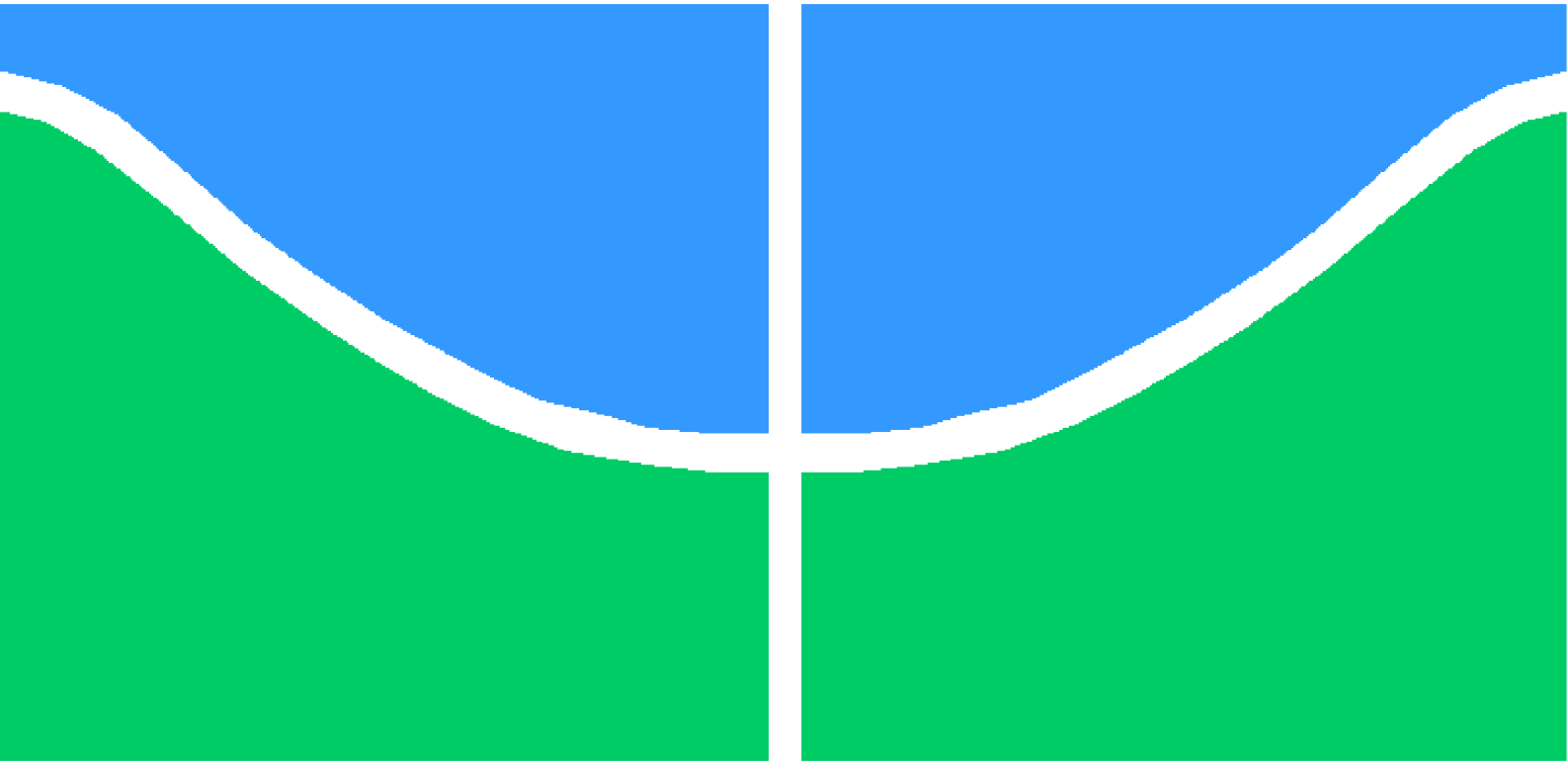

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Software Engineering

Implementation of federation protocol for social networks

Author: Gabriel dos Santos Silva

Supervisor: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

2017

Gabriel dos Santos Silva

Implementation of federation protocol for social networks

Dissertation submitted to the undergraduate course in (Software Engineering) of the University of Brasília, as a partial requirement to obtain the Title of Bachelor in (Software Engineering).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Supervisor: Prof. Dr. Paulo Roberto Miranda Meirelles

Brasília, DF

2017

Gabriel dos Santos Silva
Implementation of federation protocol for social networks/ Gabriel dos Santos Silva. – Brasília, DF, 2017-
58 p. : il. (algumas color.) ; 30 cm.

Supervisor: Prof. Dr. Paulo Roberto Miranda Meirelles

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2017.

1. Federation. 2. Social Networks. I. Prof. Dr. Paulo Roberto Miranda Meirelles. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implementation of federation protocol for social networks

CDU 02:141:005.6

Gabriel dos Santos Silva

Implementation of federation protocol for social networks

Dissertation submitted to the undergraduate course in (Software Engineering) of the University of Brasília, as a partial requirement to obtain the Title of Bachelor in (Software Engineering).

Approved work. Brasília, DF, July 17, 2017:

Prof. Dr. Paulo Roberto Miranda Meirelles
Supervisor

Dr. Tiago Alves da Fonseca
Member 1

Dr. Ewout ter Haar
Member 2

Brasília, DF
2017

Abstract

The federation of social networks aims at integrating users by means of a decentralized structure, enabling the interoperability among multiple social networks in a transparent way. Despite a few isolated initiatives in federating open social networks, there is no adoption of any standard, which hinders the emergence of new, effective federated systems. To understand the difficulties in the development and standardization of federated services, we have conducted research on existing specifications and implementations of interoperability among social networks. We have developed a federation proof of concept within the Noosfero platform, implementing a subset of the Diaspora protocol to federate users and public content, in addition to complementary specifications, such as Salmon and WebFinger. In this work, we introduce our results to federate Noosfero with Diaspora networks, pointing the required steps before further development. We aim to implement the Diaspora protocol within Noosfero, finishing its specification and improving its documentation, encouraging more projects to adopt this protocol.

Key-words: federation. social networks. Noosfero. Diaspora.

List of Figures

List of Tables

List of abbreviations and acronyms

AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
HTML	*Hyper Text Markup Language*
HTTP	*Hypertext Transfer Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
JSON	*Javascript Object Notation*
LRDD	*Link-based Resource Description Discovery*
POP	*Post Office Protocol*
PubSubHub	*PubSubHubbub*
SMTP	*Simple Mail Transfer Protocol*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*

Contents

1 Introduction

Social networks, or social media, can be defined as platforms that allow individuals to connect to others, share personal information, and provide content (BOYD; ELLISON, 2007). The information that flows in these networks is not always public, but usually takes place by means of private infrastructure that belongs to service providers.

In the context of computer security, privacy can be defined as someone's capacity of controlling which information related to them can be consumed and stored, and with whom it can be shared (STALLINGS, 2010). In addition to guaranteeing their users' privacy, social networks must ensure the confidentiality of the information they host, i.e. making sure that users' private information is not exposed to unauthorized individuals.

It can be argued that a central provider having all the control over the flow of private information would put user privacy in jeopardy, for a few reasons. First, the central provider is a single point of failure, and a security breach in its infrastructure exposes every user of the service. Second, users are left with no option other than blindly trusting the service provider to not misuse their personal information. In the case of a commercial entity, that implies trusting the company *but also any other company that may acquire it in the future* to not secretly violate its publicly available privacy policy, and to not revise the privacy policy itself to include new terms that are not favorable to its users.

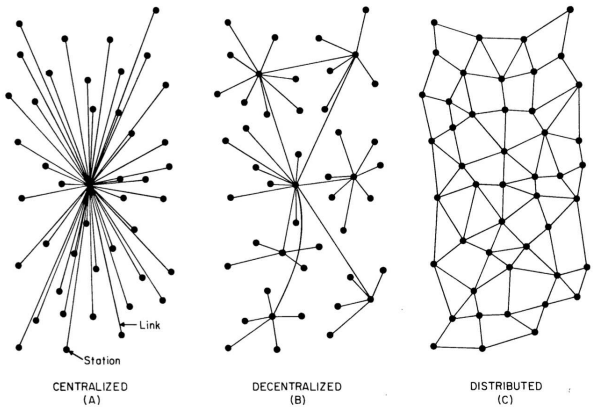

Figure 1 – Conceptual models for communication networks (Source [1]).

The concept of decentralized networks is an alternative to the centralization of private data flow. Decentralization is even one of the original characteristics of the Internet, and a common organization pattern in communication networks. Communication networks can be centralized or distributed (BARAN, 1964). Centralized networks rely on a central node to mediate communication between all the other nodes. Decentralized

networks have multiple mediating nodes, while, in fully distributed networks, nodes can communicate directly in any pattern that is possible and/or necessary. As illustrated by Baran (BARAN, 1964) in Figure 1.

Centralized social networks, even if provided by a decentralized infrastructure with the goal to improve performance and efficiency, conceptually still follow a centralized network model. The central node represents the service provider while the rest represents the millions of users. Adopting a decentralized model means dividing users among connected, intermediary providers that offer the same service, guaranteeing interoperability, so that a user that is served by provider "A" can still interact with a user that is served by provider "B".

