated depending on parameters of the HTTP request. Furthermore, extensions for sending information to a server were implemented and the hyperlinks initially used to connect documents became methods to invoke actions on servers.

This way, applications were created that allow a user to submit and share pieces of content with others, such as texts, pictures and videos. Nowadays, many types of websites are implemented that provide access to user generated content, e.g., weblogs, microblogging sites, and Online Social Networks (OSNs). However, many features necessary for those

platforms are not supported by HTTP and, therefore, are implemented on top of this protocol. In turn, while the protocols and data formats used are open standards, the applications are not.

As a result, users are unable to share content across platforms and can often access shared data only with a provider-specific client, although the features of the platforms are often very similar. This includes, but is not limited to, authentication and authorization, as well as adding, deleting, rating, and modifying content, or sending notifications. Furthermore, the structure of shared data shows similarities, i.e., content is often hierarchically stored, e.g., comments added to a text, video or photo. These features have to be reimplemented by each provider and the resulting implementations are largely incompatible.

In this paper, we provide the outline of a Federated Object Sharing Protocol (FOSP), i.e., a common protocol for applications that allow users to share pieces of content and operate on this content in the aforementioned manner. On the basis of functionality existing OSNs provide, this protocol allows users to register an account and log-in to a platform, share arbitrary kind of content, grant and deny access on content to certain users or groups of users and to subscribe to streams of content of others. Additionally, to prevent user lock-in with respect to a single provider, it allows federation of all platforms that implement the protocol. The novelty of this approach constitutes the handling of common operations, as well as the federation of platforms already on a protocol level. For reasons of acceptance, the protocol aims to have low complexity regarding implementation and deployment, as well as provide users with easy-to-use and familiar features.

In Section 2, we discuss existing protocols and related work. Section 3 presents common use case scenarios of OSNs and similar applications that are used to compile a list of requirements. Section 4 describes the new protocol-based approach that is evaluated and discussed in Section 5. Section 6 concludes the paper and gives an outlook on future work.

2 Related Work

First, we review several protocols that are widely used on the web today. Second, we examine selected open source projects that aim at supporting users in cross-platform interaction. Third, we explore scientific work that focuses on federated sharing of information.

As mentioned, the **Hypertext Transfer Protocol (HTTP)** forms the protocol basis of today's WWW. Due to its extensibility, HTTP allows the implementation of new request methods, response statuses, headers, and content types. Therefore, this protocol can be used for posting and retrieving almost all kinds of data, as well as for basic authentication. However, the semantics of provided operations is only partially defined. This results in systems that misuse operations, e.g., some applications misuse GET operations to delete a resource even though a DELETE operation exists. Furthermore, as HTTP is state-less, each request usually opens a new connection, i.e., authentication is required for each request and authorization is not handled by default. Additionally, a server cannot notify

clients about updates and while workarounds like long-polling, e.g., BOSH[1], are available, it led to the development of the WebSocket protocol [FM11].

Web-based Distributed Authoring and Versioning (WebDAV) is an extension to HTTP that introduces new methods and semantics for specific resources [Dab09]. It implements missing functionality regarding resource manipulation for supporting authoring of websites. By adding meta-data to the resources, access control lists, information about resource ownerships, and further meta-information of the resources can be stored. Additional methods for manipulating meta-data allow, for instance, to list children of hierarchical structured resources. However, WebDAV is also not designed for pushing updates and to be implemented in a federated network.

SPDY[2] constitutes an extension to the way HTTP requests are transferred over TCP and serves as the working base for **HTTP 2.0** [BTP+13]. It allows multiplexing more than one request over a single connection and the server can even push resources over an existing connection. While it improves the transfer rate, it does not solve the problems of HTTP or WebDAV, like pushing notifications or federating access control.

The **eXtensible Messaging and Presence Protocol (XMPP)** [SA11] was developed as an instant messaging (IM) protocol. XMPP allows the exchange of structured messages between multiple network entities and enables the federation of servers. It is designed to be extensible, i.e., additional functionality can be defined by so called XMPP Extension Protocols (XEP). Methods for sending files, publishing the location of the user, initiating video and voice calls, and many more have been implemented. However, it is not designed for storing data and extensions are possible but lead to defining new protocols.

