How decentralized is Bluesky really?

By Christine Lemmer-Webber on Fri 22 November 2024

Recently due to various events (namely a lot of people getting off of X-Twitter), Bluesky has become a lot more popular, and excitement for its underlying protocol, ATProto, is growing. Since I worked on ActivityPub which connects together Mastodon, Sharkey, Peertube, GotoSocial, etc, etc, etc in the present-day fediverse, I often get asked whether or not I have opinions about ATProto vs ActivityPub, and the answer is that I do have opinions, but I am usually head-down focused on building what I hope to be the next generation of decentralized (social) networking tech, and so I keep to myself about such things except in private channels.

This debate has been growing harder to ignore, with articles ranging from "Bluesky is cosplaying decentralization" on the one hand and "Nobody cares about decentralization until they do" on the other (which I suppose went subscriber-only; it had a big splash recently and wasn't previously) in favor of ATProto and arguing that other approaches are not as decentralized. Still, I mostly believed that anything I had to say on the subject would not be received productively, and so I figured it was best to reserve comment to myself and those close to me. But recently I have received some direct encouragement from a core Bluesky developer that they have found my writings insightful and useful and would be happy to see me write on the subject. So here are my thoughts.

Let us open with a framing. Decentralization is the result of a system that diffuses power throughout its structure, so that no node holds particular power at the center. "Federation", as has been used as a *technical* term since the emergence of the "Fediverse" (which presently is mostly associated with ActivityPub, though I would argue XMPP and email are also federated), is a technical approach to communication architecture which achieves decentralization by many independent nodes cooperating and communicating to be a unified whole, with no node holding more power than the responsibility or communication of its parts.

Under these definitions, *Bluesky and ATProto are not meaningfully decentralized, and are not federated either*. However, this is not to say that Bluesky is not achieving something useful; while Bluesky is not building what is presently a decentralized Twitter, it is building an excellent replacement for Twitter, and Bluesky's main deliverable goal is something else instead: *a Twitter replacement, with the possibility of "credible exit".*

Bluesky's strengths

I'm sure some people are already bristling having read the previous paragraph, and I will get to explaining my rationale, which is a technical analysis in nature. But let's open with the positive, because I think there are positive things to say about Bluesky as well.

Bluesky has done an incredible job scaling to meet the present moment. Right now, Bluesky is facing a large influx of users who are looking for an alternative to X-Twitter since Musk's takeover (and particularly since Trump's re-election). In other words, the type of user who would be sympathetic to

the post X is a White-Supremacist Site is now looking for someplace else to be. The future of X-Twitter is a place where only hard-right people are going to feel comfortable; anyone else is going to be looking for a replacement *now*, and Bluesky is going to be their quickest and easiest option.

In many ways, Bluesky was built for this. Its experience is basically a one-to-one feature-for-feature replacement for the Twitter that many people loved. And the original directive that Bluesky was given, when it was Jack Dorsey and Parag Agrawal's joint pet project (my understanding at the time was that it was Jack's vision, but Parag took the lead, and my impression was also that they were both very sincere about this) to kick off a decentralized protocol *which Twitter could adopt*. This informed a lot of the initial architectural decisions of Bluesky, including its scaling needs. It also incidentally lead it to become an excellent offboarding platform when it turned out that many X-Twitter users no longer felt comfortable on the platform. You miss old Twitter? Bluesky already has been building an alternative: hop on board, it's just like old Twitter!

The fact that Jack Dorsey kicked off Bluesky as an initiative and a funded effort and that Jack was originally on Bluesky's board often leads to snarky or snide comments on the fediverse that Bluesky is owned by Jack Dorsey. However, this isn't true: Jack Dorsey quit Bluesky and has been focusing on Nostr (which I can best describe as "a more uncomfortable version of Secure Scuttlebutt for Bitcoin people to talk about Bitcoin"). So I don't think this particular criticism holds true. Bluesky is also fully independent of Twitter; my impression is that this only happened because Jay Graber (Bluesky's CEO) very carefully negotiated to make sure that when Bluesky got its funds that it would receive them without Twitter having control, and this shows a lot of foresight on Jay's part.

For that matter, I think the part of Bluesky I probably respect most personally is Jay Graber. I was not surprised when she was awarded the position of leading Bluesky; she was the obvious choice given her leadership in the process and project, and every interaction I have had with Jay personally has been a positive one. I believe she leads her team with sincerity and care. Furthermore, though a technical critique and reframing follows, I know Jay's team is full of other people who sincerely care about Bluesky and its stated goals as well.

There is one other thing which Bluesky gets right, and which the present-day fediverse does not. This is that Bluesky uses content-addressed content, so that content can survive if a node goes down. In this way (well, also allegedly with identity, but I will critique that part because it has several problems), Bluesky achieves its "credible exit" (Bluesky's own term, by the way) in that the main node or individual hosts could go down, posts can continue to be referenced. This is possible to also do on the fediverse, but is not done presently; today, a fediverse user has to worry a lot about a node going down. indeed I intentionally fought for and left open the possibility within ActivityPub of adding content-addressed posts, and several years ago I wrote a demo of how to combine content addressing with ActivityPub. But nonetheless, even though such a thing is spec-compatible with ActivityPub, *content-addressing is not done today on ActivityPub*, and *is* done on Bluesky.

Bluesky's architecture and centralization

On blogs, search engines, and Google Reader

> When you build architecture that in theory anyone can participate in, but the barrier to entry is so high so that only those with the highest number of resources can participate, then you've still built a walled garden. -- Morgan Lemmer-Webber, (summarizing things succinctly in our household over breakfast)
>
> Think of our app like a Google. -- ATProto quick start guide

I believe that to this day, the web and blogs are still perceived as a decentralized and open publishing platform. However, I don't know of anyone who would consider Google to be decentralized. In theory, anyone could build their own search engine: the web, which is being consumed and indexed, is trivial enough to parse and aggregate. However, few can, and in practice, we see two (or maybe three) search engines in practice alive today: Google, Bing, and maybe DuckDuckGo (which, per my understanding, uses several sources, but is largely Bing).

This is all to say, in many ways Bluesky's developers have described Bluesky as being a bunch of blogs aggregated by Bluesky as a search engine, and while this isn't really true, it's a good starting point for understanding its challenges. (To understand the rest of the terms involved in the document in detail, Bluesky's architecture document is a good source.)

The same way that in theory the web and blogs are not tied to Google, neither are ATProto's Personal Data Stores (necessarily) tied to Bluesky, the company. Since a small number of people are running Personal Data Stores right now, which is quite viable, Bluesky may have the appearance of being decentralized. And at present, there's really only one each of the Relay and (Twitter-like) AppView components used in practice, but there is a real possibility of this changing and real architectural affordance work to allow it. So perhaps things to not seem all that bad.

