Technische Universität Berlin

Integrating Didcomm Messaging to ActivityPub-based Social Networks

by

Adrián Isaías Sánchez Figueroa

Matriculation Number 397327

A thesis submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

Master's Thesis

August 9, 2022

Supervised by:
Prof. Dr. Axel Küpper

Assistant supervisor:
Dr. Sebastian Göndör

SNET
SERVICE-CENTRIC NETWORKING

Eidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

Berlin, August 9, 2022 Chuck Norris’ son

Abstract

In this thesis, we show that lorem ipsum dolor sit amet.

Zusammenfassung

Hier kommt das deutsche Abstract hin. Wie das geht, kann man wie immer auf Wikipedia nachlesen `http://de.wikipedia.org/wiki/Abstract`...

Contents

1 Introduction

This section gives an introduction into the general field in which you are writing your thesis. It further describes the situation today, and the problems that are solved and not.

One of the most accepted and used definitions of OSN services was given by Boyd and Ellison in, who defined a Social Network Site as a web based service that allows individuals to

1.1 Motivation

Why choose this topic? Because identities are important. Web3 brings decentralization, and identities should stay behind. W3C provides us with new standards for interoperability, and we should take this goals in mind. [**?**]

1.2 Digital Identities

1.3 Identity Management

Identity management, sometimes called identity and access management, is composed of all the different ways to identify, authenticate and authorize someone to access systems or services within an organization or associated organizations.

There are several problems with our current identity management systems:

A paper-based identification such as a passport, birth certificate or driver's license is easy to lose, copy or be lost to theft. The bureaucracy behind this type of identity management is typically slow and hard to organize. The current identity and access management systems are storing your data on a centralized server along with everybody else's. This puts your digital property in danger as centralized systems are huge targets for hackers. Since 2019 alone, over 16 billion records have been leaked due to hacks and data breaches. This includes credit card numbers, addresses, phone numbers and other highly sensitive personal data. Current identities are not easily portable or verifiable. Blockchain identity management solves these issues.

1.4 Missing factors

It was this gap in technology that prevented digital currencies from being created or adopted in the past. Blockchain fills the gap for currencies, but it also fills the gap for decentralized identity systems, opening up a whole new world of possibilities for data ownership.

Your Blockchain-Based Digital Identity At least 1 billion people worldwide are unable to claim ownership over their identities. This is one of the huge drawbacks of physical identity documentation. It's not widely available in every country. This leaves 1/7th of the entire planet unable to vote, open a bank account, or, in some cases, find a job.

Our current identity management systems are unfair and outdated, but there is a blockchain ID solution: a decentralized identity system that will revolutionize digital freedom. This is more critical now than ever before, with centralized companies left, right and center hoping to create the metaverse.

Your digital identity will be portable and verifiable all over the world, at any time of day. In addition, a blockchain-based decentralized identity is both private and secure. With verifiable credentials, your decentralized identity will empower you to interact with the SmartWeb without restrictions.

Your unique digital identity.

Imagine being able to verify your education qualifications or your date of birth without needing to actually show them. For example, a university degree could theoretically be on the blockchain and you could certify the credentials by checking the university or other issuing authority.

Similarly, you wouldn't have to show your physical ID to verify your date of birth. The authority that wants to know your age could instead use decentralized identifiers and verifiable credentials to find out if you're of the required age or not.

Authorization can be conducted in a trustless manner in which the digital identity in question is verified by an external source, and the person or organization checking can in turn verify the integrity of said source.

The verification of a proof is established by the verifier's judgment of the trustworthiness of the testimony.

This is known as a zero-knowledge proof.

Everyone in a distributed network has the same source of truth. This guarantees the authenticity of data without having to store it on the blockchain.

Blockchain technology has made it possible, for the first time ever, to have a trustlessly verifiable self-sovereign identity.

Blockchain Identity Management Blockchain identity solutions, such as Elastos DIDs, integrate state-of-the-art cryptography technology to ensure that your data is protected and private. By using decentralized identifiers, we can rebuild the structure of several flailing industries.

All of the following have poor identity management and could do with being brought up to date.

1.5 Problem statement

This section explains why this is important, why it is a problem, and why this hasn't been solved already yet. Centralized Identities, secure and open communication protocols. ActivityPub implementations at the present moment rely on HTTPS as their transport, which in turn relies on two centralized systems: DNS and SSL certificate authorities. Is there any way to bring self-sovereignty to the federated social web? [?]