A set of interconnected servers that seamlessly provide a service is defined by some authors as a federated network (PEETERS, 2013), which is similar to the definition of decentralized networks shown in (BARAN, 1964), but that is also defined as a set of interoperable implementations that follow a client-server model (BAROCAS et al., 2012). This definition is important because it generalizes the concept of federation to other types of communication systems that are not computer networks and indicates the property of extension independence — any entity that guarantees interoperability can be part of the federation without the need of previous coordination with its existing members.

Service federation contradicts the very existence of a single provider, common in centralized social networks like Facebook and Twitter. Decentralization removes information storage restrictions to a single provider. More importantly, extension independence allows new providers to independently appear in the federation as long as they respect the interoperability criteria, what encourages autonomy. The decentralization of private information distributes the responsibility for maintaining confidentiality, which is no longer dependent on a single entity with arguably concealed intentions.

For a federated network to work, there is a need of interoperability among systems. The probable scenario includes communication among vastly distinct systems, in which protocols and standards are essential for successful communication. This requirement tends to increase the complexity of the applications, as it introduces problems such as data replication and consistency management.

In this work, we study federation in the context of social networks, investigate which aspects are involved in the interoperability of this type of media, and report on the state of the art of the standardization and interoperability. We then present a case study of implementation of federation features in Noosfero, a free software [1] platform for social networking, discussing the design and implementation of this proof of concept.

The remainder of this work is organized as follows. Chapter 2 lists some federation

[1] Free/Libre/Open Source Software (FLOSS)

alternatives that obtained popularity among open social network projects, describing the basic flow and used technologies. Chapter 3 enumerates a number of related works on the development of decentralized social networks and federation infrastructures. Chapter 4 presents the research design and defines our case of study. Chapters 5 and 6 report the implementation of a proof of concept and the results we obtained, and Chapter 7 concludes the paper highlighting its main contributions and pointing paths to future works.

2 Background

The absence of a standard protocol for the federation of social networks hinders the development of interoperable applications, as it leads to the adoption of divergent technologies. This segmentation affects the network effect and does not help on the emergence of a *de facto* standard.

To discuss the standardization of federated systems, it is interesting not only to analyze the community effort to discuss and adopt specifications, but also to identify the reasons that make it difficult to reach a consensus. The subject is incipient in the literature, so we had to resort to reviewing existing community discussion in the relevant mailing lists. Given the nature of the World Wide Web Consortium (W3C) as the standards body of the Internet, we used the archives of the W3C's Federated Social Web Community Group[1].

Taking communication protocols (SILVA, 2017b) as a starting point, in the next subsections we present a documentation of the existing initiatives and attempts to establish protocols for the federation of social networks, describing the used technologies and the state of practice in different projects.

2.1 OStatus

OStatus[2] is a protocol suite to enable real-time interaction between social networks. It was proposed by Evan Prodromou for the implementation of StatusNet, a federated microblogging network that later originated the GNU Social[3] project.

The project obtained visibility in the community, and had its W3C community group[4] created in 2012, one of the few formal attempts to propose a federation protocol. However, the group did not produce any results so far. The OStatus specification received a fair amount of criticism over its limitations, and did not advance into a standard accepted as good enough for all participating projects.

OStatus is defined as a combination of protocols, each covering a part of the interactions that make federation possible. It incorporates standards that already existed when it was proposed, such as Atom/RSS, used for syndicating content from different sources. OStatus proposes to arrange these standards and protocols in a suite for real-time interactions between servers, automating the flow of social interactions among users.

1 <http://w3.org/community/fedsocweb>
2 <http://w3.org/community/ostatus>
3 <http://gnu.io/social>
4 <http://w3.org/community/ostatus/>

The specialized protocols that are included in the OStatus specification are described below.

2.1.1 PubSubHubbub

PubSubHubbub specifies a distributed system for publishing and subscribing (ATKINS B. FITZPAT 2014). It makes use of central hubs used by services to publish new contents and updates, and by other services to subscribe, receiving real-time notifications on new activities. It is an extension to technologies such as Atom/RSS, that depend on users manually fetching for updates.

2.1.2 WebFinger

WebFinger is a protocol used for discovering information about entities based on standard identifiers (JONES G. SALGUEIRO; SMARR, 2013). It aims to solve the problem of sharing identities among servers.

The specification requires every resource to be identified by a URL. WebFinger servers should respond to HTTP requests, providing information in JSON or XML formats. The only information required for other servers to retrieve information about any entity is its resource URL, which can be generated with public information using a LRDD process (HAMMER-LAHAV, 2010).

2.1.3 ActivityStreams

The ActivityStreams specification defines a format to represent activity in social networks, such as “Bob started following Alice”, “Alice posted a new message”, etc. It aims to standardize entity formats, making it easier to share and consume those objects from different servers (ATKINS et al., 2011). The latest specification proposes the use of JSON objects with a set of predefined attributes, including the action type, involved users, and content data.

2.1.4 Salmon

Combining solutions like Atom and PubSubHubbub allow publishing and updating contents across servers in real-time. However, to allow a content to be updated – for example, adding a comment to it – from any of those servers, it is also important to make sure the content state will always be the same in the entire network.

Salmon is a message exchange protocol that provides a way to change and track the state of contents across servers (PANZER, 2009). It proposes a standard process to

securely exchange information among servers, merging all modifications in a stream of messages.

2.1.5 OStatus Current State

The OStatus project was an important step towards the standardization of a federation protocol, what is clear given the creation of a W3C work group. However, the specification was not complete enough to fulfill the requirements of most projects, mainly due to its limitation to public content and the lack of mechanisms to handle privacy.

In 2012, Evan Prodromou announced the development of *pump.io*, another approach for federating social networks, with more modern technologies. It contributed to the weakening of the OStatus community, which is not currently active. The StatusNet project is also no longer active.

Beyond there being no active community to maintain or develop the project, most of the technologies used in OStatus suite evolved. OStatus still specifies the use of technologies considered by many as legacy, such using Atom (instead of JSON) for ActivityStreams messages.

Despite using technologies said to be outdated, OStatus is still used on the development of federated social networks, such as GNU Social[5], which describes itself as a continuation of StatusNet, and the Mastodon project[6], which promises compatibility with GNU Social.

2.2 Diaspora

The Diaspora project[7] proposed the implementation of decentralized social networks in response to concerns raised on privacy and freedom in centralized social platforms. Its first version was released on September, 2010 as a result of a crowdfunding campaign. By August, 2012, a community of users started to maintain it.

Diaspora's main selling point is avoiding content centralization, with a lack of central control over user data, by building a network of personal servers, called pods. Each Diaspora pod stores only the data of its own users, and allows them to interact with users on other servers through a federated network.

The Diaspora federation protocol was created to converge with OStatus as soon as the latter supported private contents, what did not happen so far. Most of the protocol specification relies on the concept of remote users, and on an exchange mechanism.

[5] <http://gnu.org/s/social/>
[6] <https://github.com/Gargron/mastodon>
[7] <http://diasporafoundation.org>

2.2.1 Remote Users

A fundamental concept to implement federated networks using the Diaspora protocol is taking into consideration the existence of remote users. Most applications only recognize local users, that is, those who have their credentials and profiles saved in the local database. However, to enable interaction with users from other networks, at some point it is necessary to handle users that do not exist locally.

Diaspora classifies its users in two categories: local and remote. While local users follow the classic implementation, remote users only interact with the application through federation mechanisms. Having two category of users may affect the project architecture, and this concern should ideally be taken into consideration on early design stages.

2.2.2 Message Exchange

In the Diaspora protocol, messages are exchanged using a subset of the Salmon protocol. Essentially, the Diaspora specification describes how the message should be built, encrypted, and sent to the Salmon endpoint of the destination pod.

The payload of a Salmon message is an object that represents an activity in a pod. Those objects are called entities and can represent content publications, private messages, comments, retractions, or even notifications of follow and unfollow actions.

Every message is sent using HTTP to certain endpoints in the destination server. The endpoint depends on the nature of the entity. For example, private messages are sent to user-specific salmon endpoints, while public contents are sent to public endpoints.

2.2.3 Protocol Flow

Besides using a subset of the Salmon protocol to exchange messages, Diaspora also uses WebFinger and hCard standards to discover identities and fetch public profiles from remote servers. It is also possible to use ActivityStreams and PubSubHubbub to provide public feeds.

The basic flow is to discover users, fetch their public information, then create and send Salmon messages according to the desired interaction. Diaspora pods will also send Salmon messages to subscribed servers, notifying about publications and updates.

2.2.4 Diaspora Current State

Both the project and the protocol specification remain in active development. The latest specification dates from July, 2016. There is also a protocol implementation written as a Ruby library, used in the main Diaspora project, that includes most of the message

exchange mechanism. That library can be reused by other projects that want to implement federation with the Diaspora network.

The visibility gained by the Diaspora project has provided it with a significant following, including adoption of its specifications by other projects such as Friendica[8] and Hubzilla[9], which support some level of integration with Diaspora pods. That visibility was not enough, though, to trigger a formal standardization process.