However, if we look back at the metaphor of blogs and Google, it's important to note that before social networking in its present form took off, blogs and the "blogosphere" were the primary mechanism of communication on the internet, aggregated by RSS and Atom feeds. And RSS and Atom feed readers started out with an enormous amount of "biodiversity" and largely ran on peoples' desktops. Then along came Google Reader and... friends, if you are reading this and are of a certain age range, there is a strong chance you have *feelings* just seeing the phrase "Google Reader" mentioned. Google Reader did a great job of providing not all of RSS and Atom feed readers, but enough of it that when Google shuttered it, blogging (and my favorite offshoot of blogging, independent webcomics) crumbled as a primary medium. Blogs still exist, but *blog feed aggregation* fell quickly to the wayside. Browsers removed the feed icon, and right around that time, social networks in their present shape took their place. To this day, blogs are now primarily shared on social networks.

This is all to say: blogging plus feed readers started out a lot more decentralized than Bluesky has, and having one big player enter the room and then exit effectively killed the system.

And that's even without feed readers being particularly expensive or challenging to run as independent software. I have this concern for the fediverse as well (in case you think this article is harsh on Bluesky and I am a fediverse fangirl because I co-authored the spec it uses, stay tuned; I plan on releasing a critical analysis of the fediverse as-is here shortly). mastodon.social is certainly a "meganode" and

Threads... let's not even get *started* with Threads, that's a long topic. And running your own server is much more challenging than I'd like. But even so, if you check some of the fediverse aggregators such as FediDB or Fediverse Observer you will see thousands of servers across many interoperating implementations.

Self-hosting resources: ActivityPub and ATProto comparison

Hosting a fediverse server is cheap particularly if you use something like GotoSocial is lightweight enough where one could host a server for one's family or friends on a device as light as a Raspberry Pi style form factor. It may require a lot of technical expertise, I may have many critiques of how it runs, but it's possible to host a *fully participating* fediverse node quite cheaply.

Now, you may see people say, running an ATProto node is fairly cheap! And this is because comparatively speaking, running a Personal Data Store is fairly cheap, because running a Personal Data Store is more akin to running a blog. But social networks are much more interactive than blogs, and in this way the rest of Bluesky's architecture is a lot more involved than a search engine: users expect real-time notifications and interactivity with other users. This is where the real architecture of Bluesky/ATProto comes in: Relays and AppViews.

So how challenging is it to run those? In July 2024, running a Relay on ATProto already required 1 terabyte of storage. But more alarmingly, just a four months later in November 2024, running a relay now requires approximately 5 terabytes of storage. That is a nearly 5x increase in *just four months*, and my guess is that by next month, we'll see that doubled to at least ten terabytes due to the massive switchover to Bluesky which has happened post-election. As Bluesky grows in popularity, so does the rate of growth of the expected resources to host a meaningfully participating node.

"Message passing" vs "shared heap" architectures

The best way to understand the reason for this difference in hosting requirements is to understand the underlying architecture of these systems. ActivityPub follows an **message passing** architecture (utilizing publish-subscribe architecture prominently for most "subscription" oriented uses), the same as email, XMPP, and so on. A message is addressed, and then delivered to recipients. (Actually a more fully peer-to-peer system would deliver more directly; all of email, XMPP, ActivityPub and so on use a client-server architecture, so there is a particular server which tends to operate on behalf of a particular user. See comments on the fediverse later in this article for how things can be moved more peer-to-peer.) This turns out to be pretty efficient; if only users on five servers need to know about a message, out of tens of thousands of servers, only those five servers will be contacted. Until recently, every system I knew of described as federated used a message passing architecture, to the degree where I and others assumed that federation *implied* a message passing architecture, because achieving the architectural goal of many independent nodes cooperating to produce a unified whole seemed to imply this was necessary for efficiency of a substantially sized network. If Alyssa wants to write a piece of mail to Ben, she can send it directly to Ben, and it can arrive at Ben's house. If Ben wants to reply, Ben can reply directly to Alyssa. Your intuitions about email apply exactly here, because that's effectively

what this design is.

Bluesky does not utilize message passing, and instead operates in what I call a **shared heap** architecture. In a shared heap architecture, instead of delivering mail to someone's house (or, in a client-to-server architecture as most non p2p mailing lists are, at least their apartment's mail room), letters which may be interesting all are dumped at a post office (called a "relay") directly. From there it's the responsibility of interested parties to show up and filter through the mail to see what's interesting to them. This means there is *no directed delivery*; if you want to see replies which are relevant to your messages, you (or someone operating on behalf of you) had better sort through and know about *every possible message* to find out what messages could be a reply.

It is curious then that the reason for not taking a message passing architecture such as ActivityPub as the foundation for Bluesky and ATProto is often described as wanting users to not have the experience of seeing a reply thread containing missed messages. From the AT Protocol paper:

> The distinction between servers in Mastodon introduces complexity for users that does not exist in centralized services. For example, a user viewing a thread of replies in the web interface of one server may see a different set of replies compared to viewing the same thread on another server, because a server only shows those replies that it knows about.

This is particularly curious because experiencing missed messages is a frequent complaint of other shared heap architecture designs such as Secure Scuttlebutt and Nostr, where missing message replies are even *more* common than on ActivityPub and other message-passing federated architectures. (Both Secure Scuttlebutt and Nostr take steps so you don't need to necessarily fetch everything; in SSB you fetch the feeds of your friends and 3 degrees removed from your friends from the hubs you use, and anything else you simply don't see. In Nostr you simply "embrace the chaos" of only grabbing the information from hubs you use, and hubs don't try to fetch all information.) For instance, if Ben replies to Alyssa's message in one of these systems but does not leave the reply message in the relay which Alyssa pulls from, Alyssa would never see Ben's reply. If multiple relays were to exist in Bluesky, this same problem would presumably occur, so how does Bluesky solve this?

The answer is: Bluesky solves this problem via centralization. Since there is really just one very large relay which everyone is expected to participate in, this relay has a god's-eye knowledge base. Entities which sort through mail and relevant replies for users are AppViews, which pull from the relay and also have a god's-eye knowledge base, and also do filtering. So too do any other number of services which participate in the network: they must operate at the level of gods rather than mortals.

The reality of the fediverse today is that due to the complexity of hosting an instance, many users join nodes hosted by either a friend or a larger group, but there are still many nodes on the network. As mentioned earlier, this is closer to being an apartment building than a house, but the ideal version of decentralization is that everyone self-hosts, and from a resource perspective, this is perfectly possible to do. It would be possible, for mere tens of dollars, for everyone to get a cheap computer and self-host something like GotoSocial and (ignoring the challenges of firewalls and ISPs frowning upon self-hosting at home these days) from an architectural perspective, it's certainly possible. The primary challenge preventing this future is in the technical difficulty of hosting these services presently (and the

way the internet has generally become hostile to home-hosting, but shared hosting for this level of service is also still relatively cheap for individuals). The ideal version of decentralization is, from a message transmission perspective (we shall look at other aspects later), fully possible.

