ActivityPub - Final thoughts, one year later.

2019-01-13 2019-01-17 activitypub, diaspora, opensource, software, standards

Almost exactly one year ago, I published a blog post with lots of detail why I deemed ActivityPub not suitable for implementation as the base federation layer in diaspora* at that time. Over the course of last year, I have received multiple requests for a follow-up. Unfortunately, my opinion still holds up today, but after some consideration, I figured I should post a follow-up anyway, with some additional notes and things I have learned during that time.

Preface

Before continuing here, I would like to recommend two other articles first. This piece by ActivityPub co-author Christopher Webber explains the difficulties of working in this working group and goes into great detail about mixed expectations and the attempt of bringing everyone together. Also, there is this piece by Friendica contributor Michael Vogel, who explains the difficulties they experienced while adding ActivityPub support, and how they attempted addressing those.

As you can probably extract from the title, this article will more than likely be my final piece on ActivityPub, and I actually will not even go into any technical details this time. My goal is to use this as a reference piece to point towards if questions arise. At this point, everyone is spinning in circles, and I do not feel like we actually *can* make any progress. I will explain the reasoning for that a bit later in this post.

What ActivityPub is

One of the most common *misunderstandings* about ActivityPub I see starts at the very simple question of what ActivityPub is, and what the SocialWG wanted it to be. To answer that, we should start by looking at the *Social Web Working Group Charter*, which basically is the document that describes the goals of that working group:

> a common JSON-based syntax for social data, a client-side API, and a Web protocol for federating social information

So basically, they want to build two things:

- a protocol used to exchange *stuff* between servers, and also between servers and clients; and
- a JSON'y syntax for representing *social data*

Let's look at their most prominent results.

ActivityPub

ActivityPub is the "web protocol for federating social information" part of the equation, but it is also responsible for the client-to-server interaction. ActivityPub is not a full social network federation protocol. It is a *transport protocol* to transfer *things* between servers and clients[1].

If you are not too familiar with web terminology, think of ActivityPub as the postal service. The postal service drives around and delivers letters into inboxes, but it does not care about what actually is *inside* those letters. This is not a perfect analogy, but close enough.

Activity Streams

Activity Streams is the "JSON-based syntax for social data" the Working Group wanted to design. Activity Streams is nothing the SocialWG directly invented. It is sticking around since 2011, where the first version was published. The SocialWG used Activity Streams 1.0 as a base, modified some parts, added some other parts, and published the result as Activity Streams 2.0.

To understand what Activity Streams is, think of it as an abstract syntax to represent basically anything that can be an *action* on social media. The Activity Streams Vocabulary specification defines, amongst other things, three types of objects:

- **Actors**: `Application`, `Group`, `Organization`, `Person`, `Service`.
- **Activity types**: `Accept`, `Add`, `Announce`, `Arrive`, `Block`, `Create`, `Delete`, `Dislike`, `Flag`, `Follow`, `Ignore`, `Invite`, `Join`, `Leave`, `Like`, `Listen`, `Move`, `Offer`, `Question`, `Read`, `Reject`, `Remove`, `TentativeAccept`, `TentativeReject`, `Travel`, `Undo`, `Update`, `View`.
- **Objects**: `Article`, `Audio`, `Document`, `Event`, `Image`, `Note`, `Page`, `Place`, `Profile`, `Relationship`, `Tombstone`, `Video`.

To build a valid Activity Streams activity, you pick one of each category and add some metadata to it. You describe that *something* did *something* to or with *something*, and you explain those *things* in more detail.

This concept is pretty cool. If the list of object types is complete (or can be extended), you can represent everything you need with that. So, kudos to the Working Group for achieving their goal here.

What it is not

Now that we have established a baseline on what ActivityPub is, we have to talk about the more controversial topic. About the things ActivityPub *is not*. Everyone agrees that ActivityPub and its sister specifications are a great *framework* for building federated social network interactions. However, I am having a hard time considering it as an actual *protocol* to implement in social networks to build reliable communication channels.