The **Network File System (NFS)** [SCR+03] is a protocol commonly used for distributed file systems in Unix environments. It allows accessing files over the network but does not support sending notifications about changes. The Glamor research project by IBM aims to provide a federated file system layer on top of NFS [LNT08]. It works by providing a virtual file system based on multiple NFS servers but relies on a central configuration. Therefore, it is not truly a globally federated file system.

Wave [WW11] is a protocol and former Google product for real time collaboration. Users work on XML-like documents called *wavelets* and changes on these documents are synchronized over the network. The protocol allows federation of servers so that users can collaborate on wavelets that are hosted by different providers. However, access control only works at the level of a single wavelet and content is primarily text based.

In addition to existing protocols, the open source community introduced many projects dedicated to free users from platform lock-in and to increase privacy[3]. Next to projects like StatusNet[4] for microblogging and SecureShare[5] for peer-to-peer (P2P)-based encrypted social networking, approaches to implement traditional OSNs over HTTP are invented:

[1] http://www.xmpp.org/extensions/xep-0124.html
[2] http://www.chromium.org/spdy/spdy-whitepaper
[3] https://gitorious.org/social/pages/ProjectComparison
[4] http://status.net
[5] http://secushare.org

Diaspora*[6] defines an HTTP API for federating OSN servers. The content is formatted according to the ActivityStreams specification[7] and transported using various HTTP-based protocols[8]. However, whereas Diaspora* includes sharing of posts, comments and other content, the API is limited to certain types of content[9]. Additionally, instead of retrieving posts that are of interest, i.e., a user-specific or even user-defined preselection of content, the protocol simply broadcasts new posts to users who are connected with the author.

Buddycloud[10] defines an XMPP API and additionally an HTTP API. In the Buddycloud model, posts, pictures, and other files can be submitted to so called "channels" that users can subscribe to. This way, content can be shared with users even if they are registered with other providers. However, the channels restrict the way content can be organized and also restrict the types of content that can be shared. Additionally, access control only happens on the channel level, i.e., access to single posts cannot be restricted. While Buddycloud provides multiple APIs and services for OSNs it does not focus on creating a simple and versatile way of sharing data, setting access rights and sending notifications.

Besides existing protocols and open source projects, a lot of scientific work on decentralized sharing of social content has been published. Approaches range from connecting existing OSNs (cf. [KTS11, OP12]) to building new OSNs, for instance, based on P2P networks (cf. [BH13, BSVD09, BFG+10, NPA10]).

Tramp et al. introduced the **Distributed Semantic Social Network** (DSSN) [TFE+14] that exchanges data represented according the Resource Description Framework (RDF)[11] and uses the vocabulary of the Friend of a Friend (FOAF)[12]. Users can discover each other using WebID[13] and publish new content via the PubSubHubbub protocol[14]. Notifications about updates are distributed with the Semantic Pingback protocol[15] but are not delivered directly to the user. Access control is implemented by using WebID and by adding support for access delegation. In contrast to our approach of an integrated single protocol, the DSSN combines multiple existing protocols and data formats.

The **Distributed Platform for Multimedia Communities** of Graffi et al. is built on top of a P2P network that provides a Distributed Hash Table (DHT) [GPM+08]. The DHT is used to store distributed linked lists that contain and structure the shared content. Several services form the base of their framework: a storage and replication layer, a storage dispatcher for serialization and storage of application specific data, a message dispatcher for direct communication, etc. Different features of OSNs are then implemented as plug-ins on top of these services. To implement access control, they use asymmetric public keys per user and symmetric keys to encrypt content shared with a group of users. However, a general mechanism to receive notifications about new or changed content is missing.