ActivityPub security concerns. Encryption, non-repudiation, confidentiality..... ? → No agreed-upon security mechanisms for ActivityPub. → No encryption in scope of ActivityPub. Research Questions What are the implications of introducing DIDs to Mastodon and ActivityPub in terms of usability, discovery and human-readability? Can a DID-compliant ActivityPub protocol use DIDComm for its standard communication? Can DIDs allow ActivityPub to stop relying on the DNS for its server-to-server discovery? Can DIDComm allow ActivityPub to stop relying on transport-level security for its communication?

1.6 Expected Outcome

The goal of this thesis is to have a fully functioning Mastodon instance that is DID-compliant and that implements ActivityPub using DIDComm as its communication protocol. Replacing the existing centralized identity management with a Self-Sovereign Identity approach through DIDs, and enabling a communication protocol that allows confidentiality, non-repudiation, authenticity, and integrity without being bound to a platform-specific security mechanism.

1.7 Outline

Overview of your thesis structure in this chapter.

2 Related Work

The following chapter covers the most important and significant concepts that are required to fully comprehend the approach of this thesis. This includes a closer look at decentralized communication protocols and identifier standards, as well as social networks that implement them. The revision and structuring of these concepts allows us to understand, build upon, and apply them in order to address our identified research questions.

2.1 ActivityPub

Present first activityStreams to then go ahead and explain how ActivityPub works:

In 2018, Mastodon started using ActivityPub. A decentralized social network protocol based upon the Activity Streams 2.0 data format, which provides a data model for representing an *"Activity"* using a JSON-based syntax [**?**]. ActivityStreams also provides a vocabulary that includes all the common terms you need to represent all the activities and content flowing around a social network. ActivityPub can be described as decentralized, because it provides two different APIs. The first one provides endpoints to connect clients to servers, and the second one, used in federated implementations, includes a server-to-server API [**?**]. After it was published as a Recommendation by the Social Web Working Group of the W3C in January 2018, Mastodon switched and pioneered the use of ActivityPub on a large scale. Promoting this way, its adoption.

2.1.1 Security and Privacy

Some text here...

2.2 Mastodon

In today's most popular social networks like Facebook, Twitter, or Youtube there exists a centralized architecture that keeps millions of users in one single platform. In which control, decision-making, user data, and censorship depend on a single profit-driven organization. On the contrary, Mastodon is a decentralized microblogging social network created with the idea of bringing social networking back into the hands of its users. In the time when switching apps, services every few years and having friends on a dozen different chat applications was a standard in terms of social networking, the German founder envisioned something that could put an end to this, and *last forever* [**?**]. Today, Mastodon is a network of more than 3,500 communities. Each operated by different individuals and organizations that implement their own policies, codes of conduct and discussion topics. By means of this, the user has the opportunity to choose whichever instance he finds the most fitting. Mastodon takes a big share in the Fediverse. A single interoperable ecosystem of different social networks that can communicate with each other. In other words, it is a collection of federated social networks running on free open software on thousands of servers across the world, that implement the same protocol in order to be able to interact with each other. The Fediverse is developed by a community of people around the globe that is independent of any corporation or official institution. It is not profit-driven, and it gives you the freedom of choosing which service and instance of the Fediverse to register to. On top of that, you are free to run your own instance with your own policies and allow other users to join [**?**]. The range of services that can be found inside the Fediverse includes blogging, microblogging, video streaming, photo, music sharing as well as file hosting. Since its standardization, ActivityPub has been widely adopted in the Fediverse, being today the dominant protocol.

2.2.1 Well-known Endpoints

[**?**] As per documentation, Mastodon implements the following 4 different well-known endpoints:

2.2.1.1 NodeInfo

NodeInfo is an initiative to standardize the presentation of metadata about a server operating one of the distributed social networks. The two main aims are to get greater insights into the distributed social networking user base and to provide tools that allow users to select the most suited software and server for their requirements. Mastodon is one of the implementers of this protocol, along with other federated social networks such as Diaspora, Peertube and Wordpress [http://nodeinfo.diaspora.software/]. NodeInfo specifies that servers must provide the well-known path /.well-known/nodeinfo and provide a JRD document referencing the supported documents via Link elements, as shown in . [http://nodeinfo.diaspora.software/protocol.html] Accessing the hypertext reference from the JRD response will give a schematized series of metadata of the instance running the endpoint, such as NodeInfo schema version, software, protocols supported by the server, statistics and even a list of third-party services that can interact with the server via an API. Figure shows the NodeInfo 2.0 schema of the Mastodon instance mastodon.social.