[8] <http://friendi.ca/>
[9] <https://project.hubzilla.org>

3 Related Work

Decentralized social networks are well established in early works. Au Yeung et al. (YEUNG et al., 2009) describes decentralized models for social networks and proposes the use of technologies such as linked data. Chao et al. (CHAO; GUO; ZHOU, 2012) further explores this method describing seamless interaction and single point of access to the network, important concepts for federated systems.

In addition to reference models, several works propose implementations for decentralized infrastructures. Buchegger et al. (BUCHEGGER et al., 2009) propose an infrastructure for P2P networks, explore issues related to decentralized social networks, such as exchanging messages over a decentralized architecture and encryption. They also provide insights on asynchronous message exchange and the use of a centralized storage mechanism to avoid content replication.

There are other works that propose implementations of decentralized networks. Boelmann et al. (SCHWITTMANN et al., 2013) explore privacy and security issues. Aberer et al. (NARENDULA; PAPAIOANNOU; ABERER, 2012) work on content replication, exploring the impacts of availability and performance on the propagation of messages.

Bielenberg et al. (BIELENBERG et al., 2012) explore the growth of Diaspora networks, but more importantly, they give an overview of privacy in the given platform. The authors show that, at the time, users preferred to join popular and reliable servers, indicating that they do not host their own data, even though this is the proposal of decentralized networks.

Finally, Fan et al. (HU; FAN; LAU, 2014) propose a different approach to develop a decentralized social network, presenting a proof of concept with a large network. The proposed approach is to, instead of building a decentralized network, decentralize ordinary social networks by providing a layer to relay contents from one network to another.

Even though there are some implementation proposals, we choose to explore the approach of the Diaspora project. The discussion regarding the implementation of federation protocols in existing projects is still incipient, and constitutes a motivation to proceed with the proof of concept we propose.

Our goal is not to present another federation infrastructure, but to investigate the implementation of existent protocols in projects that were not designed to be federated, and differently from the related works, to report our experiences implementing the said proof of concept.

4 Research Design and Strategies

Given the non-existence of standards for federated social networks, we define the following research questions to guide this work.

RQ1: *What are the protocols in use for federation of social networks and what is the status of their development and adoption?* During the development of this research, we explored the protocols proposed and being used to approach the implementation of federated social networks, reporting the current status of the alternatives and projects that use it.

RQ2: *What engineering problems need to be solved in order to add support for federation in a platform that was not conceived with federation as a requirement?* There is virtually no literature on the implementation of federation support on existent projects. Most of the previous work describe the design and implementation of federated systems, without assuming a previous, non-federated system. We wanted to investigate which aspects of a project can affect the implementation of federation support.

4.1 Design Alternatives

There are currently two possible strategies for implementing federation. Some of the discussion on this topic is recorded in W3C mailing lists[1] and we have organized them to be presented in this section.

We could have the development of a system or protocol that should be supported by all applications interested in joining the federation, a method characterized by a sole entity acting as the common denominator among all networks, or alternatively by a massive coordination effort among loosely-coupled, independent projects working on the topic.

The alternative is every application explicitly implementing the protocol of every other application it wants to connect with. This strategy does not depend on a standard protocol, or on a central authority to define what needs to be implemented and how. It can be called the "polyglot strategy".

The polyglot strategy brings some restrictions to the federation of systems, since it depends on every application responding to the protocol of every other application in the network. This strategy seems to support the segmentation of specifications as opposed

[1] In the following thread from the federated social networks group <http://lists.w3.org/Archives/Public/public-fedsocweb/2013May/0058.html>

to what would happen when using a single protocol as the common denominator. On the other hand, the adoption of a common denominator depends on a protocol capable of covering the peculiarities of all possible networks, including types of relationships among users, content sharing mechanisms, and privacy policies.

The criticism over the lack of a notion of privacy in OStatus[2] is an example of the barriers caused by these differences. The project only supports public content, so networks with more complex privacy definitions are not completely covered. In the meantime, other projects were created to cover these flaws, such as Diaspora and Friendica.

A specification that meets the requirements of all possible social networks would require a very large design and development effort. An alternative would be a protocol that establishes a subset of policies as a base to the implementation of more complex or specific features. It can be argued that, eventually, these policies would have to support incompatible concepts, given the difficulty of finding ways to satisfy all systems.

Even if such an agreement could be found, every application would have to solve its own needs based on the restrictions of the common denominator. It would possibly require projects to be heavily modified to adopt such a protocol, which does not contribute to the viability of that strategy.

The tendency to adopt a polyglot strategy indicates that the difficulty in finding a universal common denominator surpasses the interoperability benefits. On the other hand, if projects keep implementing integration with individual projects we could eventually find some kind of convergence, leading to a network effect, and identifying a common denominator for a subset of applications.

For now, federation can only be achieved by a polyglot strategy, adopting the specifications of individual projects, which are usually a suite of well established protocols. A good start is to choose a project considering its community activity and the adherence of its standard with your particular needs.

4.2 Case study: Noosfero

Noosfero[3] is our free software platform that can be used to build social and collaboration networks, providing a platform with blogs, CMS, and feeds. It is written in Ruby, using the Rails framework, is licensed under the Affero GNU General Public License version 3, and has an active development community.

The need for supporting federation is a long-standing issue for the Noosfero project (SILVA, 2017a). Started in 2007, when the ideas around federation were still incipient,

[2] The discussion regarding OStatus privacy can also be found in W3C mailing lists, <http://lists.w3.org/Archives/Public/public-fedsocweb/2013May/0061.html>.

[3] <http://noosfero.org>

Noosfero ended up evolving a large set of features required by the different organizations that use it before a proper plan for adding federation could be laid out.

We decided to spearhead the development of federation support in Noosfero, with the main goal of allowing Noosfero sites to integrate with other social network providers, whether they also use Noosfero, or not. Our proposal was to use the Diaspora protocol.

Even though the objective is to eventually support the entire protocol, the first step was producing a proof of concept that implements a subset of it. This would help identifying the required modifications in the Noosfero code, and designing the rest of the federation infrastructure. It will provide an initial level of federation that can already be used by end users.

To define the scope of this proof of concept we took into consideration roadmaps that were result of previous discussions in the community. We derived the following requirements from the features we believe will help the most to build the federation infrastructure, mainly the users discovery and message exchange mechanisms.

1. Remote users must be located via the Noosfero people search. The implementation must follow the discovery standard used by Diaspora, based on WebFinger. It must also be possible to find Noosfero users from any other Noosfero or Diaspora site.

2. Users from a Noosfero and a Diaspora site must be able to follow each other. Both sides must be aware of the relationship.

3. The Noosfero server should receive and handle publications sent by Diaspora servers, making it possible to consume contents from remote servers. Sending these messages from Noosfero will make it possible for local users to be followed, what is also the scope of the proof of concept.

4. Noosfero sites should also send publications and comments to any other server that hosts a user that follows the activity of local profiles. This adds symmetry to the previous feature, providing reciprocal interaction between the servers.

5 Implementation

The first step to implement interoperability through the Diaspora protocol is to offer a mechanism to discover and provide users and public profiles, in this case using WebFinger and hCard.

The second step is to implement a message exchange mechanism to provide the means to talk to third-party servers. The Diaspora protocol proposes the exchange of messages containing entities that represent contents and interactions. For now, Noosfero should handle the following entities:

- Profiles creation, update, and removal
- Publications and comments creation and removal
- Content subscriptions and unsubscriptions
- User relationships (follow and unfollow)

The following sections also describe technical specifications of the Diaspora protocol, and could be used as reference for future implementations.

5.1 User Discovery

The Diaspora protocol proposes the use of WebFinger to fetch user identities and hCard to share public profiles. The implementation used in the Diaspora project still uses XML to format WebFinger payloads, what is considered legacy.

The discovery process is described in Figure 2, where the remote server is queried with an identifier in the format *user@host*. The discovery endpoint can be found in the server metadata, also obtained via WebFinger, on a standard endpoint. After the identity lookup, the remote server is queried again for the public profile, which in turn is obtained via hCard in HTML format.

As the discovery implementation is based on WebFinger, it is possible to find users on any application that responds to this format.