[6] https://diasporafoundation.org
[7] http://activitystrea.ms/
[8] http://www.salmon-protocol.org/, http://tools.ietf.org/html/draft-ietf-appsawg-webfinger-18
[9] http://wiki.diasporafoundation.org/Main_Page
[10] http://buddycloud.com
[11] http://www.w3.org/RDF/.
[12] http://www.foaf-project.org/
[13] http://www.w3.org/wiki/WebID
[14] http://code.google.com/p/pubsubhubbub/
[15] http://aksw.org/Projects/SemanticPingback.html

Project	Architecture	# Components	Protocols
Diaspora*	Federated	1 server	HTTP (Salmon, Webfinger)
Buddycloud	Federated	3 services	HTTP, XMPP
DSSN	Federated	1 and more services	HTTP (WebID, PubSubHubbub, Semantic Pingback)
Distributed Platform for Multimedia Communities	P2P	1 (+ framework plugins)	DHT (PAST[17])
SODESSON	P2P	4 framework components	DHT
Safebook	P2P	1 client + 1 service	DHT

Figure 1: Comparison of OSN projects

Another P2P-based approach constitutes **SODESSON** [BH13] introduced by Baumgart et al. SODESSON abstracts from users' devices to be able to directly address the users and is focused on providing services directly on (mobile) devices while using the social graph as basis for access control decisions. The framework is divided into multiple components, such as the connection manager, the distributed data storage, the contact manager, and the service manager that constitutes the interface to the actual applications. Services that can be provided are divided into three types, namely direct services between online devices, persistent services stored in a distributed storage and hybrid services as a mix of both. Similar to our goals, SODESSON enables distributed sharing of data in OSNs but specializes on providing a transparent access to services on multiple user devices.

Cutillo at al. introduced **Safebook**, i.e., a P2P-based OSN, too [CMS09]. It is focused on user privacy and the utilization of multiple techniques to prevent information leakage on the application layer and on the transport layer as well. In Safebook, each user has a layered network of peers that forward requests, similar to the Tor network[16]. The most inner layer of peers that is directly connected to the user also mirrors profile information and comments, so that they are available even if the user is offline. To prevent eavesdropping, all communication is encrypted. While it provides methods for publishing and retrieving profile information, it lacks the possibility of subscribing and sending notifications and is limited to certain types of content. Furthermore, an additional service is needed for creating a new identity and uses out of band mechanisms to verify it.

In summary, existing protocols satisfy a part of the requirements for federated sharing of social content (cf. Section 3). However, none of them constitutes a "Jack of all trades" that, additionally, can be easily deployed. HTTP/WebDAV and NFS work well for storing data and handling access control, whereas XMPP and Wave support federation and include notifications. Community projects focus on a smaller set of features and prioritize working software. Diaspora* uses multiple data formats and does not include notifications while Buddycould uses multiple protocols. Most scientific work prefer solutions on top of P2P networks and provide security and privacy mechanisms (cf. Fig. 1).

[16]https://www.torproject.org/

3 Use Cases and Requirements

In this section, we start by introducing several use cases to evaluate a new protocol with respect to its applicability in real world scenarios. The use cases are derived from functionality provided by existing OSNs extended by federated sharing scenarios.

Use cases: Independently of the provider a user is registered with, she can share content with users of other providers, as long as they support the API or protocol **(UC1)**. Users can subscribe to new content of other users and will receive notifications about added or changed content **(UC2)**. Users can comment on or "like" content, if allowed by the content author **(UC3)**. Users can publish personal information about themselves, including but not limited to age, sex, location and personal preferences **(UC4)**. Users should not be limited in what kind of content they can share **(UC5)**. Users can restrict access to their shared content on a per-user-basis or on user-defined groups **(UC6)**.

Based on these use cases, we derive requirements that must be met by the approach introduced in this paper to allow a simple solution while still providing the necessary features:

Federation (R1): To prevent platform lock-in, the federation of servers must be possible. Each server must be able to become part of the global network just by implementing the protocol. Therefore, all users must have a globally unique identifier and their content must also have globally unique addresses.

Access control model (R2): As users want to limit access to their content, they must be able to define access rights. Therefore, a discretionary access control (DAC) [SV01] policy should be preferred. Adding role based access control (RBAC) [SV01] is desirable, as users likely sort their peers into groups and then grant or deny access to whole groups.