2.2.1.2 Webfinger

Finally, Webfinger is the protocol in which Mastodon heavily relies for the discovery process and for its normal functioning [**?**]. Webfinger allows for discovering information about persons or other entities on the Internet using HTTP that would not otherwise be useful as a locator, such as a personal profile address, identity service, telephone number or an email. Performing a query to a WebFinger endpoint requires a query component with a resource parameter, which is the URI that identifies the identity that is being looked up. Mastodon employs the *acct*[1] URI format, which aims to offer a scheme that generically identifies a user's account with a service

[1] https://datatracker.ietf.org/doc/html/rfc7565

provider without requiring a specific protocol to be used when interacting with the account. In the same way NodeInfo works, it returns a JRD Document describing the entity [?]. Fig. shows an example of the returned JRD that is being provided by the WebFinger endpoint in the mastodon social instance when querying the account "acct:bob@mastodon.social".

```
GET /.well-known/webfinger?resource=acct:bob@mastodon.social
Host: mastodon.social
Accept: application/xrd+xml
```

Listing 2.1: HTTP request to Webfinger endpoint

```
{
   "subject": "acct:bob@mastodon.social",
   "aliases": [
       "https://mastodon.social/@bob",
       "https://mastodon.social/users/bob"
   ],
   "links": [
       {
           "rel": "http://webfinger.net/rel/profile-page",
           "type": "text/html",
           "href": "https://mastodon.social/@bob"
       },
       {
           "rel": "self",
           "type": "application/activity+json",
           "href": "https://mastodon.social/users/bob"
       },
       {
           "rel": "http://ostatus.org/schema/1.0/subscribe",
           "template": "https://mastodon.social/authorize_interaction?uri={uri}"
       }
   ]
}
```

Listing 2.2: Webfinger response

3 Concept and Design

The standards presented in Chapter 2 show the potential improvements that can be achieved in key components of ActiviyPub-based social networks such as identity management, discovery, and communication. In this section, we are going to go through the different steps that are required firstly to integrate DIDs into an ActivityPub-based social network, and finally to enable DIDComm Messaging v2 for its communication.

The outline for this chapter is the following. First, individual concepts and definitions of the ActivityPub implementation are introduced and described. Then, an example of a simple use-case in a contemporary ActivityPub server is going to be illustrated and analyzed in order to be able to compare it with section 4.5, which presents the same use-case but with the proposed concept and design that includes DID integration and DIDComm enablement. Consequently, the reasons behind every decision made are going to be detailed in section 4.6, along with comparisons with other non-formalized proposals.

As explained in Chapter 2, Mastodon is the social network that pioneered the use of ActivityPub on a large scale, and it is also the social network with the most active users and presence inside the Fediverse. For this reason, the ideas presented in this section and the modus operandi of the ActivityPub server are going to be scoped to the actual implementation in Mastodon.

3.1 Definitions

Mastodon has implemented its own ActivityPub server, and with it also its own terms to express different social network vocabulary. In order to prevent confusion or ambiguities, the used terms in this chapter are explained here.

- **Username**: The username in Mastodon consists of a unique local username and the domain of the instance. Ex. alice@example.com
- **Actor object**: In this section, the term Actor object refers solely to the ActivityPub's actor object explained in Chapter 2.
- **Toot**: In the user-facing part of Mastodon, a Toot is the äquivalent of a Tweet in Twitter. This is a small status update with a 500 character limit.
- **Status**: In the backend of Mastodon, the descriptor used for a Toot is a Status. Moreover, an account in Mastodon has a 1:n relationship with status.

3.2 Use case

In order to explain the current ActivityPub flow in Mastodon, let's describe what happens in the simple use case:

Alice has an account in the Mastodon instance mastodon.social and follows Bob, who has an account in the Mastodon instance mastodontech.de. Alice sends a direct message to Bob with the text: "Hello Bob!"

