Bluesky and the AT Protocol: Usable Decentralized Social Media

Martin Kleppmann
martin.kleppmann@cst.cam.ac.uk
University of Cambridge
Cambridge, UK

Paul Frazee
Jake Gold
Jay Graber
Daniel Holmgren
Bluesky Social PBC
United States

Devin Ivy
Jeromy Johnson
Bryan Newbold
Jaz Volpert
Bluesky Social PBC
United States

ABSTRACT

Bluesky is a new social network built upon the AT Protocol, a decentralized foundation for public social media. It was launched in private beta in February 2023, and has grown to over 3 million registered users in the following year. In this paper we introduce the architecture of Bluesky and the AT Protocol, which is inspired by the web itself, but modernized to include streams of real-time updates and cryptographic authentication. We explain how the technical design of Bluesky is informed by our goals: to enable decentralization by having multiple interoperable providers for every part of the system; to make it easy for users to switch providers; to give users agency over the content they see; and to provide a simple user experience that does not burden users with complexity arising from the system's decentralized nature. The system's openness allows anybody to contribute to content moderation and community management, and we invite the research community to use Bluesky as a dataset and testing ground for new approaches in social media moderation.

1 INTRODUCTION

Over the last two decades, social media services have evolved from a fun curiosity into a cornerstone of civic life [5]. This development has been accompanied by increasing unease that mainstream "digital town squares", such as Twitter/X or Facebook, are under the control of a single corporation, and may change their policies on the whim of their leaders [62]. Their operations are opaque (e.g. regarding which content is recommended to users), and their users lack agency over their user experience. As a result, there has been increasing interest in decentralized social networks, of which the *fediverse* around the ActivityPub protocol [34] and the Mastodon software [39] is perhaps the best known (we review a selection of decentralized social networks in Section 4).

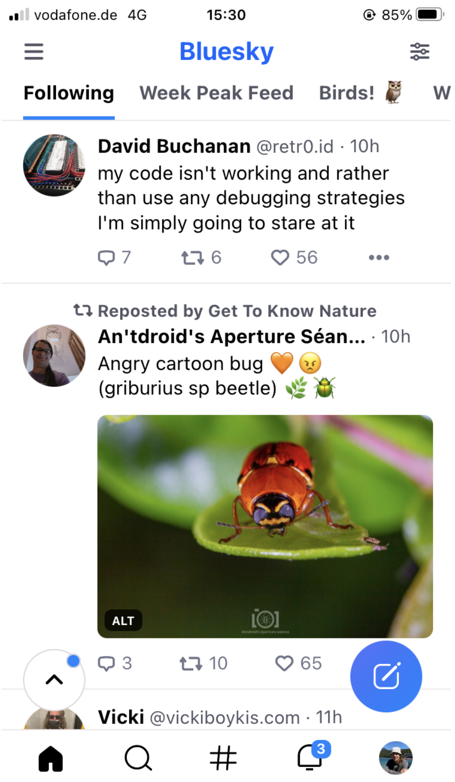
However, decentralization also introduces new challenges. For example, in the case of Mastodon, a user needs to choose a server when creating an account. This choice is significant because the server name becomes part of the username; migrating to another server implies changing username, and preserving one's followers during such a migration requires the cooperation of the old server. If a server is shut down without warning, accounts on that server cannot be recovered – a particular risk with volunteer-run servers. In principle, a user can host their own server, but only a small fraction of social media users have both the technical skills and the inclination to do so.

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 [2]. As another example, when viewing the web profile of an account on another server, clicking the "follow" button does not simply follow that account; instead, the user needs to enter the hostname of their own server and be redirected to a URL on their home server before they can follow the account. In our opinion, it is undesirable to burden users with such complexity arising from the federated architecture.

In this paper we introduce the *AT Protocol* (atproto), a decentralized foundation for social networking, and *Bluesky*, a Twitter-style social app built upon it. A core design goal of atproto and Bluesky is to enable a user experience of the same or better quality as centralized services, while being open and decentralized on a technical level. We introduce the user-facing features of Bluesky in Section 2, and in Section 3 we explain the underlying systems architecture. The AT Protocol is designed such that for every part of the system there are multiple competing operators providing interoperable services, making it easy to switch from one provider to another.

Decentralization alone is not able to solve some of the thorniest problems of social media, such as misinformation, harassment, and hate speech [46]. However, by opening up the internals of a service to contributors who are not employees of a particular company, decentralization can enable a marketplace of approaches to these problems [38]. For example, Bluesky allows anybody to run moderation services that make subjective decisions of selecting desirable content or flagging undesirable content, and users can choose which moderation services they want to subscribe to. Moderation services are decoupled from hosting providers, making it easy for users to switch moderation services until they find ones that match their preferences. Our hope is that this architectural openness enables communities to develop their own approaches to managing problematic content, independently of what any particular service operator implements [38].