The physical world equivalent for a fully decentralized fediverse then is that every user sends mail to every other user's house, as needed, similar to how sending letters works in the physical world. This is decidedly *not* the case with a fully decentralized ATProto. The physical world equivalent would be that every user had their own house at which they *stored a copy of every piece of mail delivered to every other user at their house*.

If this sounds infeasible to do in our metaphorical domestic environment, that's because it is. A world of full self-hosting is not possible with Bluesky. In fact, it is worse than the storage requirements, because the message delivery requirements become quadratic at the scale of full decentralization: to send a message to one user is to send a message to all. Rather than writing one letter, a copy of that letter must be made and delivered to every person on earth.

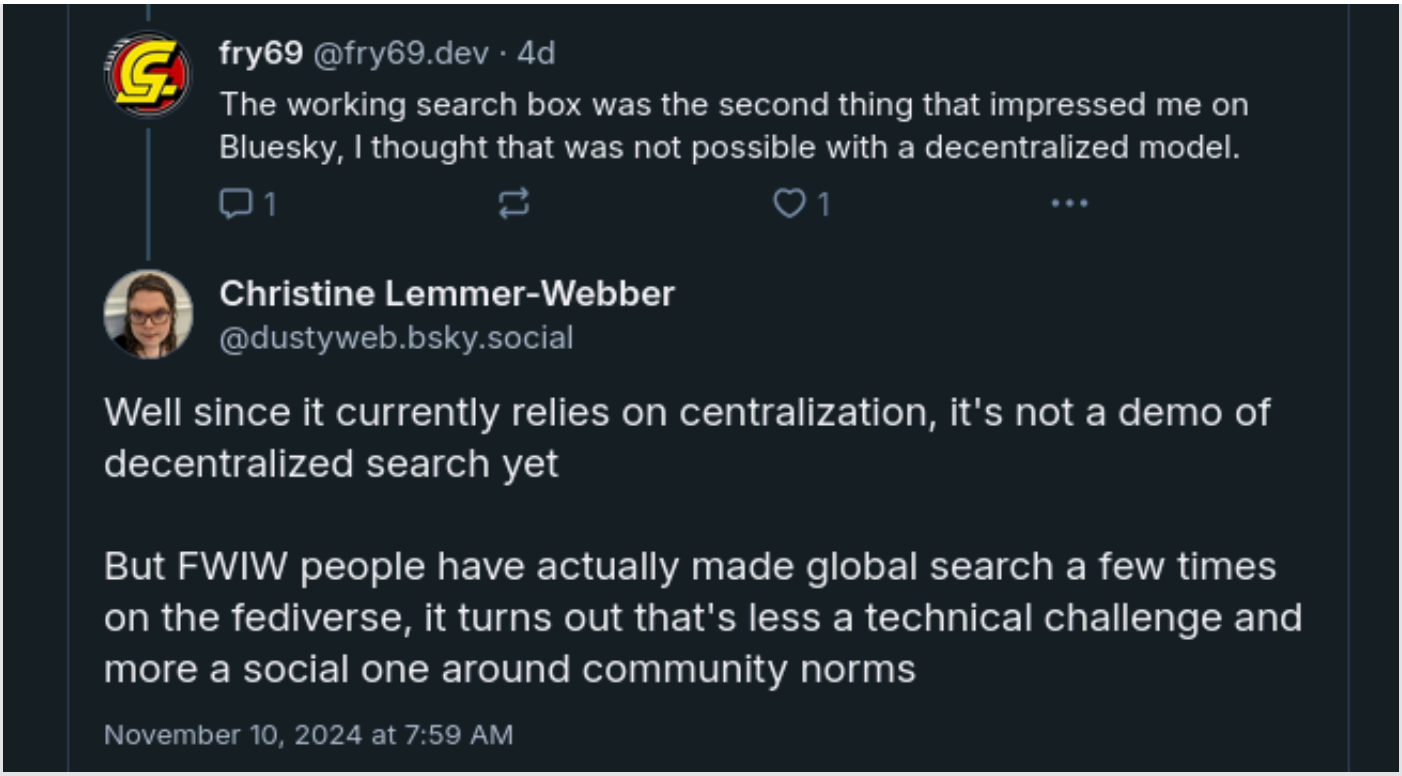
The costs of decentralizing ATProto

Bluesky's architecture documentation does acknowledge this, to some degree:

> The federation architecture allows anyone to host a Relay, though it’s a fairly resource-demanding service. In all likelihood, there may be a few large full-network providers, and then a long tail of partial-network providers. Small bespoke Relays could also service tightly or well-defined slices of the network, like a specific new application or a small community. -- Federation Architecture Overview (Bluesky blog)

What is not mentioned is that any smaller bespoke relays would have a *greater* problem with missing message replies than a directed message-passing architecture has. If larger nodes are gods, then I suppose smaller nodes are demi-gods, from which one could say that only truly fully and complete gods can participate meaningfully in the network.

In the meanwhile, many users of Bluesky seem to be operating under the impression that things are more decentralized than they are:

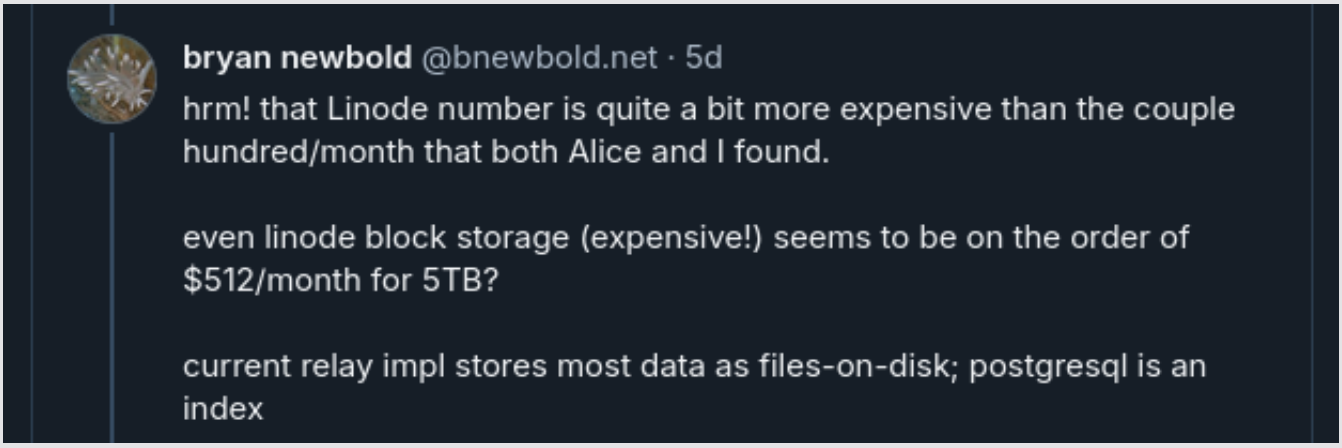


Part of the concern I have with Bluesky presently is thus that people are gaining the impression that it's a decentralized system in ways that it is not. There are multiple ways this could end up being a problem for the decentralized world; one irritating way is that people might believe there's an "easy decentralized way to do things" that Bluesky has discovered which isn't actually that at all, and another is that Bluesky could collapse at some point and that people might walk away with the impression of "oh well, we tried decentralization and that didn't work... remember Bluesky?"