I have explained this in detail in my last post, and I will not repeat myself here. However, to summarize, there is a lot of *wiggle room* in the specification. For example, it is not clear if there are any required attributes for an object type, and if so, which. Activity Streams also explains that it can be extended in any way imaginable, with basically no limitations:

> […] an “extension” is any property, activity, actor or object type not defined by the Activity Vocabulary. Consuming implementations […] MUST continue processing the

> items as if those properties were not present.

With this amount of extensibility cooked right into the specification, there is no need to ask if this is a bug or a feature. It is a feature, and it was even part of the Working Group's charter. This is fine, if we see ActivityPub as a framework, but to most people, it seems like ActivityPub is something different.

The lack of by-design interoperability

I acknowledge that this is a bit of a harsh headline, and it is not what the specification intended, but it is where implementations are headed right now.

When the ActivityPub recommendation was published, it received a lot of hype. Deservedly so, because, as I mentioned earlier, I feel like it is an excellent platform to build things on top of it. However, I feel like this hype is where things took a wrong turn somewhere. Projects started to use the ActivityPub brand as an icon for immediate and guaranteed communication channels, across project borders. They wandered around the internet and told projects to implement ActivityPub to join *the Fediverse*, and by merely adding ActivityPub support, users would be able to communicate with each other, they claimed.

Very early, I raised concerns about the *actual* interoperability, claiming many pieces of the specification are too ambiguous, or simply undefined, for the spec to be a reasonable source of truth when implementing a social network. The point where I started asking projects and implementers about their plans to ensure interop, things got interesting. I was able to have many excellent discussions with people who wrote chunks of the specs or who were involved in other ways in places like the SocialWG's IRC channel, and I also had interesting discussions with people being involved in other specifications in places like the IndieWeb channels.

To my surprise, those people never wholly disagreed with my points. Even people who contributed to ActivityPub and Activity Streams agreed that there is some validity in those claims, and while nobody ever thought ActivityPub was *dead* because of that (not even me, as you should be able to tell by now), people agreed that these are important

points to have discussions around.

Guess who is not agreeing with me? The newfound group of ActivityPub evangelists, and some implementers. Those folks claim my *explanations are lousy*, and that my demonstrated *"obstacles" are just vague statements*, but yet owe me (and everyone else) actual proof of that.

Putting ideology over users

Unfortunately, this is a pattern we can observe quite often, especially in the *hardcore* free and open source software world. It is indeed quite common for people there to put their ideology over the actual user's needs, and this saddens me quite a bit.

I always tried to make it very clear that the lack of specified interoperability is dangerous. Not for the people implementing it, but for the people *using* it. That is, in fact, the point I am most concerned about in regards to projects implementing ActivityPub.

In a recent GitHub comment, I questioned the use of the word *ActivityPub* as a label to slap on things, because I fear it is going to confuse users. In my opinion, we should be building solutions that appeal to the masses who still use large, centralized networks. If we only ever stay within the bubble that *understands* about protocols and interoperability issues, there will be no growth for us. Moreover, while I do not directly see diaspora* as the "new big thing", I would like to live in a world where *all distributed social networks* are the "new big thing". In one recent decision, after being confronted with the statement that implementations seem to praise ActivityPub as the magic bullet to solve all interop issues, one implementer claimed:

> I doubt that anyone who has ever looked at ActivityPub thinks that everything that supports ActivityPub should be automatically compatible with all the other implementations. This is not realistic. I don't know of any implementation claiming that, and there will never be one.

This was one of the claims that stunned me because I knew for a fact that this was simply untrue. Quick reality check about the expectations raised by people and projects in this

field:

- Christopher Webber in the official FSF blog: *"I agreed that we wanted to make sure that whatever federation API we used was as broadly compatible with other libre social networking sites as possible"*
- The official NextCloud blog about their new support of ActivityPub: *"By using the popular ActivityPub standard, Nextcloud users can subscribe to and share status updates with users in the so-called 'fediverse', an interconnected and decentralized network of independently operated servers!"*
- The official Mastodon blog about their thoughts on the future: *"The social network that is Mastodon isn't really Mastodon. It's bigger. It's any piece of software that implements ActivityPub. [...] All of these platforms are different and they focus on different needs. And yet, the foundation is all the same: people subscribing to receive posts from other people. And so, they are all compatible."*

The wording used in those announcements is pretty clear to my understanding.