5.2 Message Exchange

To follow an external user, it is necessary to send a private Salmon message to its endpoint. Receiving the message, the Diaspora server creates a local profile to represent

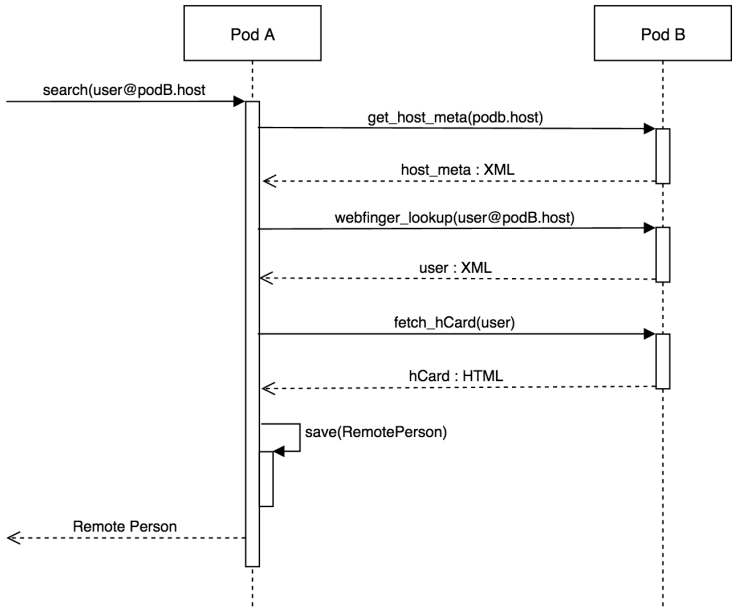

Figure 2 – Sequence diagram of the user discovery process.

the remote user locally — only after querying the Noosfero server for its server metadata and the user's public profile. This process is shown in Figure 3, and includes part of the discovery process.

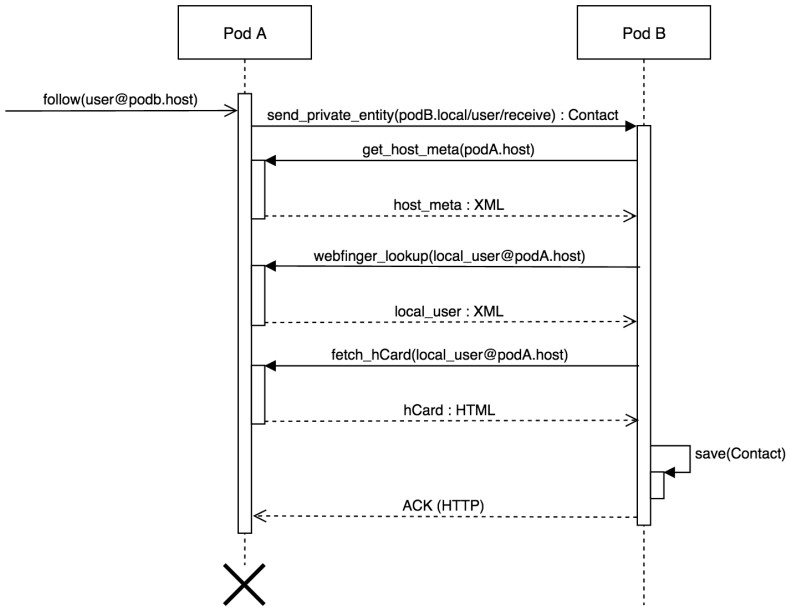

Figure 3 – Sequence diagram of the contacts sharing process.

It is crucial to make sure that the message was successfully delivered, otherwise the servers will not share the same state. The server sending the message should offer some kind of reliability, retrying or even undoing actions when facing communication problems. When trying to follow a remote user, Noosfero will retry a predefined number

of times, and then destroy the relationship locally if the remote server did not respond with a success status.

Private salmon messages are encrypted with RSA to ensure confidentiality, which requires every user to have a pair of RSA keys. The proposed implementation for Noosfero uses the OpenSSL Ruby bindings to generate a key pair for every user, as soon as that is necessary.

Security is an important aspect that still have to be considered more carefully. We are aware that the server should not hold keys in behalf of users, but for now the keys are serialized and stored in the database. To avoid storing plain text values, the private keys are encrypted using a symmetric key and the *Advanced Encryption Standard*. Ideally, we should be able to find a solution where each user holds its own keys.

The last step is to handle incoming entities representing user contents. As shown in Figure 4, new contents are sent by Diaspora to every server that is involved in the interaction — in this case, the origin of users following the author of the new content. At the time of writing, Noosfero supports only public content.

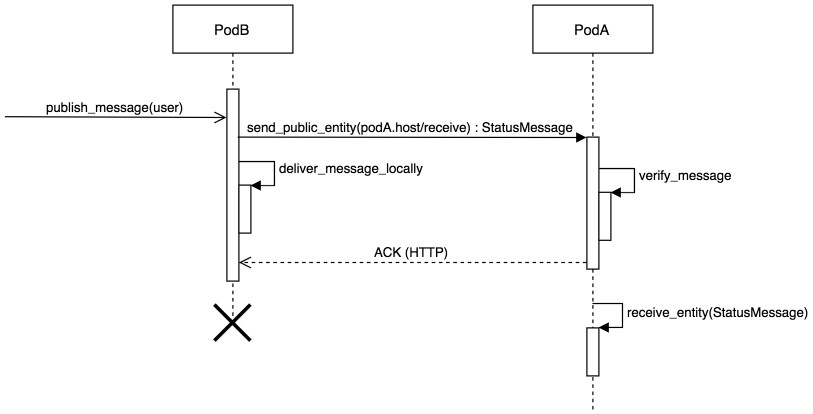

Figure 4 – Sequence diagram of the publication sending process.

After receiving the publication notification in a public Salmon message, Noosfero saves the data locally. It is also necessary to include a GUID (Globally Unique IDentifier), required as identifier for all entities sent across servers.

It is also important to handle retraction entities to maintain the same state in all servers. These entities are sent every time a content or user profile is removed, and have the same visibility as the original content. Handling a retraction involves querying for a related local entity that matches the one that has been removed remotely, so that the removal can be replicated locally.

6 Results

After the implementation described in the last chapter, it was possible to achieve initial federation functionality that showcases the possibilities of federation using the Diaspora protocol. This proof of concept counts with the following features.

Remote users can be found using Noosfero search using the user identifier and the remote pod host name (Figure 5). Remote profiles are created to store and display the public info in the local server.

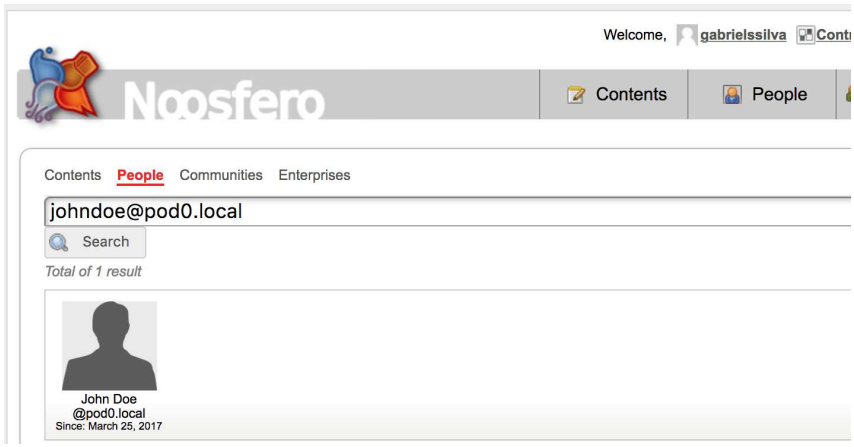

Figure 5 – Diaspora user displayed in Noosfero search results.

Noosfero users can also be found on the Diaspora side, since Noosfero servers now respond to WebFinger and hCard requests (Figure 6).

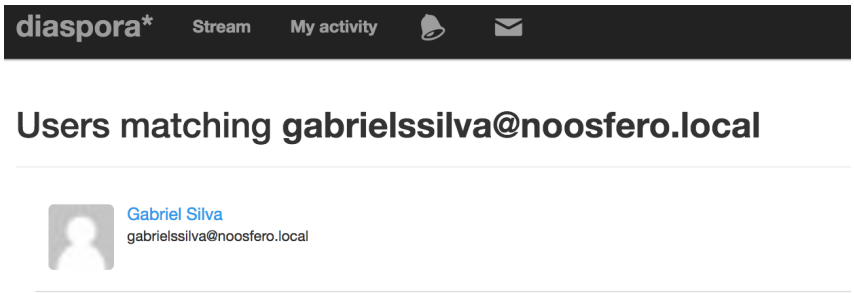

Figure 6 – Noosfero users displayed in Diaspora search results.

Remote users can be followed by users of a Noosfero server. When following a remote user, Noosfero notifies the remote server, which will handle the message by creating the relationship and notifying the related users. Figure 7 shows a notification displayed for a Diaspora user after he was discovered and followed by a remote Noosfero server.

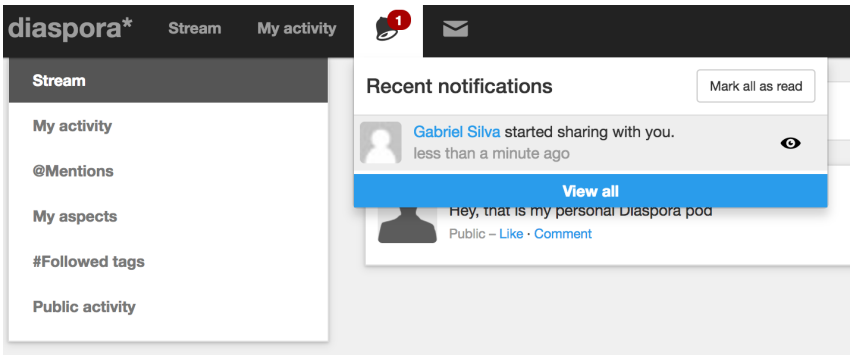

Figure 7 – Notification of remote activity in Diaspora.