But perhaps we should look at making Bluesky more decentralized by adding more meaningfully fully participating nodes to it again. How much would that cost today, and how much will that cost in the future?

Again, returning to alice's previously mentioned blogpost, the most recent time that costs of running a Bluesky relay were calculated (which does *not* also include the costs of running an AppView node or any other critical components), just looking at storage, the amount required was 5 terabytes of storage. The first glance thing I did was to look up, if you were going to pick up a complete shared hosting server configured with that storage size, how much would that be on Linode, as just a common example of a shared hosting provider? At first glance, this appeared to end up around $55k/year, just to host the last estimate of a current relay:

Bryan Newbold pointed out that this was fairly expensive and that there were cheaper options even on Linode using Linode's block storage options (though this doesn't account for still needing a database in addition):



But as both alice and Bryan Newbold have pointed out, using a dedicated server would be much cheaper. Unfortunately, this solution only scales for so long. In Bryan's previous article on running a relay, costs were calculated for one terabyte, and the server came to $152/month plus a one-time setup fee of $92. Cheap! However, the network is clearly growing and already exceeds that size, so let's take the same bare metal server and see how much storage we can add and how expensive this will get:

That's nearly 5x the expense to move to what it looks like the expenses for running a relay will be *very soon*. And this is for a single server that isn't being used, as we say, "in anger" against an actual userbase, of which the expenses of hosting such a thing are unknown, because nobody is really using it. We are now hitting the limits of a dedicated server regardless, so one will *have to* move towards more abstracted and clustered storage and indexing mechanisms past this point to keep the network running (unless disk manufacturers surprise us all with an enormous leap in capacity which is rolled out in the very short term future).

And that is *just for storage* running *without backups* or any of the other things one would need to keep such a thing going, including bandwidth and CPU cycles and so on and so forth. A single machine does not look like it can be a viable solution for very long, so pointing to dedicated servers which can currently handle an entire relay (when not actually relied upon by any number of users) isn't particularly convincing to me.

There is also the *legal liability* that one is taking on by effectively hosting the equivalent of *all of Twitter*! While Bluesky/ATProto *does* provide multiple filtering techniques which are very interesting, the relay does need to be in the business of identifying what content is not safe to use:

> [...] the Relay performs some initial data cleaning (discarding malformed updates, filtering out illegal content and high-volume spam) -- Bluesky and the AT Protocol: Usable Decentralized Social Media

The likely answer to this is that there will always have to be a large corporation at the heart of Bluesky/ATProto, and the network will have to rely on that corporation to do the work of abuse mitigation, particularly in terms of illegal content and spam. This may be a good enough solution for Bluesky's purposes, but on the economics alone it's going to be a centralized system that relies on trusting centralized authorities.

Everything is public, including who you block

You may be reading so far at this point and be wondering: so far I have only analyzed Bluesky from the perspective of public content. What about private or semi-private content? How does Bluesky provide its various services of filtering and labeling and so on in such an environment, and how does Bluesky know which messages are sent in reply to your messages with limited or entirely private messages?

The answer is that Bluesky and ATProto have no design for this at present, and most of the architectural assumptions *assume* public messages only. Now this could change of course, but everything within Bluesky's current literature and architecture assume public-only content. In fact, even blocks are public information:

> [...] for example, if one user has blocked another, and one of the users' repositories contains a record of an interaction that should not have been allowed due to the block, then the App View drops that interaction so that nobody can see it in the client apps. This behavior is consistent with how blocking works on Twitter/X, and it is also the reason why blocks are public records in Bluesky: every protocol-conforming App View needs to know who is blocking who in order to enforce the block. -- Bluesky and the AT Protocol: Usable Decentralized Social Media

I'm not sure this behavior is consistent after all with how blocking works on X-Twitter; it was not my understanding that blocking someone would be public information. But blocks are indeed public information on Bluesky, and anyone can query who is blocking or being blocked by anyone. It is true that looking at a blocking account from a blocked account on most social media systems or observing the results of interactions can *reveal* information about who is blocked, but this is not the same as this

being *openly queryable information*. There is a big difference between "you can look at someone's post and see who is being blocked" to "you can query the network for every person who is blocking or is blocked by JK Rowling".

I found this very surprising; in ActivityPub's development, I remember a conversation between Amy Guy and myself where we decided it was very important to not deliver Block activities between servers. We encoded this in ActivityPub's specification thusly:

> The Block activity is used to indicate that the posting actor does not want another actor (defined in the object property) to be able to interact with objects posted by the actor posting the Block activity. The server SHOULD prevent the blocked user from interacting with any object posted by the actor.
>
> Servers SHOULD NOT deliver Block Activities to their object. -- ActivityPub

The reason for this is very simple: we have seen people who utilize blocklists be retaliated against for blocking someone who is angry about being blocked. It was our opinion that sharing such information could result in harassment. (Last I checked, Mastodon provides the user with the choice of whether or not to send a "report" about a block to the offending instance so that moderators of that server can notice a problematic user and take action, but delivering such information is not required.)

That said, to Bluesky's credit, this is an issue that is being openly considered. There is an open issue to consider whether or not private blocks are possible. Which does lead to a point, despite my many critiques here: it is true that even many of the things I have talked about could be changed and evaluated in the future. But nonetheless, in many ways I consider the decision to have blocks be publicly queryable to be an example of *emergent behavior from initial decisions*... early architectural decisions can have long-standing architectural results, and while many things can be changed, some things are particularly difficult to change form an initial starting point.

Direct messages are fully centralized

But you may notice! Bluesky provides direct messages! So surely not *all* information is publicly available, because otherwise else direct messages would simply not work! So how do direct messages work in Bluesky?

The answer, if you guessed it, is centralization. All direct messages, no matter what your Personal Data Store is, no matter what your *relay* is, go through Bluesky, the company.

If you find this shocking, so did I, but then again, this information was publicly available even when direct messages were announced. Bluesky's direct messages are also not end-to-end encrypted, and don't use any particular kind of protocol which is amenable to decentralization or federation.

But perhaps we should back up... am I being too harsh? After all, while the fediverse works like email (actually, ActivityPub's architecture is (contrary to many users' expectations since Mastodon is also a Twitter clone and is how most people experience the fediverse), designed for direct communication *first*

and foremost... public communication is clearly supported, but the default and simplest case is direct individual or group messaging), it's also "about as private as email". Which is to say, not private enough for many kinds of security concerns these days: your administrator can read your DMs but hopefully does not in the general case, but messages are not end-to-end encrypted at present. But I can know my administrator personally, and thus the trust dynamics are often not the same.

A feature complete Twitter ASAP

So direct messages on Bluesky are centralized, and while Bluesky does say so in their blogposts (past the point at which most people have read), most users I have talked to have assumed they worked the same way that the rest of ATProto works. Why would Bluesky roll out a direct message system that they have acknowledged is not the long term direct message system they would like long term? (Though I am also still puzzled... why they didn't at least use XMPP?)