Authentication and authorization (R3): To enable access control, servers must be able to authenticate and authorize users. Furthermore, as each server can only authenticate users that are registered with itself, it has to act as an identity provider for other servers.

Content manipulation (R4): Users must have multiple methods available to interact with content. This includes adding new, as well as changing and deleting existing content. The type of content that can be shared must not be restricted. Furthermore, the user must be able to identify content she wants to manipulate and, therefore, each content unit must be addressable by a URI.

Subscription (R5): A user must be able to express interest in (a set of) specific content and will be notified if content is added or altered. The information about these subscriptions must be stored as meta-data, such as access control lists.

4 The Federated Object Sharing Protocol (FOSP)

Subsequently to the presentation of related work and requirements regarding federated sharing of social content, we provide another contribution in the following, i.e., we outline the concept and implementation of a new protocol for federated data sharing.

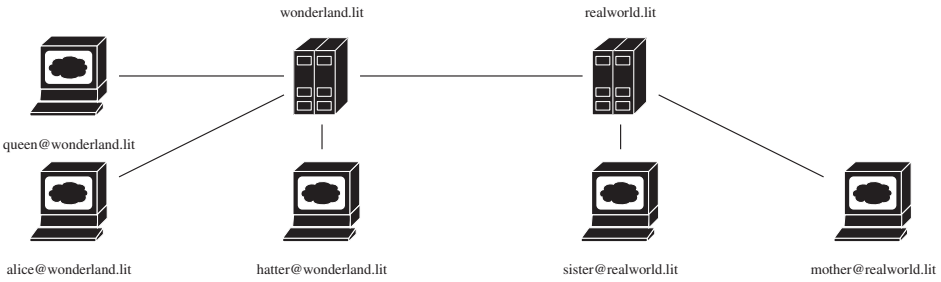

Figure 2: The network architecture of FOSP. User Alice connects to her *home server* wonderland.lit

Network Architecture: In a FOSP network, each agent is either a client or a server. Each server belongs to one provider identified by a domain name. Users connect via a client to the server of their provider that we call the user's *home server*. To retrieve or alter content, the client sends requests. If the *home server* is not responsible for the requested content, it connects to the server that actually stores the content and relays the request (cf. Fig. 2).

Data Structure: We refer to the basic unit of data as an *object* that consists of key-value pairs with specific meanings (cf. "Policies") and adheres to the JSON [Cro06] specification. Furthermore, a file can be attached to an object and all objects are stored as nodes within trees. Each user owns one tree of objects stored on her home server and the root of the tree is named by her globally unique identifier.

Policies: A set of policies define how the content of the objects is to be interpreted and the servers must enforce these policies. Some of the key-value pairs store meta-data, such as the owner of an object, the time it was created, or information about an attached file. An "acl" field of an object stores the associated access control list. A server then determines whether or not a user can perform certain operations on the object. A "subscription" field stores information about users that want to be notified on changes of an object or one of its descendants, which is evaluated by the server responsible for this object. An example for the default key-value pairs of an object is shown in Fig. 3. Furthermore, whether a server can relay a message or accept a relayed message is also subject to policies. For example, a server must only accept a relayed request if the domain name of the remote server matches the domain name in the identifier of the user the request originates from.

Messages: The messages that are sent between the servers and clients are divided into three different types. Clients send *requests* to retrieve or manipulate objects and requests have request types, i.e., SELECT, UPDATE, or CREATE that determine how the request acts upon an object. As an answer to a request, servers send *responses* that also have a type, which is either SUCCEEDED or FAILED. Furthermore, if a request changes objects, the server sends *notifications* to users that have subscribed to these changes and the notification includes an event type, i.e., UPDATED that corresponds to the type of change. All messages carry different information in their first line, depending on their type. Additionally, like in HTTP, each message can have headers that modify its meaning and a body that carries the payload. Messages are serialized as UTF-8 text, except for the body, which consists either also of UTF-8 text or of arbitrary binary data when sending files.

```
{ btime: "2007-03-01T13:00:00Z",
  mtime: "2008-05-11T15:30:00Z",
  owner: "alice@wonderland.lit",
  acl: {
    owner: ["read-data", "write-data", "read-acl", "write-acl",
      "read-subscriptions", "write-subscriptions", "read-children",
      "write-children", "delete-children"],
    users: { alice@wonderland.lit: [ "read-data", "write-data", ... ] }
  },
  subscriptions: {
    users: { alice@wonderland.lit: { events: [ "created", "updated" ], depth: 1 } }
  },
  type: "text/plain",
  data: "Just plain text" }
```