Any content created by Diaspora users being followed by someone in the local server will be sent to Noosfero, that will handle them by creating the content locally. The publication is displayed in the remote user local profile, but notifications can also be sent to all related users (Figure 8).

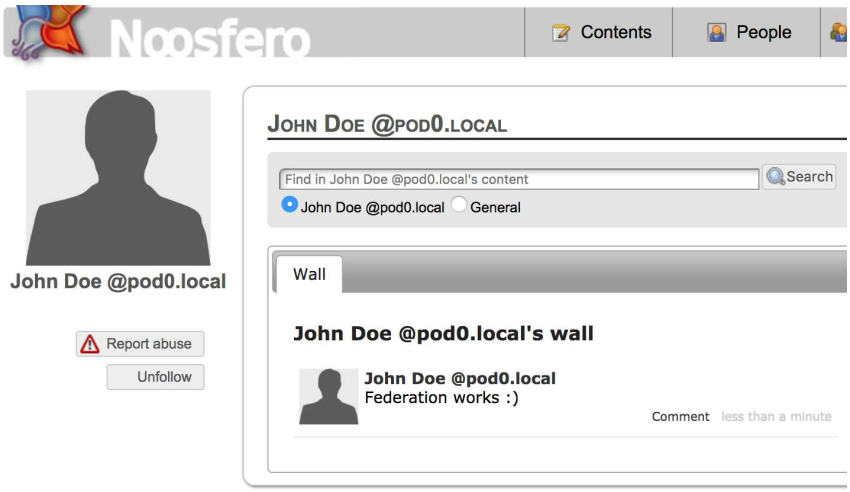

Figure 8 – Content created by Diaspora users displayed in Noosfero.

As users are able to reply existing publications, comments must also be sent to remote servers. *Noosfero will create comments locally, and send comments to other servers* when remote users are subscribed (Figure 9). Publications and comments are both important for a first set of federation features.

The information sent by other servers is usually saved to the local database, so there will be an amount of data replication, requiring further attention to maintain coherence across the network. Users can delete publications or comments, edit their personal information, unfollow other users and even destroy their own account. These tractions are also represented as entities that are handled by Noosfero, which also sends them accordingly.

These features will work between Noosfero and any application that supports the

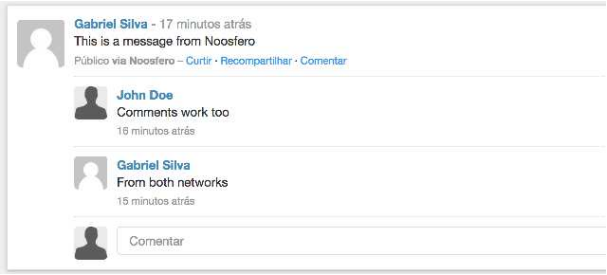

Figure 9 – Comments from Noosfero sites are visible in Diaspora pods.

Diaspora protocol, not only Diaspora itself. User discovery will work with any server that supports the WebFinger and hCard standards as well.

7 Conclusions

Our report on the existing federation initiatives helped to identify *what are the protocols in use for federation of social networks and what is the status of their development and adoption*. We described several standards that are used to build decentralized social networks, where we can highlight the Diaspora protocol, and OStatus and derived projects, such as Mastodon. While evaluating the adoption of these technologies, we could verify the lack of standardization, identifying a scenario where different projects choose to follow distinct strategies to better suit individual needs.

The Diaspora protocol provides a reliable process for finding users and exchanging messages among servers. By implementing it in Noosfero, we were able to federate an instance with both Noosfero and Diaspora servers, showing that it is feasible to implement federation in an existing application.

It is important to note that the Diaspora protocol suits only a subset of the existing projects, from what follows that it is most probably not a "silver bullet" for building a larger federation.

Even though we can point projects that successfully implemented a decentralized network, such as Hubzilla [1], which also supports the Diaspora protocol, these are usually designed for a decentralized context and aim to provide federation out of the box. An additional challenge lies with projects that were not designed as such since the beginning but still want to support federation.

Our proof of concept helps to answer *what engineering problems need to be solved in order to add support for federation in a platform that was not conceived with federation as a requirement?* To start with, our implementation would require more effort if Noosfero features were not already somewhat compatible with the specification. For example, Noosfero already supported asymmetric relationships ("follows") besides symmetric ones ("friends"). If that was not the case, the implementation would turn harder than it was.

There are a couple points that we want to list regarding the implementation of the Diaspora protocol in existing social networks. These are either challenges that can also be faced or aspects that need attention when adding federation support in projects that were not designed to be federated. These findings also offer some insight on how the Diaspora protocol can be used, and we take the opportunity to point some limitations.

[1] <https://github.com/redmatrix/hubzilla>

7.1 Lessons learned on supporting a federation protocol

Beforehand, it is important to take into account that your data model will have to be modified to contemplate Diaspora entities. For instance, all exchanged entities must have a global identifier (GUID), and each user requires a pair of RSA keys to properly send and read private Salmon messages.

The protocol may also affect the way the project handles users and contents, mostly because all remote data must be stored in the local database, what includes remote profiles and contents. It most likely means that the application must foresee the existence of remote users and provide some level of interaction, like publishing contents and sharing with the rest of the network.

There are also some points that can help to exchange entities among servers. When sending or receiving these entities, besides the message format, it is important to identify the endpoints to be used for the HTTP requests. Remote servers will use analogue endpoints to send its own entities when establishing the communication, so it will be easier to start by choosing a small set of entities and sending them from a remote server, implementing the local receiver endpoints as needed. Contacts and status messages are a good choice of entities to begin with.

Sending and Handling entities should be done in background, since it involves network usage and opening and creating Salmon envelopes, which can be computationally expensive. Whereas part of the work is to build and exchange Salmon messages, one should also consider using some tool that implements it, such as the diaspora federation Ruby library [2].

The reliability of the message sending must be assured by each server. It may be interesting to keep track of the statuses of each pod, for example storing timestamps and number of trials of failed requests. Pods do not inform the network of their status, so everything is still made based on the HTTP responses status. Considering that the failure to deliver messages affects the coherence of the information in the network, the protocol should be reviewed to handle scenario formally.

It is also important to point that a server may implement several federation protocols, i.e. servers A and C support two different federation protocols, and server B wants to join both networks. This use case will affect the implementation, since public interactions from users in B with A or C may be potentially replicated.

Consider that a user from B interacts with both A and C servers, the state of public contents will need to be the same across all networks. In this case, A and C do not acknowledge each other, so the responsibility to relay the modifications lies with B.

[2] <https://github.com/diaspora/diaspora_federation>

Also because A and C are isolated, the *relayability* restriction is not described in any of the specifications, and it is up to the local server to decide whether or not the messages should be relayed.

The relaying problem is also present when all servers support the same protocol, and it is somewhat covered by Diaspora's specification. The idea is that relaying messages to pods that are in some way related to the interaction, it will be easier to find references to new pods, helping the servers to establish new connections. Since every federated feature depends on a reference to the remote pod, it is important to provide the means to broaden the network.

7.2 Limitations and future work

Regarding the support of the Diaspora protocol, contents with limited visibility were not in the scope of this proof of concept, and it is an important step to extend the federation features turning the network privacy aware. Although we already handle private messages, we need to match Noosfero circles with Diaspora aspects, making sure the messages will only be visible for the appropriate users.

For now, only plain text messages are being shared across servers. Since Noosfero is also a CMS, it is our interest to federate rich contents, such as articles with embedded HTML and images. The protocol currently offers the means to share static images, so it will probably be the case to extend its custom entities.

It is also important to support direct messages, events and post reactions (such as a "Like"), showing some convergence with the Diaspora interaction model, and increasing the use cases of the federated network. Besides supporting the current state of the protocol, we must attend to the security issues stated previously, possibly following good practices such as using different pairs of keys to sign and encrypt the exchanged messages.

Whereas we were able to implement the protocol and join a network of Diaspora pods, it is clear the specification has to evolve in the ways it deals with content replication and reliability of the messages delivery. A standard approach to exchange messages between servers, such as a mechanism built over HTTP, would address those concerns and be a great contribution to the Diaspora protocol.

Bibliography

ATKINS B. FITZPATRICK, J. G. M.; SLATKIN, B. *PubSubHubbub Core 0.4 – Working Draft*. [S.l.], 2014. Disponível em: <http://pubsubhubbub.github.io/PubSubHubbub/pubsubhubbub-core-0.4.html>. Citado na página 20.

ATKINS, M. et al. *Atom Activity Streams 1.0*. [S.l.], 2011. Disponível em: <http://activitystrea.ms/specs/atom/1.0/>. Citado na página 20.

BARAN, P. On distributed communications networks. *IEEE Transactions of the Professional Technical Group on Communications Systems*, January 1964. Citado 2 vezes nas páginas 15 and 16.

BAROCAS, S. et al. A critical look at decentralized personal data architectures. *CoRR*, abs/1202.4503, 2012. Disponível em: <http://arxiv.org/abs/1202.4503>. Citado 2 vezes nas páginas 16 and 50.