The presumable answer is: Bluesky wanted to provide a feature-complete platform from the perspective of a user who is looking for an exit from Twitter *now*. And this is honestly a fair decision to make in many ways, since as I have said previously, while I don't see Bluesky as a very good decentralized Twitter, I do see it as a good replacement for Twitter, which is what most users are looking for immediately. But the lack of understanding of these detail by many users and media coverage is a bit maddening from someone like myself who actually really does look at and care about decentralization, and I know it's a bit maddening to much of the fediverse too.

But again: to many users, this doesn't matter. What many users fleeing X-Twitter *right now* care about is a replacement for Twitter. For that matter, if you're coming from Twitter, whether or not Bluesky is truly decentralized, it certainly seems *more decentralized than Twitter*, the same way that Twitter may seem more decentralized than cable news. Things are sometimes more decentralized in degrees, and I certainly think the fediverse could be more decentralized than it is. (More, again, on this later.) But in all ways related to the distribution of power, Bluesky's technology is notably much less distributed than existing and prominent decentralized technology *in deployment today*.

Bluesky is centralized, but "credible exit" is a worthy pursuit

In many places, Bluesky acknowledges that it is more centralized than other alternatives in its own writing. From its own paper:

> Even though the majority of Bluesky services are currently operated by a single company, we nevertheless consider the system to be decentralized because it provides *credible exit*: if Bluesky Social PBC goes out of business or loses users' trust, other providers can step in to provide an equivalent service using the same dataset and the same protocols. -- Bluesky and the AT Protocol: Usable Decentralized Social Media

It is *not a bad choice* for Bluesky to be focused on providing an alternative to X-Twitter for those who miss Twitter-of-yore and are immediately looking for an offboarding from an abusive environment. I understand and support this effort! Bluesky does use several decentralization tricks which may lend

themselves more towards its self-stated goal of "credible exit". But these do not make Bluesky decentralized, which it is not within any reasonable metric of the power dynamics we have of decentralized protocols which *exist today*, and it does not use federation in any way that resembles the way that technical term has been used within decentralized social networking efforts. (I have heard the term "federation-washing" used to describe the goalpost-moving involved here, and I'm sympathetic to that phrase personally.)

In my opinion, this should actually be the way Bluesky brands itself, which I believe would be more honest: an open architecture (that's fair to say!) with the possibility of credible exit. This would be more accurate and reflect better what is provided to users.

ATProto's portable identity challenges

Bluesky's credible exit claims rely both on content addressing but also on its use of Decentralized Identifiers (DIDs) for account migration. This is certainly a good goal, and account migration support is something we should see more broadly (including on the fediverse).

However, there are several major problems:

- ATProto supports two DID methods, `did:web` and `did:plc`, which (despite the "D" in "DID"!) are both centralized.
- The cyclic relationship between ATProto's approach to DIDs and DNS causes problems which undercuts the utility of DIDs (this is addressable, but it's not clear to me that there will be interest).
- Even if a user wishes to switch away from Bluesky's infrastructure, Bluesky probably has effective permanent control over that user's identity destiny, removing the reassurance that one need not trust Bluesky as a corporation in the long term.
- Several other concerning details about `did:plc` generated DIDs.

Some history: Decentralized Identifiers (including the centralized ones)

First, some background. The DID Core spec spec is really more of an abstract interface on which specific "DID methods" can be implemented. What a DID method *mostly* provides (it does some other things too, but but as importantly) is a mechanism by which cryptographic public keys can be registered, retrieved, and rotated (though rotation ability is not strictly a requirement; `did:key` cannot be rotated, for instance).

Surprisingly, despite the name and the original intents of the DID architects when DIDs were being envisioned, that a DID method be decentralized is *not* a requirement.

I said surprisingly, so: are you surprised? I used to be very active in this space (I never worked on the DID spec directly, but when I worked at Digital Bazaar I was sometimes active in the Verifiable Credentials calls, and I did co-author a spec also developed in that space for linked data capabilities, zcap-ld). I say this because I want to provide context that allowing decentralized identifier methods to not be decentralized at all was a *debate*.

I think I had stepped out of the group by then, but I remember talking to colleagues about `did:web`; it was the first real argument for centralized DIDs. But wait, didn't I previously say that the web was open and decentralized? Yes, but the naming+encryption system the web runs on top of is not: DNS+TLS relies on trusting ICANN on down and TLS Certificate Authorities, both of which are centralized approaches. My understanding of the justification for `did:web` was primarily that everyone would have a trivial DID method that would allow all conforming implementations to easily pass the DID test suite. I was in the camp that blessing DIDs which were centralized as "Decentralized Identifiers" would lead to decentralization-washing, and that's exactly what's happening here.

There's another silly thing about `did:web`: there's really not a real *reason* for `did:web`, since all `did:web` does is effectively get rewritten via a trivial regular expression to an `https:` link, and you could just use that very `https:` link instead of `did:web` and serve the same information in any relevant context. But the thing is, people hear that `did:web` is a decentralized identifier, so they assume it must be, even though again, `did:web` never gets us past the centralization challenges inherent in DNS+TLS, it simply uses them! Unfortunately, due to the name, many people think `did:web` provides a more robust layer of security than simply retrieving a key over `https:` does. I'm here to tell you that it doesn't, because that's exactly what `did:web` does anyway.

did:plc, the "placeholder" DID

But Bluesky has developed its own DID method, `did:plc`. Today, `did:plc` stands for "Public Ledger of Credentials", however it originally stood for "Placeholder DIDs", with the hope of replacing them with something else later. The way that `did:plc` works is that Bluesky hosts a web service from which one can register, retrieve, and rotate keys (and other associated DID document information). However, this ledger is centrally controlled by Bluesky.

This aspect of centralization, on its own, doesn't bother me as much as a reader might think! For one thing, if all works right, Bluesky can only deny rotations or retrieval of `did:plc` documents, but since future updates to the document are signed by the original DID document's key, Bluesky shouldn't (hm, we'll return to "shouldn't" in a second) be able to forge future updates to said document. And Bluesky's developers are very open to acknowledging that `did:plc` is centralized, and have expressed some interest in moving to something else, or improving its governance so that the organization is controlled by another more neutral org (Paul Frazee in particular suggests that one solution could even be to move to an ICANN-like organization).

However, there are other aspects to `did:plc` which seem strange. For one thing, `did:plc` documents' identifiers are, as best as I can tell, `sha256` hashes of the DID document truncated to 15 bytes (120 bits) of entropy. This seems like a strange decision to me; it does mean that `did:plc` URIs fit in 32 characters (8 characters for `did:plc:`, 20 characters for the truncated hash) which I guess is a nice round "computer'y" number but why throw away all that valuable entropy? For aesthetics? DID identifiers aren't meant to be read by humans, they should be encapsulated, so this is a strange decision to me. (I'm admittedly an amateur when it comes to cryptography, but I'm old enough to remember Debian getting into trouble over using PGP short ids.) (Also choosing `sha256` over `sha256d`, there's maybe the question of length extension attacks, but I suppose the parsing of the document means this is

maybe not a problem, I'm not sure.) It's just strange decisions. But again, `did:plc` was *meant* to be a placeholder. The problem, of course, is that this placeholder is now the basis of identifiers for many users who have already joined the system *today*.