Figure 3: An example object owned by alice@wonderland.lit.

The format of a message looks similar to an HTTP request. Each message is then send as an WebSocket message over a WebSocket connection.

Implementation and performance evaluation: Based on the presented concept, we built a proof-of-concept implementation of a FOSP server and two FOSP clients both written in JavaScript [Mau14]. The server is written for the node.js[18] runtime environment and uses the RethinkDB[19] as database. One of the clients is implemented as a single page JavaScript application[20] and highlights the ease of deployment that can be achieved. The other client is a command line client that can be used in a shell. For performance evaluation, we ran several tests with two instances of the server and up to 800 user clients. While the server already achieves fast response times and can handle many consecutive requests per minute, it is not yet optimized for many parallel users due to the chosen programming language and database. However, FOSP as a protocol can be implemented with almost any language and backing database. We already work on a Go[21] and Postgresql[22] implementation that performs significantly better on concurrent requests, which we plan to make public for download. Furthermore, while scaling the server of one provider might be more difficult, scaling by adding providers should be trivial, similar to XMPP and Buddycloud.

5 Evaluation and Discussion

In Section 3, we defined several use cases and requirements that should be supported by a new protocol for federated sharing of social content. In the following, we show that FOSP meets all of these criteria. The federation of different networks (UC1 and R1) is achieved by relaying messages between servers that, additionally, act as identity providers for their users (R3). Users can share arbitrary content by encoding it as JSON and storing it in an object or by saving it as a file attached to an object (UC5 and R4). They can also subscribe to changes on an object and its descendants and will receive notifications about added or

[18] http://nodejs.org/
[19] http://www.rethinkdb.com/
[20] http://singlepageappbook.com/
[21] http://golang.org/
[22] http://www.postgresql.org/

altered content (UC2 and R5). Comments can be added as new child objects to existing objects (UC3). Personal information can also be saved in an object or multiple objects to allow more granular access control (UC4). Furthermore, an access control list on each object allows users to share content with a group of peers or even a single user. Access control is enforced by the server that stores the object and users can grant or deny rights by setting corresponding entries in the access control list of the object (UC6, R2, and R3).

In this paper, we introduced the first steps towards a new protocol for federated sharing of social content. However, some limitations remain that require further work on this project to reach a protocol that can be globally used. First, server to server authentication must work reliably in order to prevent data leakage in the federated network. This can be done by using DNS but also with more sophisticated methods, such as dial-back strategies or SSL certificate verifications. Furthermore, if a user grants access to content to another user of a different provider, the access is implicitly granted to this provider as well.

However, despite open questions, FOSP could already be used today in specific scenarios in which trust across participating servers is implicitly given but central infrastructures are not deployable. For instance, sharing content between members of universities each of which operates one trustworthy FOSP server would be possible by use of the FOSP version presented within this paper (cf. necessary trust between identity providers of authentication and authorization infrastructures, such as within the DFN-AAI[23]). Furthermore, FOSP constitutes a basis for further discussion and research on a path completely different compared to existing approaches, i.e., a new protocol for provider independent content sharing. Moreover, existing approaches can be combined with FOSP and variations or extensions are imaginable, e.g., just with slight adjustments, FOSP could be used in P2P networks or objects could be extended to allow encrypted content, versioning or locking.