BIELENBERG, A. et al. The growth of diaspora - a decentralized online social network in the wild. In: *INFOCOM Workshops*. IEEE, 2012. p. 13–18. ISBN 978-1-4673-1016-1. Disponível em: <http://dblp.uni-trier.de/db/conf/infocom/infocom2012w.html>. Citado na página 25.

BOYD, D. m.; ELLISON, N. B. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, Blackwell Publishing Inc, v. 13, n. 1, p. 210–230, 2007. ISSN 1083-6101. Disponível em: <http://dx.doi.org/10.1111/j.1083-6101.2007.00393.x>. Citado na página 15.

BUCHEGGER, S. et al. Peerson: P2p social networking: Early experiences and insights. In: *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*. New York, NY, USA: ACM, 2009. (SNS '09), p. 46–52. ISBN 978-1-60558-463-8. Disponível em: <http://doi.acm.org/10.1145/1578002.1578010>. Citado na página 25.

CHAO, W.; GUO, Y.; ZHOU, B. Social networking federation: A position paper. In: *Computers and Electrical Engineering*. [S.l.: s.n.], 2012. v. 38, p. 306–329. Citado na página 25.

COMER, D. E. *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architectures, Fourth Edition*. 4th. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. ISBN 0130183806. Citado 2 vezes nas páginas 47 and 48.

HAMMER-LAHAV, E. *LRDD: Link-based Resource Descriptor Discovery*. [S.l.], 2010. 1-16 p. Disponível em: <https://tools.ietf.org/id/draft-hammer-discovery-06.txt>. Citado na página 20.

HU, P.; FAN, Q.; LAU, W. C. SNSAPI: A cross-platform middleware for rapid deployment of decentralized social networks. *CoRR*, abs/1403.4482, 2014. Disponível em: <http://arxiv.org/abs/1403.4482>. Citado na página 25.

JONES G. SALGUEIRO, M. J. P.; SMARR, J. *WebFinger*. [S.l.], 2013. 1-28 p. Disponível em: <https://tools.ietf.org/rfc/rfc7033.txt>. Citado na página 20.

KLENSIN, J. *Simple Mail Transfer Protocol*. [S.l.], 2001. 1-79 p. Disponível em: <https://tools.ietf.org/rfc/rfc1280.txt>. Citado na página 50.

KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th. ed. [S.l.]: Pearson, 2012. ISBN 0132856204, 9780132856201. Citado 6 vezes nas páginas 7, 47, 48, 49, 50, and 51.

LIEBOWITZ, S. J.; MARGOLIS, S. E. *Network Externalities (Effects)*. 1998. Disponível em: <https://web.archive.org/web/20160410041613/http://www.utdallas.edu/~liebowit/palgrave/network.html>. Citado na página 49.

NARENDULA, R.; PAPAIOANNOU, T. G.; ABERER, K. A decentralized online social network with efficient user-driven replication. In: *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. [S.l.: s.n.], 2012. p. 166–175. Citado na página 25.

PANZER, J. *Salmon Protocol Summary*. 2009. <http://www.salmon-protocol.org/salmon-protocol-summary>. Accessed: 2016-07-15. Citado na página 20.

PEETERS, S. *Beyond distributed and decentralized: what is a federated network?* 2013. <http://networkcultures.org/unlikeus/resources/articles/what-is-a-federated-network/>. Accessed: 2016-07-13. Citado na página 16.

POSTEL, J. *IAB Official Protocol Standards*. [S.l.], 1992. 1-32 p. Disponível em: <https://tools.ietf.org/rfc/rfc1280.txt>. Citado na página 49.

SCHWITTMANN, L. et al. Sonet – privacy and replication in federated online social networks. In: *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. [S.l.: s.n.], 2013. p. 51–57. ISSN 1545-0678. Citado na página 25.

SILVA, G. dos S. *Federação no Noosfero*. 2017. <https://pt.wikiversity.org/wiki/Federa%C3%A7%C3%A3o_no_Noosfero>. Accessed: 2017-07-13. Citado na página 28.

SILVA, G. dos S. *Padronização de Protocolos de Comunicação*. 2017. <https://pt.wikiversity.org/wiki/Padroniza%C3%A7%C3%A3o_de_protocolos_de_comunica%C3%A7%C3%A3o>. Accessed: 2017-07-13. Citado na página 19.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0136097049, 9780136097044. Citado na página 15.

TANENBAUM, A. S.; WETHERALL, D. J. *Computer Networks*. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0132126958, 9780132126953. Citado 3 vezes nas páginas 47, 49, and 50.

YEUNG, C. man A. et al. Decentralization: The future of online social networking. In: *In W3C Workshop on the Future of Social Networking Position Papers*. [S.l.: s.n.], 2009. Citado na página 25.

Appendix

APPENDIX A – PROTOCOLOS DE COMUNICAÇÃO

A comunicação envolve a troca de uma série de mensagens entre duas entidades. Parte fundamental para a comunicação bem sucedida é a premissa a respeito das informações transmitidas e recebidas, como por exemplo a linguagem e o meio de transmissão (COMER, 2000). Caso as entidades envolvidas na comunicação não concordem em relação a estas premissas, não será possível estabelecer um diálogo adequada.

Restrições a respeito do formato, meios de transmissão, e ações a serem tomadas no envio e recebimento de mensagens são definidas por meio de protocolos. A partir do momento em que duas entidades sigam o mesmo protocolo, pode-se garantir que a comunicação será estabelecida. Assim como qualquer outra entidade, componentes de *hardware* e *software* também estão sujeitos a protocolos de comunicação (KUROSE; ROSS, 2012).

Antes de abordar os protocolos de federação, é necessário entender a necessidade de protocolos de comunicação a partir da utilização em redes de computadores. Também se faz necessário discutir o processo de padronização desses protocolos, o que deve ajudar na discussão a respeito da definição e utilização de padrões na construção de sistemas federados.

A.1 SUÍTES DE PROTOCOLOS

No contexto da computação, redes de comunicação são construídas com diferentes tecnologias de acordo com necessidades e restrições específicas, o que prejudica a capacidade de intercomunicação entre dispositivos (COMER, 2000). A Internet, por exemplo, é uma coleção de redes menores que eventualmente utilizam tecnologias diferentes, como é o caso de linhas telefônicas, e transmissão a rádio (TANENBAUM; WETHERALL, 2010). Um desafio diferente é alcançar a intercomunicação em um sistema complexo como tal, onde as redes que o compõem utilizam seus próprios protocolos, desta vez específicos ao meio de transmissão.

De acordo com (COMER, 2000), existem duas observações fundamentais ao projeto de redes de comunicação:

1. Não existe nenhuma tecnologia de rede capaz de satisfazer às restrições de todos os possíveis contextos;

2. Usuários desejam intercomunicação universal.

A primeira observação sugere que a necessidade de interoperabilidade eventualmente pode surgir entre sistemas incompatíveis. Ainda assim, a discrepância deve ser invisível ao usuário, que de qualquer forma espera a interoperabilidade, ponto reafirmado pela segunda observação.

Esse tipo de interoperabilidade pode ser implementada tanto no nível das aplicações quanto no nível da rede. Enquanto a primeira estratégia supõe que as aplicações prevejam explicitamente suporte a cada uma das tecnologias, a segunda estratégia é mapeada desde o *hardware*, providenciando uma camada de abstração para os componentes utilitários.

A intercomunicação no nível de rede (ou simplesmente internet (COMER, 2000)) também pode empregar um modelo de camadas. Neste caso, os protocolos são organizados em uma hierarquia vertical, de acordo com seus objetivos. A interface entre cada camada é bem definida, o que concede modularidade ao sistema (KUROSE; ROSS, 2012).

Neste tipo de hierarquia as mensagens são enviadas sucessivamente da camada mais superior até a camada mais inferior durante a transmissão, o inverso do que acontece no recebimento. Apenas as camadas mais inferiores do transmissor e destinatário são conectadas por meio de um meio de transmissão arbitrário, o que garante abstração do meio para as camadas superiores.