I have not spent much time auditing `did:plc` myself, just reading high level details and wondering, but there are some other strange details which can be found in the blogpost Hijacking Bluesky Identities with a Malleable Deputy. From that post, the most alarming is:

> However, there's one other factor that raises this from "a curiosity" to "a big problem": bsky.social uses the same rotationKeys for every account. This is an eyebrow-raising decision on its own; apparently the cloud HSM product they use does billing per key, so it would be prohibitively expensive to give each user their own. (I hear they're planning on transitioning from "cloud" to on-premise hosting, so maybe they'll get the chance to give each user their own keypair then?)

I have not looked, but I would assume this is not the case anymore, but I find it surprising and alarming that reusing the same key per user was *ever* the case. It feels like this flies in the face of the fundamental goals one would have around building a DID system and it is difficult for me to fathom how such a decision could ever have been made.

But there is a bigger problem regarding centralization and `did:plc`:

> In principle, the cryptographic keys for signing repository updates and DID document updates can be held directly on the user's devices, e.g. using a cryptocurrency wallet, in order to minimize trust in servers. However, we believe that such manual key management is not appropriate for most users, since there is a significant risk of the keys being compromised or lost.
>
> The Bluesky PDSes therefore hold these signing keys custodially on behalf of users, and users log in to their home PDS via username and password. This provides a familiar user experience to users, and enables standard features such as password reset by email. The AT Protocol does not make any assumptions about how PDSes authenticate their users; other PDS operators are free to use different methods, including user-managed keys.
>
> -- Bluesky and the AT Protocol: Usable Decentralized Social Media

I am sympathetic to this position: it is true that key management for users is an incredibly hard user experience and coordination problem, so this decision might not even be wrong. But the more concerning thing is that users are being told that if they want to walk away from Bluesky the company, they can at any time! After all, it's possible for a user to change both their key and the location it points to at a future time. However, what does that look like for a user who trusts Bluesky *today* but does not trust Bluesky *tomorrow*? The truth of the matter is: Bluesky controls users' keys, and therefore even if users "move away" they must trust Bluesky to perform this move on their behalf. And *even if* Bluesky delegates authority to that user to control their identity information in the future, there is still a problem in that Bluesky will *always* have control over that user's key, and thus their identity future.

Zooko's triangle, petnames, and a cyclic dependency between DNS and DIDs on Bluesky

Alas, this is not the end of the identity challenges, because there is a fundamental challenge (or set of challenges) around the way Bluesky binds a user's DID to the *handle* of that user which the user sees. Zooko's triangle tells us that a decentralized and globally unique name cannot also be human meaningful, and indeed Decentralized Identifiers which are actually decentralized cannot be. Ignoring, again, that `did:plc` is not actually decentralized, it is a non-human-meaningful identifier. So what's the solution? How do we provide a human meaningful name that the user *can* understand?

I strongly believe that the right answer is a Petname System, which allows for local human meaning to globally non-human-meaningful names. However, the discussion of why I believe that is the right approach and how to accomplish it is too large for this writeup; I will only say that Ink and Switch did a great petnames demo and (while not particularly polished) there are more ideas one can read about in a prototype Spritely put together. But admittedly, petname systems have not been widely deployed to this date, and so the UX challenges around them are not fully solved.

Perhaps because of this reason, Bluesky did *not* adopt a petname system for users' handles and instead adopts something much more familiar to users today: domain names! Every user on Bluesky's handle is effectively its own domain name. But users can also change their handle by associating with a different domain name later!

In one way, Bluesky is doing the right thing here by taking a human meaningful name and mapping to the less human meaningful identifier, which is what you would want to do if using an actually decentralized Decentralized Identifier, so this appears to be a good sign for the future. But wait... let's take a look a bit deeper, and the situation seems to bet a bit murkier.

ATProto uses the `alsoKnownAs` field on the DID document itself for the DID to proclaim what URLs it is associated with. It is not really possible for DID documents to be able to verify this information on their own since verifying such things is a "live" operation, and the `did:plc` method's documentation correctly identifies this:

> The PLC server does not cross-validate `alsoKnownAs` or `service` entries in operations. This means that any DID can "claim" to have any identity, or to have an active account with any service (identified by URL). This data should not be trusted without bi-directionally verification, for example using handle resolution.

As such, it's the job of the rest of the ATProto consuming infrastructure (at every step of the process, if we are being robust, but less robustly we could choose to trust a source of incoming data) to verify whether or not a DID does indeed map to the handle it claims to.

But this is puzzling. Consider: the point of DIDs originally was to provide a decentralized path to identity, which DNS+TLS is decidedly not. But even in the case of `did:plc`, one must still rely on the *liveness* of the web to ensure that a handle and a DID document bidirectionally map to each other. So the problems of `did:web`, in effect, still exist for `did:plc` too.

But here are some other thorny questions: if at one point we verified that `did:plc:<blah>` mapped bidirectionally to `alyssa.example`, what happens if `https://alyssa.example` goes down temporarily or permanently? What if a new user claiming to represent `alyssa.example` shows up instead, and this seems to "check out"? How do we represent posts by the previous `alyssa.example` to the user? The current one? I don't know how to answer these questions... do you?

For users, DIDs don't really come into account: if `bsky.app` (and `bsky.social`) went down because Bluesky the company folded, it would be challenge for users to tell whether or not `alyssa.bsky.social` continues to represent Alyssa. It isn't clear to me how a sudden "null" mapping of identity to a domain that no longer exists should be represented to the user, and I'm not sure there is one, and at any rate as far as users are concerned, the DNS record of `alyssa.bsky.social` is the record for Alyssa, not the DID. A petname system solves these problems, Bluesky's user experience does not.

Can you credibly exit with your identity from a Bluesky takeover?

But in total, if a hostile company were to take over Bluesky, `did:plc` seems to not fare well:

- Bluesky controls most users' keys anyway, so can control their identity future regardless, including in terms of signed updates
- Most users are mapped to domains controlled as `*.bsky.social` subdomains, so Bluesky can remap those to different users, and this will be true with any "Personal Data Store" provider one might use too
- It's not clear how Ben would know that the person who used to be `alyssa.bsky.social` is now `alyssa.example` even if she migrated to her own domain without Ben reading previous interactions (in general, "sudden changes" in one's handle being the norm means that any domain update seems to be a ripe opportunity for phishing attacks anyway)
- Bluesky could block future `did:plc` document updates, and the proposed solution seems to be "another ICANN"

At any rate, in both `did:plc` and `did:web`, ICANN must literally be trusted, because domain names are what users know. But if the solution to `did:plc` is an ICANN-like entity, then I guess users must trust two ICANNs.