Putting the ActivityPub logo on a project's website and writing "we support ActivityPub" announcement posts makes technically versed people very happy, and people supporting open standards will read them with shining eyes. However, there is a secondary effect: these announcements carry over something to non-technical users as well. It tells users that this piece of software is *compatible* with other pieces of software that carry the same logo. But it is not. In another recent discussion, when someone asked me why diaspora* does not support ActivityPub yet, I claimed the project has two options here, which has a direct impact to how we can explain the compatibility with users on other networks:

> 1. Sorry, Alice, Bob is using software that is not compatible with us, so you can't communicate with Bob here.
> 2. Yes, you can communicate with Bob, but since he is using ExampleNet, please be aware that Bob will not receive your photo albums and will be unable to interact with those. Carol will see your photos, though, but unfortunately, she will not be able to see your geo-location updates. Moreover, because of technical limitations, Dan can comment on your posts, but we cannot make sure that Carol and Bob see those, because we cannot redistribute Dan's comments.

I asked one person vigorously defending ActivityPub in that discussion to tell me what they would prefer, but they did not answer. Quoting someone working on an ActivityPub

implementation and actively pushing other projects to do so the same, upon being confronted with the above situation:

> Yes, ActivityPub means an “inconsistent user experience”. That’s part of the deal. If you don’t accept that - don’t build a federated platform. The nature of the federation is to provide an inconsistent experience on some ends of the network.

When I wrote the article last year, I honestly thought everyone was building federated systems *for their users*, not for their warm feeling for implementing some new standards. I was wrong, and people have now accepted that whatever comes out at the end is probably not a good solution for their users, and whenever this point comes up, the critique simply gets disregarded. This is deliberately throwing non-technical user’s expectations under the bus.

”But we can talk to each other to make it compatible”

With Michael Vogel’s recent blog post being up, people now claim it *debunks* my original post, leaving me quite baffled. I do not think I ever claimed what Michael did is impossible, and I do not see what he is trying to *debunk* here. In his post, he describes, in a series of points somewhat related to my post, the issues caused by ambiguities he experienced, and how he resolved those. His ultimate conclusion:

> The specification leaves space for interpretation. So - to ensure interoperability - the different implementing persons should talk to each other and should discuss solutions that are fine for all.

So in the end, I am very grateful that he took the time to write this article, as it is not only not *debunking* my article, it actually *confirms* what I claimed all along.

There is nothing wrong if implementers prefer reading other people’s code to understand what they need to do, and there is also nothing wrong with asking questions to other implementations if there is something unclear. I acknowledge that the language used in specifications can be quite complicated, and those documents tend to be a bit on the

longer side. People referring to other implementations for guidance is not an issue, I actually can understand that very well.

The issue arises when you *have to* read other project's source codes and when you *have to* ask questions to be able to implement something. If that is the case, then the *knowledge* needed to implement this protocol reliably is no longer collected inside the specifications, but in other people's bug trackers, source code snippets, and brains. Without access to that information, you cannot build a reliable implementation.

This might be fine for some use cases and some specifications, but when building a federated social network with supposedly open protocols, this is *absolutely unacceptable*. What some of the current implementers are doing right now is, excuse the wording, shockingly naive and ignorant towards everyone but themselves. I am sure those people have a lot of "oh look, it works!" moments these days, and on reading this article, I am relatively sure most of them will heavily disagree with everything I write because *it works for them right now*. However, this will not end well.

Let us imagine a future project, starting their work in two years from now. What are they expected to do? There will be no central specification for them to look at[2]. There might be a lot of different projects using ActivityPub in the future, so is the new implementer expected to venture through all of their code and all of their bug trackers? What happens if they do not understand other people's code? What happens if people from other projects do not have the time or interest to reply to the new project's questions? What happens if they miss an important decision because it was made in some projects GitLab instance that is hard to discover? Do future implementations really have to depend on *luck*?

The sole purpose of specifications and the support structures around them is to *avoid* precisely that. Specifications are not there to be an inspiration to someone; they are there to be the single source of truth in questions about the thing they are trying to specify. Discussions about specifications should happen *at* those specifications so that everyone has a chance to participate in those processes, and to keep the whole history conserved and accessible to everyone, even in the future.