3.3 Today's implementation

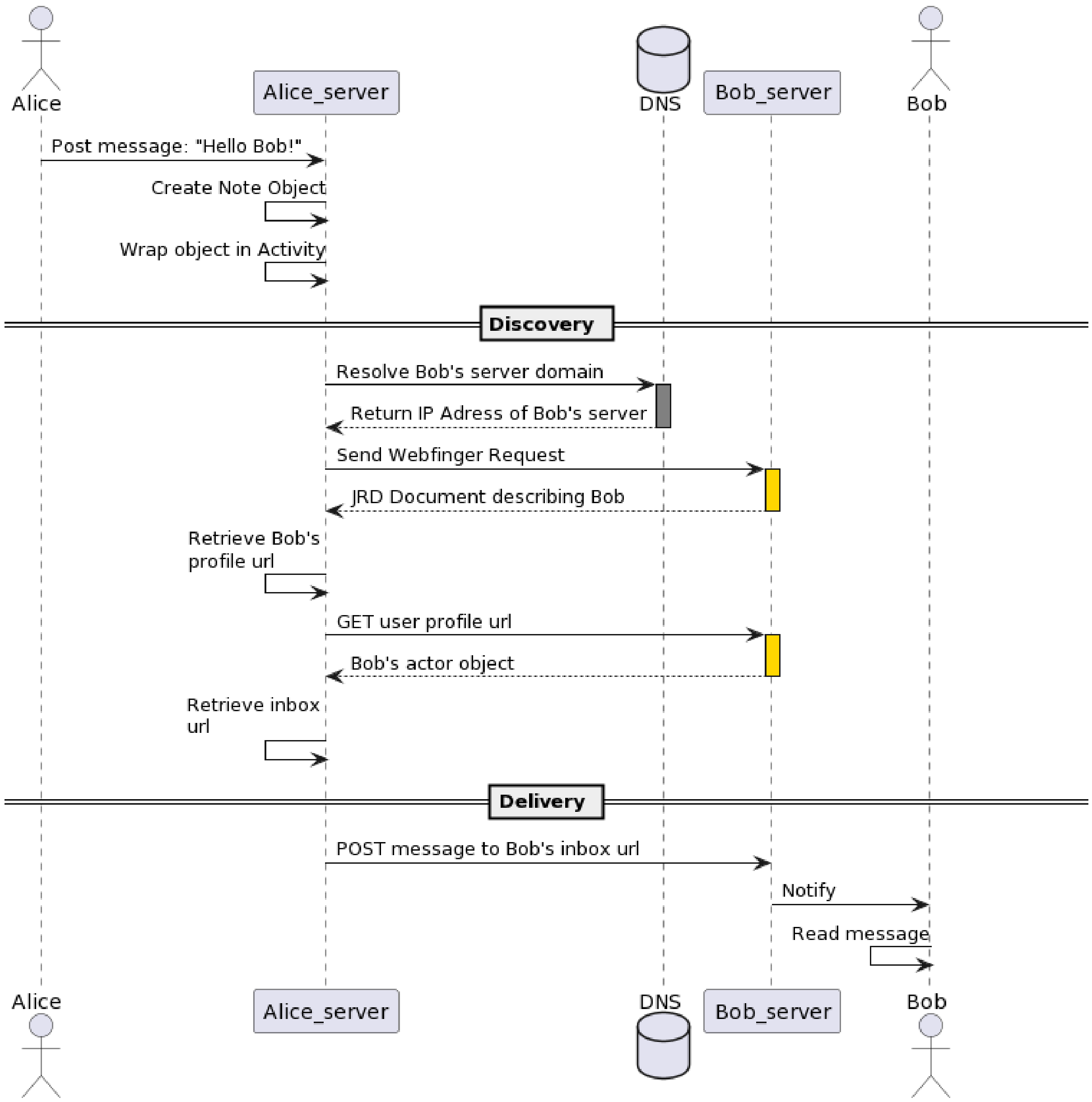

Figure 3.1: Current flow for sending message

3.3.1 Object creation

The first thing that happens when Alice presses the send button is the creation of an ActivityStreams object. In this case, the object is of type *Note*, and will be created by the ActivityPub server inside the Mastodon instance, as shown in 3.2. Then, following the ActivityPub pattern of *some activity by some actor being taken on some object*, the server wraps it in an ActivityStreams *Create* activity, which contains Alice as the actor. This is illustrated by 3.3. Now that the actor, the activity, and the object are well defined and wrapped, it is time to shift our focus to the recipients of this note object. The ActivityPub server will now look at all the fields of the ActivityStreams *Audience*, which includes: to, bto, cc, bcc, and audience [**?**], where all the addresses of the recipients can be retrieved. Afterwards, depending on where the recipient's account lives, the ActivityPub server may take one of two options. Even though the use-case explicitly dictates that Bob's account resides in a different Mastodon instance, both cases will still be explained.

```
{
  "a":  1,
  "b":  "2",
  "c":  true
}
```