So at the moment, Bluesky's identity system is in no real way decentralized, but that's only part of the problems, as outlined above. Still, the fact that Bluesky is using the Decentralized Identifiers *interface* means that perhaps *actually* decentralized identity solutions can be layered on top. In the meanwhile, effectively everything bottoms out to trusting the domain name system, and for that matter, trusting Bluesky. Users effectively trust domains, so in the end, we might as well be just retrieving a key from a particular domain. While continuity of identity is in theory possible from one domain onward, this is not done in a useful way that users can understand; a shift from one domain to the next is a complete shift of the handle one is known by, potentially even opening a phishing attack vector. Petname systems could address this issue, but integrating them at this point would be a major shift in how users perceive of the network, and it seems unlikely that downplaying the role of domains is something Bluesky as an organization will be motivated to do since selling domains is currently a Bluesky business strategy.

What should the fediverse do?

I promised a critique of the fediverse, and the reality is that I have been doing critiques of the fediverse the entire time since ActivityPub has been released. It is *not* the case that I believe that ActivityPub-as-deployed is an end-all-be-all solution. Quite the opposite.

The most succinct version of what I think the fediverse/ActivityPub should do is actually in ActivityPub + OCaps which was, of all things, a proposal about what Bluesky might be which I co-submitted with Jay Graber when Twitter was still evaluating Bluesky proposals. I am not bringing this up because I think it was the proposal which should have been chosen; I think Bluesky had directive based needs around scaling quickly and I ultimately think both that Jay Graber was the correct choice to lead Bluesky and that it also made most sense to run Spritely as a separate organization, because Spritely needed to spend its first few years focused on research fundamentals. However, I think the proposal really is the best writeup I know of on how to transform the fediverse from where it is to where it should be:

- Answer the missing authorization part of the ActivityPub spec by integrating capability security throughout. A writeup about why this is important, which includes significant critique of the fediverse, can be found in my OCapPub writeup.
- Integrate decentralized / content addressed storage (ideally with an encryption layer a-la Tahoe LAFS) for posts which can survive server shutdown (like in the Golem demo I put together).
- Use mutable files within decentralized storage (Tahoe and IPFS are both examples of immutable systems which layer mutable files on top) to permit portable identity. When a user wishes to switch servers, point the `inbox` property at a new endpoint.
- Use a petname system to make the portable / decentralized identifiers (not DIDs necessarily) more human-understandable and to make the system more robust against phishing attacks generally.
- More anti-spam / anti-harassment tooling built on top of capability security foundations.
- Improve privacy by bringing in End-to-End Encryption (there has been some work on this but I can't say I have followed closely at all).

Some of these tasks are quite feasible for the fediverse to pick up today: the content-addressed storage and the portable identity stuff I think would be a major thing to introduce into the system but would be quite doable and would give the fediverse properties of surviving nodes going down better.

Some of the rest might be more challenging, though these aren't new directions for me to be pushing in the direction of. I put together a document called OCapPub a few years ago to present an alternative vision for how the fediverse should go.

However at the time I found that many fediverse implementers didn't really understand what I was pushing for, and for that matter, didn't really understand how they could possibly implement this on top of web 2.0 frameworks like, say, Ruby on Rails. Fair enough, and the work we've been doing at Spritely is in many ways what I think is the answer: we're designing things such that capability secure, distributed systems are what falls out of Spritely's tech when you write a program in it.

Blaine Cook said that the correct version of ActivityPub and the correct version of ATProto are "the same picture" at one point. This is true insofar as I believe addressing the serious issues of both converges on a shared direction: the fediverse needs to adopt content addressing and portable identity (criticisms of Bluesky's approach to this latter one aside at the moment), Bluesky needs to support a messaging architecture such that participating meaningfully and fully in decentralization does not mean needing to host everything (adopting such a solution will probably mean adopting something that ultimately looks a lot like ActivityPub). And of course, I think both need to move towards supporting privacy and stronger collaboration tools with capability security. While others have argued that these are "different approaches" -- and perhaps this is because I am overly ambitious in what I think decentralized networks should do -- to me this is because both are not being all they could be. Instead to me it feels that there is a "fixed point" of resolving these issues to iterate towards.

But perhaps that's too ambitious to suggest taking on for either camp. And maybe it doesn't matter insofar as the real lessons of Worse is Better is that both first mover advantage on a quicker and popular solution outpaces the ability to deliver a more correct and robust position, and entrenches the less ideal system. It can be really challenging for a system that is in place to change itself from its present position, which is a bit depressing.

This last paragraph applies to both Bluesky and the fediverse, but again, the fediverse is currently actually decentralized, and from my analysis, if there was willingness to take on the work, the gap of moving towards resolving content addressing and portable identity at least are not so large architecturally. But I'm not sure there's interest or not. Maybe there will be so more now.

But for me, I am more interested in "secure collaboration" these days than anything else, and that's where my work continues at Spritely. We are working our ways towards our own answer for social systems, but we aren't there yet. And so in the meanwhile, I feel like I am sounding incredibly grouchy about all of the above, but really, it's just that I think these really are important things to get right. Conway's law applies in both directions: a technical system reflects the communication and social structure of those who build it, but the communication and social structures that we have available to ourselves are informed by what technology is available to ourselves.

Regardless, that's enough of my Cassandra complex about the fediverse and otherwise. Perhaps, at the risk of making me sound grouchy and bitter (I'm not, I hope, I usually am just focused on building the things I think *are* heading in the right direction) you have seen that I am not lacking in fediverse critiques also. But one thing I think *is* true: the fediverse *is* decentralized and *is* federated. My critiques of Bluesky as not achieving either such thing still hold. So let us move onward to: can such concerns be addressed in time? Or, at least, can a "credible exit" be made possible?

Preparing for the organization as a future adversary

One interesting thing about Bluesky is that its team uses a very self-reflective phrase: "the organization is a future adversary" (here are a couple of examples). This is a very self-aware phrase that one rarely sees in an organization and is thus commendable. In many ways it reminds me of Google saying "Don't Be Evil", which was an internal rallying cry which, while perhaps never fully sincere, gave a lot of

opportunity to challenge decisions internally and externally and hold Google to account to some degree. While questionable things happened while the phrase was in place, when the term was decommissioned, things at Google really did seem to be getting a lot worse.

Bluesky is a Public Benefit Corporation, which means that profit is not its only motive; Bluesky has also declared its work as being for the public good in addition to seeking profit. I can say with confidence that many of the people working at Bluesky fully believe this and, as I have emphasized earlier in the article, I think the people working at Bluesky are good and earnest about the goals of Bluesky.