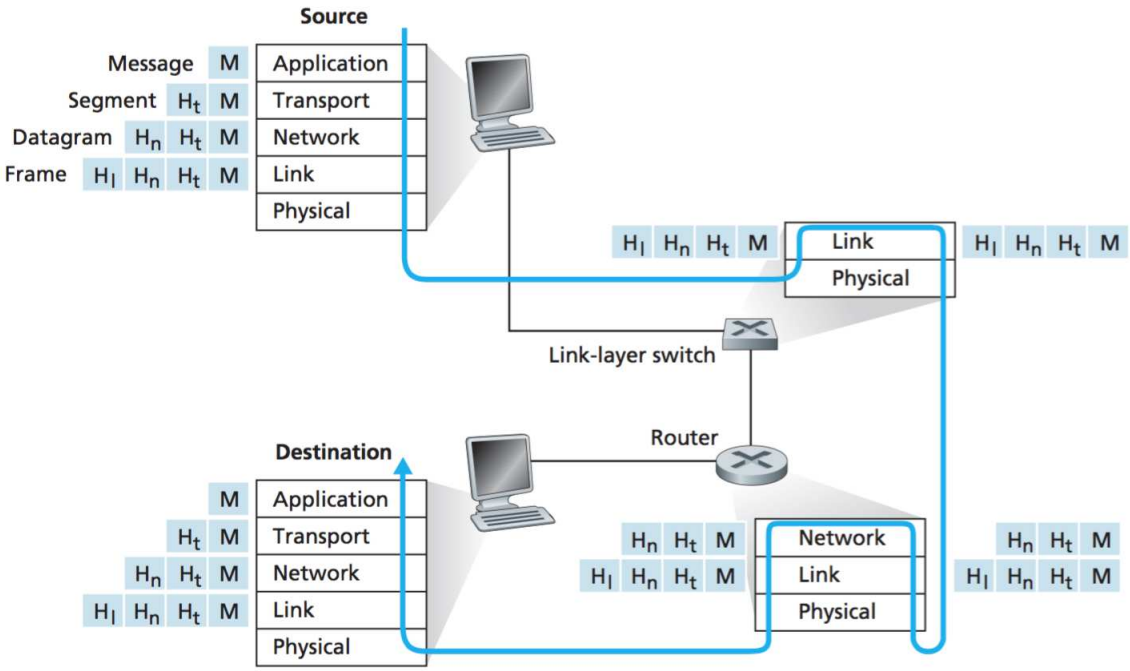

Figure 10 – Encapsulamento de mensagens em um modelo de camadas (KUROSE; ROSS, 2012).

A Figura 10 mostra este processo no contexto do TCP/IP, também conhecido como a suíte de protocolos da Internet. Neste caso, os protocolos de cada camada adicionam um

cabeçalho às mensagens recebidas do nível anterior, o que permite a adição de informações adicionais destinadas ao mesmo protocolo implementado no comunicante.

A.2 PADRONIZAÇÃO DE PROTOCOLOS

A definição de protocolos é apenas o primeiro passo para a intercomunicação de sistemas. Se não existir um acordo a respeito da especificação utilizada será mais difícil estabelecer a comunicação entre serviços (KUROSE; ROSS, 2012). A padronização garante o estabelecimento deste acordo.

O conceito de efeito de rede indica que a adoção de um protocolo se torna mais valiosa à medida que um maior número de entidades também o utilize (LIEBOWITZ; MARGOLIS, 1998). Justifica-se portanto o interesse em incentivar a padronização de protocolos.

A partir do momento em que uma série de entidades entra em consenso a respeito da especificação de um protocolo, estabelece-se um padrão *de facto*. A iniciativa de padronização também pode surgir através de entidades regulamentadoras, como a Organização Internacional para a Padronização (ISO), ou a *Internet Engineering Task Force* (IETF), o que leva ao estabelecimento padrões *de jure* (TANENBAUM; WETHERALL, 2010).

O processo de definição de novos padrões *de jure* depende da entidade regulamentadora relacionada, e ocasionalmente parte de padrões *de facto* já utilizados na comunidade. Geralmente uma especificação é proposta, discutida, e revisada pela entidade antes de se tornar um padrão, o que no caso da ISO pode levar de seis meses a alguns anos (TANENBAUM; WETHERALL, 2010).

Propostas e padrões estabelecidos devem ser documentados de alguma forma. A IETF adota o formato de *Request for Comments* (RFC), publicações que descrevem completamente uma especificação, e estão disponíveis a consulta pela comunidade.

Uma proposta deve cumprir uma série de requisitos antes de ser endossado por uma organização. No caso da IETF, cada proposta passa por vários níveis de maturidade até alcançar a categoria de padrão. Cada um destes níveis pode ser alcançado ao satisfazer as recomendações de determinados grupo da comunidade. Um exemplo destes requisitos é a exigência de uma prova de conceito de interoperabilidade, como uma implementação de referência entre duas ou mais entidades distintas (POSTEL, 1992).

Protocolos de federação também estão sujeitos ao efeito de rede, e apresentam as mesmas necessidades de padronização. O *Simple Mail Transfer Protocol* é um exemplo de especificação utilizado na interoperabilidade de sistemas federados que passou por um esforço oficial de padronização, tornando-se um caso interessante na análise deste processo.

A.2.1 Simple Mail Transfer Protocol

Um sistema de *e-mails* pode ser considerado federado, já que respeita a definição de implementações interoperáveis no modelo cliente servidor apresentada por (BAROCAS et al., 2012). O SMTP é um dos protocolos utilizados na implementação de interoperabilidade entre serviços distintos.

Trata-se de um protocolo para o transporte e entrega de mensagens de e-mail entre processos. A especificação garante que a troca de mensagens aconteça entre clientes que se localizam em redes diferentes, o que permite a construção de um serviço que funcione de maneira confiável sobre a internet (KLENSIN, 2001).

Caracterizado como um protocolo orientado a conexões entre clientes e servidores, ou transmissores e receptores, o SMTP é guiado por uma série de comandos predefinidos. Os servidores também são responsáveis por retransmitir mensagens, caso não sejam os destinatários finais (KUROSE; ROSS, 2012).

A troca de mensagens geralmente acontece em um único salto após o estabelecimento de uma conexão orientada entre o remetente e o destinatário. A retransmissão de mensagens é uma alternativa, utilizada por exemplo em casos em que um usuário moveu sua caixa de e-mails de um servidor para outro e deseja receber as mensagens no seu novo endereço.

Como pode ser visto na Figura 11, o SMTP é um protocolo intermediário entre servidores de e-mail que alternam entre os papéis de transmissor e receptor. Cada um destes servidores possui seus próprios clientes, constituindo sistemas menores onde a interação não é necessariamente coberta pela especificação do SMTP. Os mecanismos de comunicação entre o servidor e o usuário destes sistemas não é necessariamente compatível e eventualmente utilizam outros padrões, como por exemplo POP3 ou IMAP (TANENBAUM; WETHERALL, 2010).

O SMTP eventualmente passou pelo processo de padronização da IETF que levou à publicação de uma RFC. Trata-se de um padrão *de jure* aberto, definido por uma organização ainda que adotado pelo mercado.

Considerando a descentralização da arquitetura e a padronização do protocolo, qualquer novo sistema é capaz de implementar e se tornar parte da federação com relativa facilidade. Nestes casos a heterogeneidade entre os sistemas intermediários é completamente transparente para os usuários do serviço, o que o indica como um protocolo de federação bem sucedido.

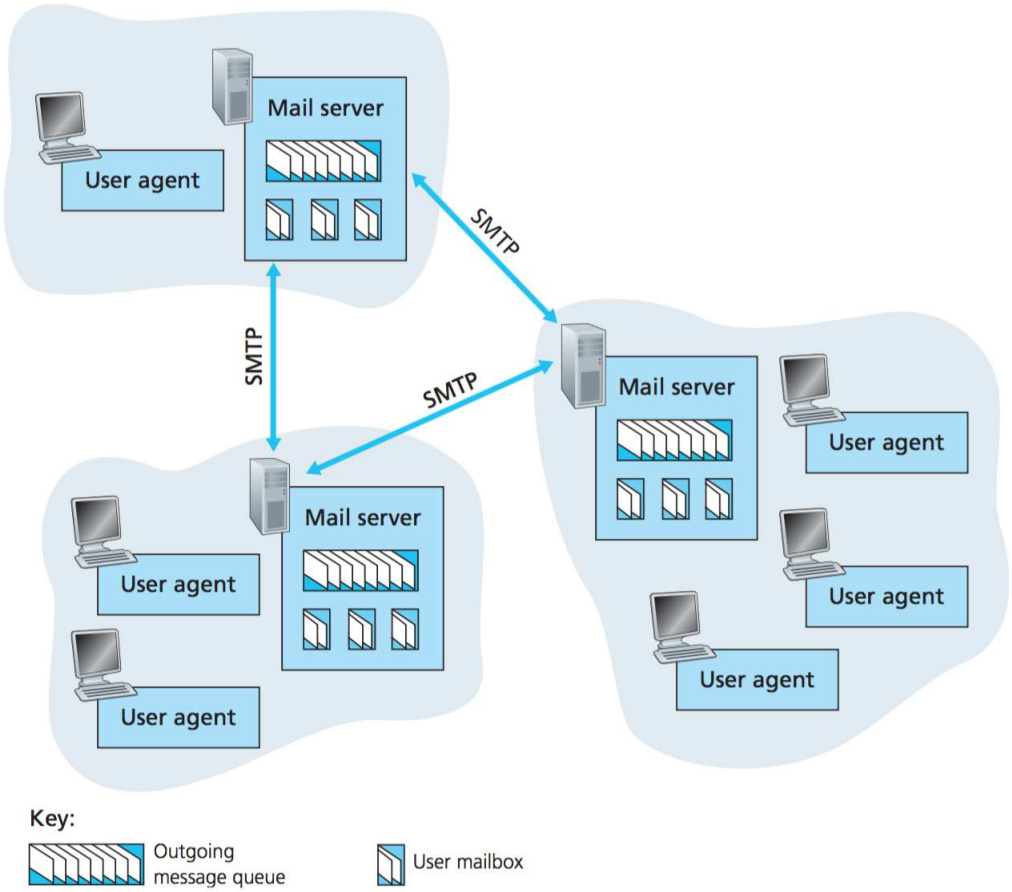

Figure 11 – Uma visão geral de um sistema de e-mails (KUROSE; ROSS, 2012).