Listing 3.1: JSON example

```
  {
    "@context": "https://www.w3.org/ns/activitystreams",
    "type": "Note",
    "to": "http://bob_server.com/users/bob",
    "attributedTo": "http://alice_server.com/users/alice",
    "content": "Hello Bob!"
  }
```

Listing 3.2: ActivityStreams note object

```
  {
    "@context": "https://www.w3.org/ns/activitystreams",
    "type": "Create",
    "id": "https://alice_server/users/alice/statuses/634367/activity",
    "to": "http://bob_server.com/users/bob",
    "actor": "http://alice_server.com/users/alice",
    "object": {
      "type": "Note",
      "to": "http://bob_server.com/users/bob",
      "attributedTo": "http://alice_server.com/users/alice",
      "content": "Hello Bob!"
    }
  }
```

Listing 3.3: ActivityStreams create activity

3.3.2 Same-server delivery

If the recipient's account is on the same server, there is then no explicit discovery process. A simple query in the ActivityPub's server would find the right account and save the status within the account's statuses.

3.3.3 Discovery

On the contrary, when the recipient's account is not in the same server, then a discovery process must be started. Discovery is the fundamental part of the federated side of Mastodon. Without it, users within different instances would not be able to interact, as the instance itself does not know where to find the actor object with all required endpoints to send or receive activities from and to external accounts. For this reason, the current way to look up for other accounts is through the DNS. In the same way Email works, the domain part of the username in Mastodon points to the domain of the instance where the account lives. The purpose of the discovery in this specific use-case is to find the inbox url of Bob, which can can be found in Bob's actor object. As explained in chapter 2, Mastodon includes a series of well-known endpoints that are used to retrieve information about the host itself, as well as resources or entities that are managed under the same host.By default, when the account is not found on the same server, a Webfinger query is performed. In our case, Bob's account lives inside *bob_server.com*. The request shown in 3.4 will return a JRD Document, as shown in fig. 3.5. Based on this document, the Mastodon instance retrieves the link with the *rel: 'self'* which includes the type and the href where the Bob's actor object can be retrieved. If the Webfinger request returns a 404 code, it will then try, as a fallback, using the Host-Meta endpoint. The request and response are displayed in 3.6 3.7. If successful, it will then proceed to take the link template provided and try the Webfinger request once more before throwing an error. After retrieving the needed URL, a subsequent HTTP GET request to this endpoint with the specific *application/activity+json* header will resolve Bob's actor object.

```
GET /.well-known/webfinger?resource=acct:bob@bob_server.com HTTP/1.1
Host: bob_server.com
Accept: application/ld+json
```

Listing 3.4: Webfinger request

```
{
  "subject": "acct:bob@bob_server.com",
  "aliases": [
    "https://bob_server.com/@bob",
    "https://bob_server.com/users/bob"
  ],
  "links": [{
      "rel": "http://webfinger.net/rel/profile-page",
      "type": "text/html",
      "href": "https://bob_server.com/@bob"
    },
    {
      "rel": "self",
      "type": "application/activity+json",
      "href": "https://bob_server.com/users/bob"
    },
    {
      "rel": "http://ostatus.org/schema/1.0/subscribe",
      "template": "https://bob_server.com/authorize_interaction?uri={uri}"
    }
  ]
}
```

Listing 3.5: JRD Document

```
GET /.well-known/host-meta HTTP/1.1
Host: bob_server.com
Accept: application/xrd+xml
```

Listing 3.6: Hostmeta request

```
<?xml version="1.0" encoding="UTF-8"?>
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0">
    <Link rel="lrdd" template="https://bob_server.com/.well-known/webfinger?resource={uri}"
  />
</XRD>
```

Listing 3.7: Hostmeta response

3.3.4 Delivery

Succeeding the retrieval of Bob's actor object and therefore the needed inbox URL, a delivery can now take place. In this case, an HTTP POST request with the previously generated activity object. In order to provide end-to-end message integrity and to authenticate Alice in Bob's server, the request is signed by Alice's ActivityPub server using the HTTP Signature specification. Finally, upon receiving the POST request to Bob's inbox URL, Bob's server verifies the validity of the signature using Alice's public key. After a successful validation, it saves the Note object in Bob's statuses.

As indicated in chapter 2, HTTP signatures are not part of the ActivityPub protocol standard. These integrity and authentication features are within the Mastodon implementation of an ActivityPub server.