The federated social networking universe is a world where many people have solid

opinions on the open web, open processes, and open standards. People who complain vocally about how bad everything is when one browser vendor decides to stop their efforts, writing essays on how bad it is that the decision-making gets pushed outside the standardizing organizations. People who yell at others for driving decisions in a "not open or not visible to everyone" space. And yet here we are, with the same people doing exactly the things they claim to hate, wittingly.

How people *could* have worked together

I never pushed for ActivityPub to be rewritten, and I never pushed for stricter rules to be added to ActivityPub or Activity Streams itself. Although I have been asking for stricter rules, I explained multiple times that this does not have to happen inside ActivityPub itself[3]. The problem of having maximal extensibility as a goal while also making sure that systems and implementations are interoperable is nothing new. It has been done a couple of times, and there are great examples the federated social networking space could have learned from.

I would like to hold XMPP up as an example, here. If you do not know XMPP, it is short for *Extensible Messaging and Presence Protocol*, a protocol for building distributed instant messaging applications. While extensibility was so important they even made it part of the protocol's name, they did not want to end up with a runaway specification that depends on *having implementers talk to each other* to be working reliably.

Instead, they wrote a very strict base set of the absolute minimums required to build on XMPP, and pushed that through the IETF's standards track, ending up with what is now RFC 6120. Acknowledging that they can never address all needs in such a strict specification document, they opened themselves for more flexibility:

> This document specifies how clients connect to servers and specifies the basic semantics of XML stanzas. However, this document does not define the "payloads" of the XML stanzas that might be exchanged once a connection is successfully established; instead, those payloads are defined by various XMPP extensions. For example, [XMPP-IM] defines extensions for basic instant messaging and presence functionality. In addition, various specifications produced in the XSF's XEP series [XEP-

> 0001] define extensions for a wide range of applications.

They introduced XEPs, XMPP Extensions Protocols. The very first XEP ever published, XEP-0001, describes the process of getting a new XEP published to the world, and that process of that is not very hard to understand or tricky to follow. *Everyone* is invited to submit a proposal, you do not have to be a member of any organization, and in fact, you do not even need to maintain your own implementation. As long as your proposal follows the rules and matches the required format, it will be taken into consideration. After a proposal has been made, there is a specific chain of events, but it boils down to an open discussion on those proposals, and if there are no more doubts or things to improve, and if people agree that the proposal would be a useful thing to standardize, the proposal will eventually end up as an actual XEP.

At the time of writing, there are 151 active, draft, experimental, final, or proposed XEPs, and all of them can be found on the XMPP Standards Foundation's website. Everyone interested in reading up on those topics, for example because they want to implement a client or a server, can find those documents there, and there is also a central infrastructure for asking questions and proposing improvements, which is open to everyone. This way, the XSF has ensured XMPP can be extended to all imaginable use cases, without ending up with watered-down or vague specifications.

If, for example, I wanted to implement an XMPP server or a client[4], I could read the relevant RFCs as well as the XEPs that are relevant to my project. Some documents, like XEP-0387 for example, would even help me out deciding *which* XEPs are essential, by listing the things that are important when building a web client, or an advanced desktop client, or a mobile application, … There would be no need for me to *talk* to other XMPP server or client implementers for my implementation to work, and I could be very comfortable about the interoperability of my systems if I built them in accordance to the specifications. That is what specifications are meant to be.

It does not take much to imagine an *ActivityUniverse Standards Foundation*, with everyone working on an implementation in the board, the same open submission process, the same open approval workflow, and reliable and complete specification documents in the end. There could have been a very strict base set of things that are *absolutely* needed, and a set of optional *extensions*, alongside a mechanism to reliably discover support for

those extensions. Besides, that *base set* could also address the problem of receiving contents the receiver is not capable of parsing. Maybe we could have come up with an alternative representation, and maybe we could have figured out a way to embed those contents, like a more secure, more sandboxed OEmbed.