APPENDIX B – SUPORTE À FEDERAÇÃO NO NOOSFERO

Já existe uma iniciativa de federação no Noosfero em desenvolvimento por parte da comunidade [1], tendo o autor deste trabalho colaborado desde então. O objetivo é possibilitar a integração tanto com outras instâncias do Noosfero como com outras redes sociais, o que exige a adoção de especificações que tenham o mínimo de aderência na comunidade. A utilização de padrões difundidos amplia as possibilidades de integração dentre outras redes sociais federadas.

Antes da execução deste trabalho os protocolos Diaspora e OStatus já haviam sido escolhidos como referência para a implementação da federação no Noosfero, resultado de uma observação das discussões e execução de projetos como o Hubzilla, Friendica e o próprio Diaspora.

A Seção B.1 descreve as atividades para implementação do suporte à federação que tiveram início antes da execução deste trabalho.

B.1 FEDERAÇÃO ENTRE REDES NOOSFERO

As atividades de implementação de federação já desenvolvidas podem ser separadas em quatro fases, que cumpriram objetivos distintos de integração, que cobrem desde as funcionalidades até a reestruturação da arquitetura da aplicação.

Foi definido que um usuário de uma rede Noosfero pode acessar qualquer outra instância com as credenciais de sua rede de origem. Um usuário federado deve ser capaz de visualizar conteúdos públicos, comentar publicações, seguir usuários, e deixar mensagens em murais. As notificações destas interações devem ser entregues tanto aos usuários na rede local, quanto ao usuário na rede de origem.

O protocolo construído entre redes Noosfero é baseado nas especificações do WebFinger e OAuth para a descoberta de identidade e autorização de perfis, respectivamente. Em relação à comunicação entre as redes, o protocolo Diaspora foi definido como referência.

[1] A discussão a respeito da federação no Noosfero está registrada no endereço a seguir <https://softwarepublico.gov.br/gitlab/noosferogov/noosfero/wikis/federacao>.

B.1.1 Fase 1: preparação

Até a versão 1.5 do Noosfero, todos os relacionamentos entre as entidades da rede eram baseados no conceito de relacionamento simétrico. No entanto, as demais redes federadas, e a maioria dos padrões mais implementados, trabalham apenas com o conceito de relacionamentos assimétricos, o que incentivou o desenvolvimento da funcionalidade de seguidores no Noosfero.

Na fase de preparação foram introduzidos os relacionamentos assimétricos através desta funcionalidade. Os seguidores são notificados a respeito de atividades públicas de perfis seguidos. No Noosfero, cada perfil pode permitir ou não que usuários o sigam. Usuários por sua vez organizam os perfis seguidos em círculos, categorizando suas relações.

B.1.2 Fase 2: intercomunicações

Durante a fase de intercomunicações foi construída a infraestrutura básica para a integração entre redes Noosfero. Ambientes e usuários externos foram introduzidos à arquitetura do Noosfero, que passa a suportar ações de usuários que não possuem perfis locais.

O conceito de usuário externo introduzido nesta fase é importante para toda a implementação da federação do Noosfero. Um usuário local do Noosfero é definido por basicamente dois objetos de negócio — um usuário, que armazena as credenciais de acesso, e um perfil, que armazena suas demais informações na aplicação.

Um usuário externo não possui credenciais de acesso na instância visitada, apenas um objeto que representa o seu perfil externo, e que do ponto de vista da implementação deve ser capaz de reproduzir o comportamento de um perfil comum. A implementação alcançada faz uso de métodos *stub* e relações polimórficas para a reprodução do comportamento.

Nesta fase também foi implementada a especificação do WebFinger, que já está sendo utilizada para a descoberta de usuários na autenticação entre redes Noosfero.

Inicialmente, apenas as redes listadas no diretório central do Noosfero [2] podem ser habilitadas no painel de administração da federação. A descentralização desta lista ou a automatização do processo de descoberta não fizeram parte do planejamento inicial.

B.1.3 Fase 3: integração externa

A fase de integração externa teve como objetivo aproveitar a infraestrutura de usuários externos para autenticar usuários de outros serviços sem a necessidade de perfis

[2] <directory.noosfero.org>

locais. Com isto, usuários de sistemas que suportem OAuth podem acessar o Noosfero, consumir conteúdo, e executar um conjunto limitado de ações.

Durante esta etapa, o *plugin* que torna o Noosfero em um cliente OAuth foi evoluído para permitir que usuários possam tanto criar um perfil local a partir das informações da rede de origem, como também apenas acessar o Noosfero com um perfil temporário. Por enquanto, os únicos fornecedores OAuth suportados são o Google, Facebook, Twitter, GitHub, e o próprio Noosfero. No entanto, novos fornecedores podem ser facilmente adicionados.

B.1.4 Fase 4: inter-relações

A última fase de desenvolvimento da federação de redes Noosfero foi implementar o relacionamento entre usuários de instâncias diferentes. De modo geral, esta fase consistiu em permitir que usuários externos sejam capazes de seguir perfis, comentar conteúdos, e publicar em murais de outros usuários.

De forma a permitir relações entre usuários federados foi necessário refatorar a funcionalidade de seguidores, adicionando o suporte a perfis externos. Neste ponto, usuários federados podem tanto seguir usuários locais, quanto serem seguidos por eles.

Essa fase também envolve a implementação da infraestrutura de troca de mensagens, que seria utilizada nas notificações e interações entre os usuários. Até a execução deste trabalho esse mecanismo não foi completamente definido.

APPENDIX C – DEMAIS CONTRIBUIÇÕES

A documentação da biblioteca utilizada para a implementação do protocolo não cobre todos os aspectos da especificação, portanto foi necessário instanciar dois *pods* locais do Diaspora para analisar a comunicação. A comunicação do Diaspora depende que cada um dos *pods* responda a HTTPS e seja capaz de resolver os nomes dos demais servidores.

Para os testes desse trabalho foram utilizadas duas máquinas virtuais com Debian 8 em rede privada, com os nomes registrados no arquivo de *hosts* para a resolução local. Em cada uma das máquinas, o Diaspora foi servido por um servidor NGINX com SSL configurado com um certificado auto-assinado. Para que a comunicação não fosse prejudicada pela verificação dos certificados, eles foram manualmente adicionados aos certificados reconhecidos em cada uma das máquinas.

C.1 AUTENTICAÇÃO COM O DIASPORA

A federação através do Diaspora foi planejada com base na implementação já existente no Noosfero, que conta com um mecanismo de autenticação entre as redes. Por esse motivo inicialmente se julgou necessário permitir a autenticação de usuários com credenciais de redes Diaspora.

O Diaspora não disponibiliza nenhum *endpoint* de *login* em sua API, mas implementa o papel de fornecedor de identidades OpenID Connect[1]. Portanto, a fim de utilizar as credenciais do Diaspora, é necessário que o Noosfero possa desempenhar o papel de cliente OpenID.

Mesmo que o Noosfero também forneça identidades OpenID, ainda não seria possível usar credenciais locais para a autenticação em *pods* Diaspora, que utiliza apenas estratégias de autenticação local. Isso reforçou a decisão de implementar apenas a função de cliente OpenID por enquanto.

Ainda que o *plugi-in* tenha sido implementado como parte deste trabalho, o restante da federação não depende deste mecanismo de autenticação, já que os usuários só interagem com o servidor remoto através de sua rede de origem.

[1] O OpenID Connect é um padrão de autenticação construído sobre a segunda versão do OAuth. <http://openid.net/specs/openid-connect-core-1_0.html>

C.1.1 Desenvolvimento do plugin OpenID Client

Apesar do Noosfero já oferecer suporte à autorização com OAuth 1.0 através de um *plugin*, foi necessário implementar o suporte ao OpenID. A decisão foi criar um novo *plugin* que transforme o Noosfero em um cliente OpenID.

A *gem* openid_connect foi utilizada na implementação do consumidor. A biblioteca já oferece as diretrizes de descoberta, registro de clientes e autenticação, todas interações que envolvem requisições HTTP ao servidor fornecedor. Um perfil externo também é criado no Noosfero para armazenar algumas informações do perfil remoto, como *link* para o seu perfil, ou sua imagem do avatar.

Os passos realizados pelo *plugin* na autenticação de um usuários são:

1. O usuário digita o endereço do fornecedor OpenID de sua escolha;
2. O Noosfero tenta descobrir informações a respeito do provedor, solicitando informações do emissor de identidades;
3. Caso a resposta do provedor seja válida, o Noosfero solicita o registro como um novo cliente, solicitando acesso a informações necessárias para a criação de um perfil externo;
4. A requisição é redirecionada ao fornecedor OpenID, onde o usuário deve se autenticar, e revisar a solicitação enviada pelo Noosfero;
5. Se o usuário se autenticar no seu provedor e aprovar as informações solicitadas, a resposta do servidor será usada na criação de um perfil externo, e o usuário será autenticado no Noosfero