6 Conclusion and Future Work

In this paper, we analyzed related work and identified use cases and requirements for federated sharing of social content in order to propose an approach for implementing today's common operations already on a protocol level. Furthermore, we introduced the concept and implementation of the Federated Object Sharing Protocol (FOSP) that is designed to be simple and deployable while meeting the identified requirements and use cases. FOSP combines storing data, access control and publish-subscribe mechanisms for future content sharing. Furthermore, it enables federation of servers that belong to different providers by default. This way, FOSP can simplify the implementation of traditional social networks and allows competing but compatible server and clients. In the future, we will refine the FOSP specification into a document that covers all implementation relevant details and will add further features, such as content encryption that can remedy remaining security problems. We also plan to publicly provide our ready-to-use FOSP client and FOSP server implementation that can handle a large amount of users for download.

[23] https://www.aai.dfn.de/.

References

[BFG+10] Marin Bertier, Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. The Gossple Anonymous Social Network. In *Middleware 2010*, volume 6452. Springer Berlin Heidelberg, 2010.

[BH13] I. Baumgart and F. Hartmann. User-centric networking powered by SODESSON. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 36(2), May 2013.

[BSVD09] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. PeerSoN: P2P social networking: early experiences and insights. In *Proc. of the 2nd ACM EuroSys Wksp. on Social Network Systems (SNS'09)*, New York, NY, USA, 2009. ACM.

[BTP+13] M. Belshe, Twist, R. Peon, Inc Google, M. Thomson, Microsoft, A. Melinkov, and Isode Ltd. Hypertext Transfer Protocol version 2.0, August 2013.

[CMS09] L.A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12), 2009.

[Cro06] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Informational), July 2006.

[Dab09] C. Daboo. Extended MKCOL for Web Distributed Authoring and Versioning (WebDAV). RFC 5689 (Proposed Standard), September 2009.

[FM11] I. Fette and A. Melnikov. The WebSocket Protocol. RFC 6455 (Proposed Standard), December 2011.

[GPM+08] K. Graffi, S. Podrajanski, P. Mukherjee, A. Kovacevic, and R. Steinmetz. A Distributed Platform for Multimedia Communities. In *Proc. of the 10th IEEE Int'l Symp. on Multimedia (ISM'08)*. IEEE, 2008.

[KTS11] Moonam Ko, H. Touati, and M. Shehab. Enabling Cross-Site Content Sharing between Social Networks. In *Proc. of the 3rd Int'l Conf. on Privacy, Security, Risk and Trust (PASSAT'11) and on Social Computing (SOCIALCOM'11)*. IEEE, 2011.

[LNT08] U. Lanjewar, M. Naik, and R. Tewari. Glamor: An architecture for file system federation. *IBM Journal of Research and Development*, 52(4.5), 2008.

[Mau14] Felix Maurer. FOSP: Federated Object Sharing Protocol. Bachelor's thesis, Karlsruhe Institute of Technology (KIT), Dept. of CS, Inst. of Telematics, January 2014.

[NPA10] R. Narendula, T.G. Papaioannou, and K. Aberer. Privacy-Aware and Highly-Available OSN Profiles. In *Proc. of the 19th IEEE Int'l Wksp. on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE'10)*. IEEE, 2010.

[OP12] Alexandra Olteanu and Guillaume Pierre. Towards robust and scalable peer-to-peer social networks. In *Proc. of the 5th Wksp. on Social Network Systems (SNS'12)*, New York, NY, USA, 2012. ACM.

[SA11] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120 (Proposed Standard), March 2011.

[SCR+03] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck. Network File System (NFS) version 4 Protocol. RFC 3530 (Proposed Standard), April 2003.

[SV01] Pierangela Samarati and SabrinaCapitani Vimercati. Access Control: Policies, Models, and Mechanisms. In *Foundations of Security Analysis and Design*, volume 2171. Springer Berlin Heidelberg, 2001.

[TFE+14] Sebastian Tramp, Philipp Frischmuth, Timofey Ermilov, Saeedeh Shekarpour, and Sören Auer. An Architecture of a Distributed Semantic Social Network. *Semantic Web*, 5(1), 2014.

[WW11] T. Weis and A. Wacker. Federating Websites with the Google Wave Protocol. *Internet Computing, IEEE*, 15(3), 2011.