It probably would have been surprisingly little effort, if everyone acknowledged the need for that in the first place, and not ran off to build their own little bubble. Now, people have *already* build systems on top of the current set of mutual agreements, so why should they even participate in specifying something that ends up being incompatible with their implementation, requiring even more time to adjust those. Even if suddenly many people agree that this is how the federated network space should be working on things, it, unfortunately, is unlikely that it will succeed. And that is mainly because so many people were not interested in meaningful discussions in the first place.

Conclusion

Quite frankly, I am having a hard time coming up with a conclusion here.

It is not a surprise that the space of federated social networks is a difficult one. Everyone *using* or *working* on a piece of software in this space is very opinionated, and probably has personal reasons to do the work in the first place. However, this is usually a good thing. It turns out it probably is not a good thing. Having meaningful discussions on the nature of ActivityPub is impossible when some people go ahead and disregard everything you say because they do not like you as a person. I expected people to be able to differentiate between their personal opinion and their professional opinion, but it seems like that, even today, too many people are unable to do so. Instead of taking the arguments I wrote throughout the last year so many times into consideration, it was easier and *more comfortable* to disregard them. It could have been a better space.

To me, it is almost a bit funny when people suggest that diaspora* is digging its own hole, or claiming that everything we did not invent ourselves is not good enough for us, merely for taking a step back and questioning if we, as a social web collective, do the right things, or if we should be doing something differently. I really do not care if diaspora* ends up dying out, and in fact, I would be happy to let it die if there is something better. But for

that to happen, we have to be sure we actually build something better. This is achieved by working *together*, not by bashing each other's heads out of personal reasons.

Amendments

- **2019-01-14**: Cory Slep has reached out to me with their blog post published just while I was writing this. There is some overlap in opinions here, and it is an interesting read as well. They are a bit on the optimistic side, assuming that most people strive to build compatible solutions and that most people are interested in maintaining that as a long-term goal. While I *hope* that is indeed the case, I do not want this space to be based on someone's hopes. If people already have discussions, then why not have those at a central entity and ultimately cast the results in referenceable specification documents?
- **2019-01-14**: There also is a series of blog posts by kaniini, one of the people behind Pleroma, describing ActivityPub as the "'Worse Is Better' Approach to Federated Social Networking". The posts go into great technical detail behind some of the challenges working on an interoperable AcitivyPub implementation, and while I do not agree with everything they say, some of the points are valid.

Appendix

This post forced me to call out some groups of people, and individual people. I would like to stress that not a single point on my list of interpersonal conflicts is related to any of the folks who contributed significantly to the creation of the ActivityPub specification and the push of Activity Streams. Quite the opposite is true, every time I had interactions with some of those people, it ended up being very constructive and friendly, and I still have massive respect to the authors for what they achieved. Whenever I am talking about individuals in this article, I am explicitly not talking about those who did great work. The problem mainly lies within a different group of people. I tried to make that clear, but please reach out to me if you have any concerns, or even feel personally offended.

Some of the quotes appearing in this article, especially those from people *strongly disagreeing* with me on a personal level, have been made without linking the source. Also, I altered the wording, while making sure the substance stayed untouched, to make them

less easy to discover using search engines. Although the statements have been made in public, there is no point in pointing fingers at someone else. I am quoting these people exemplarily to make a point about the reasoning used against my opinion, and singling out individuals would only fuel personal conflicts, which I am not at all interested in.

Footnotes

1. Yes, ActivityPub references Activity Streams for its content, but that does not change the fact that those two are different things, and I want to talk about those things separately. ↩
2. Besides me rambling around all the time, the only other serious attempt I have seen so far was the suggestions of LitePub, but at the time of publishing this article, their *documentation* website is still nothing but a collection of ideas and 404 links to non-existent actual documentation. It's been more than 6 months, and people supporting LitePub in its beginning are now happily participating in project's bug trackers all over the internet to work on different *dialects* of ActivityPub, so I do not expect this to turn into a fully developed effort, unfortunately. ↩
3. And in fact, I think it should *not* happen inside ActivityPub or Activity Streams. The nature of these things, especially of Activity Streams, is to be open-ended, and I see no reason to change that. One can build on top of things. There is no need to replace the foundations of that. ↩
4. Spoiler: I have done *both*. ↩