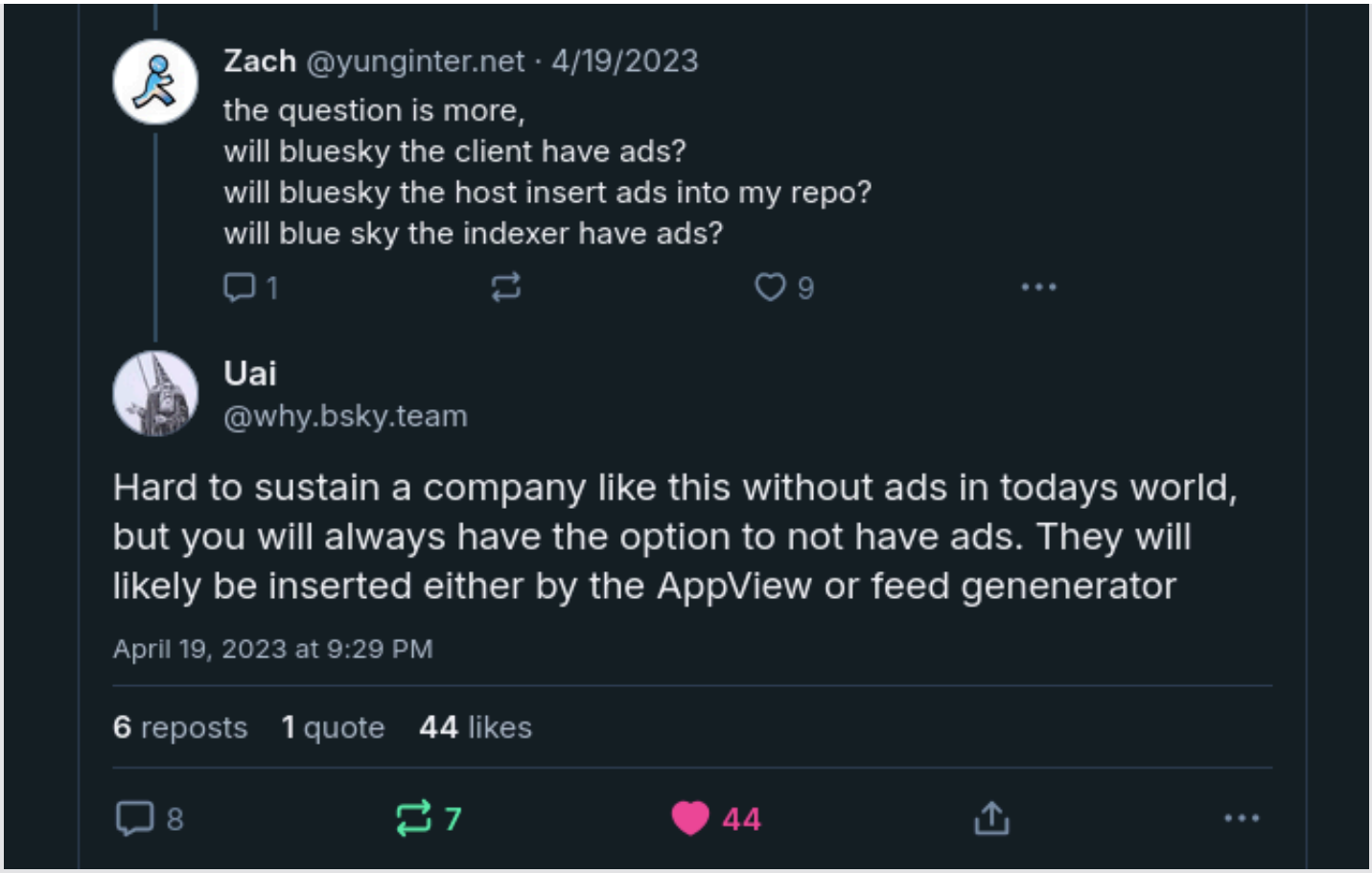
So "The organization is a future adversary" is thus a prescient phrase for the moment. In addition to its launching funds from Twitter (which my understanding is Bluesky received with few if any strings attached other than to carry out its stated work), Bluesky has raised two rounds of venture capital funding. I have respect for this insofar as I have done nearly every role possible in free and open source software orgs at some point or another, and fundraising is by far the hardest and most stressful of all of them. And when you're building an organization and building good people in, their future livelihood can really weigh on you. So I am glad to see Bluesky get funding, in this regard.

But venture capital is not a donation; investors want a return. I have many friends who have taken VC money, including some running decentralized social network orgs, and have seen an exciting and positive time early on and then have seen their organization clawed away from them by investors looking for returns. I'm not judging the choice to take venture capital negatively, just acknowledging: this is the state of affairs, and we should recognize it.

And by using the phrase "the organization is a future adversary", Bluesky *has* acknowledged it. The right next step then is to start planning all work to survive this situation of course.

I've analyzed previously in the document the challenges Bluesky has in achieving meaningful decentralization or federation. Bluesky now has much bigger pressures than decentralization, namely to satisfy the massive scale of users who wish to flock to the platform now, to satisfy investors which will increasingly be interested in whether or not they can see a return, and to achieve enough income to keep their staff and servers going. Rearchitecting towards meaningful decentralization will be a big pivot and will likely introduce many of the problems that Bluesky has touted their platform as not having that other decentralized platforms have.

There are early signs that Bluesky the company is already considering or exploring features that only make sense in a centralized context. Direct messages were discussed previously in this document, but with the announcement of premium accounts, it will be interesting to see what happens. Premium accounts would be possible to handle in a fully decentralized system: higher quality video uploads makes sense. What becomes more uncertain is what happens when a self-hosted PDS user uploads their own higher quality videos, will those be mirrored onto Bluesky's CDN in higher quality as well? Likewise, ads seem likely to be coming to Bluesky:



A common way to make premium accounts more valuable is to make them ad-free. But if Bluesky is sufficiently decentralized and its filtering and labeling tools work as described, it will be trivial for users to set up filters which remove ads from the stream. Traditionally when investors realize users are doing this and removing a revenue stream, that is the point at which they start pressuring hard on enshittification and removing things like public access to APIs, etc. What will happen in Bluesky's case?

Here is where "credible exit" really is the right term for Bluesky's architectural goals. Rearchitecting towards meaningful decentralization and federation is a massive overhaul of Bluesky's infrastructure, but providing "credible exit" is not. It is my opinion that leaning into "credible exit" is the best thing that Bluesky can do: perhaps a large corporation or two always have to sit at the center of Bluesky, but perhaps also it will be possible for people to leave.

Conclusions

Bluesky is built by good people who care, and it is providing something that people desperately want and need. If you are looking for a Twitter replacement, you can find it in Bluesky today.

However, I stand by my assertions that Bluesky is not meaningfully decentralized and that it is certainly not federated according to any technical definition of federation we have had in a decentralized social network context previously. To claim that Bluesky is decentralized or federated in its current form moves the goalposts of both of those terms, which I find unacceptable.

However, "credible exit" is a reasonable term to describe what Bluesky is aiming for. It is Bluesky's

term, and I think Bluesky should embrace that term fully in all contexts and work that they can.

Tags: bluesky decentralization federation