3.4 Proposed implementation

Proposed implementation Having seen the current flow of our use case mentioned above in a working ActivityPub-based social network, this section will now address the first research question of this bachelor thesis. That is, the implications of integrating DIDs to a Mastodon instance, and therefore to ActivityPub.

3.4.1 Implications of integrating DIDs

As mentioned in the definitions section, Mastodon's full username includes the domain of the server, and a locally-unique username. This type of username accomplishes the goals of human-readability and uniqueness. In addition, they are resolvable using DNS and the discovery methods previously mentioned in section chapter 2. An ActivityPub actor object using such a username is shown in 3.8.

The first question that needs to be addressed when approaching the integration of DIDs to Mastodon and therefore ActivityPub is, what are the implications of switching from standard mastodon usernames to DIDs. Integrating DID to ActivityPub points immediately to the actor's object. Making the switch would mean that the DID has to be included. Currently, most of the interactions of the ActivityPub server inside Mastodon require the ID attribute to resolve to

```

{
  "@context": [
      "https://www.w3.org/ns/activitystreams",
      "https://w3id.org/security/v1",
  ],
  "id": "http://alice_server.com/users/alice",
  "type": "Person",
  "following": "http://alice_server.com/users/alice/following",
  "followers": "http://alice_server.com/users/alice/followers",
  "inbox": "http://alice_server.com/users/alice/inbox",
  "outbox": "http://alice_server.com/users/alice/outbox",
  "featured": "http://alice_server.com/users/alice/collections/featured",
  "featuredTags": "http://alice_server.com/users/alice/collections/tags",
  "preferredUsername": "alice",
  "name": "",
  "summary": "",
  "url": "http://alice_server.com/@alice",
  "manuallyApprovesFollowers": false,
  "discoverable": false,
  "published": "2022-06-14T00:00:00Z",
}
```

Listing 3.8: Alice's actor object

the Actor's profile and thus the Actor's object. Following a simple strategie, we could simply replace the username with the DID. Thus having an ID attribute that may look like this:

DID URLs provide a lot of freedom of usability for DIDs. In addition to the ID, the actor's object must provide a supplementary set of URLs that point to different collections related to the Actor. These include mandatory attributes, like the outbox and the inbox, and other optional attributes such as the followers and following attributes. What if, instead of using the actual URL of these collections, we specify DID URLs that then point to the correct endpoint inside the DID-Document. This example is illustrated in fig. (Activitypub actor with all DID URLS).

4 Implementation

Describe the details of the actual implementation here...

5 Evaluation

The evaluation of the thesis should be described in this chapter

6 Conclusion

Describe what you did here

List of Tables

List of Figures

Bibliography

[1] Guttenberg, “Double Brackets WiLL preserve cApItAlIzAtIoN,” *Guttenberg Press*, 2011. [Online]. Available: http://www.examplelink.de

[2] C. Webber and M. Sporny, “Activitypub: From decentralized to distributed social networks,” Dec 2017. [Online]. Available: https://github.com/WebOfTrustInfo/rwot5-boston/blob/fcbc33835c1b76b7526a8a82cd9cac9c23828711/final-documents/activitypub-decentralized-distributed.pdf

[3] “Activity streams 2.0,” May 2017. [Online]. Available: https://www.w3.org/TR/activitystreams-core/

[4] C. Lemmer-Webber, J. Tallon, A. Guy, and E. Prodromou, “1.1 social web working group,” Jan 2018. [Online]. Available: https://www.w3.org/TR/activitypub

[5] S. Tilley, “One mammoth of a job: An interview with eugen rochko of mastodon,” Jul 2018. [Online]. Available: https://medium.com/we-distribute/one-mammoth-of-a-job-an-interview-with-eugen-rochko-of-mastodon-23b159d6796a

[6] J. Holloway, “What on earth is the fediverse and why does it matter?” Oct 2018. [Online]. Available: https://newatlas.com/what-is-the-fediverse/56385/

[7] M. Nottingham, “Rfc 8615 - well-known uniform resource identifiers (uris),” May 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8615

[8] E. Rochko, “Webfinger,” Jan 2020. [Online]. Available: https://docs.joinmastodon.org/spec/webfinger/

[9] P. Jones, G. Salgueiro, M. Jones, and J. Smarr, “Rfc 7033 - webfinger,” Sep 2013. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7033

Appendices

Appendix 1

```
for($i=1; $i<123; $i++)
{
    echo "work harder! ;)";
}
```