For example, researchers wanting to identify disinformation campaigns can easily get access to all content being posted, the social graph, and user profiles on Bluesky. If they are able construct an algorithm to label suspected disinformation, they can publish their labels in real time, and users who wish to see those labels can enable them in their client software. One goal of this paper is to bring Bluesky and the AT Protocol to the attention of researchers working on such algorithms, and to invite them to use the rapidly growing dataset of Bluesky content as a basis for their work.



David Buchanan @retr0.id · 10h
my code isn't working and rather than use any debugging strategies I'm simply going to stare at it
7 6 56

Reposted by Get To Know Nature
An'tdroid's Aperture Séan... · 10h
Angry cartoon bug 🧡😠 (griburius sp beetle) 🌿🪲
ALT
3 10 65

Vicki @vickiboykis.com · 11h

Figure 1: Screenshot of the Bluesky home screen.

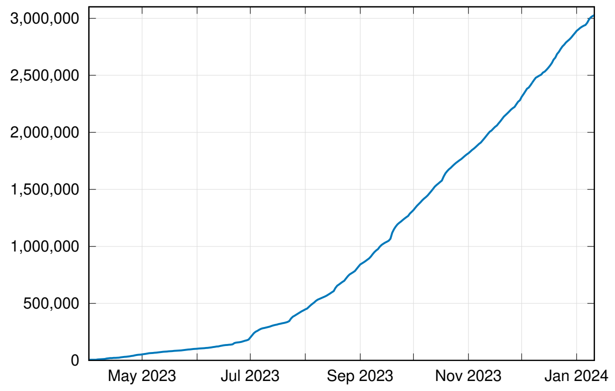


Figure 2: Number of registered users on Bluesky since April 2023.

2 THE BLUESKY SOCIAL APP

Bluesky presents itself to users as a straightforward microblogging application in the style of Twitter/X (see Figure 1). The "official" client app is available on iOS, Android, and the web; several independently developed client apps are also available, such as Graysky [42] and deck.blue [24]. Users can make public posts containing up to 300 characters of text, and up to four images, and they can interact with posts by replying, reposting, or liking. A user can also follow other users, and the default feed shows posts by accounts that the user is following in reverse chronological order. There are also alternative feeds that show content on various topics, without the user needing to follow the poster (see Section 2.3), which helps users discover each other.

Bluesky launched an invite-only beta release in February 2023, and has grown to over 3 million registered users in January 2024, as shown in Figure 2. Bluesky Social PBC (a public-benefit corporation) develops the official client app and operates the core services; the client and several server-side components are open source under the MIT license [8]. The protocols they use are defined by open specifications [7]. Several parts of the system, such as feed generators (Section 2.3) and various alternative clients [6] are developed and operated by independent third parties.

2.1 Moderation Features

Bluesky currently has the following moderation mechanisms (additional mechanisms are under discussion [14]):

Content filtering: Automated systems label potentially problematic content (such as images of a sexual or violent nature, posts promoting hate groups, or spam), and the app's preferences allow users to choose whether to show or hide content in each of these categories in their feeds.

Mute: A user can mute specific accounts or threads, which hides the muted content from their own feeds and notifications. The content continues to be visible to other users, and the target does not know that they were muted. A user can also publish a mutelist of accounts, and other users can subscribe to that list, which has the same effect as if they individually muted all of the accounts on the list.

Block: One user can block another, which prevents all future interactions (such as mentions, replies, or reposts) between those accounts in addition to muting. Similarly to mutelists, a user can also publish a list of accounts, and other users can block all accounts on that list by subscribing to it.

Interaction gating: A user who makes a post can restrict who is allowed to reply to it (anyone, anyone they follow, anyone mentioned in the post, and/or anyone on a particular list of accounts) [11].

Takedown: Users can report content that violates the terms of service to server operators, and the operators can take down violating media, posts, or accounts.

Custom feeds: While the aforementioned mechanisms provide negative moderation (helping users avoid content they do not want to see), feed generators (see Section 2.3) can actively select high-quality content.

2.2 User Handles

Like on Twitter/X, a Bluesky user has two names: the *display name* can be almost any string, and the *handle* needs to uniquely identify a user. A handle, prefixed with an @ sign, is used to mention another user in a post. Examples can be seen in Figure 1 (the display name is in bold, and the handle is in a smaller font and lighter color).

The need for handles to be unique creates challenges in decentralized systems, since it requires an authority that determines which handle is assigned to which user. Mastodon's approach is to include the server name in the handle, which makes it difficult to move to another server. An alternative would be to use a blockchain-based naming system, such as the Ethereum Name System (ENS) [19]; this has the disadvantage of requiring the user to buy cryptocurrency in order to create an account, which we wanted to avoid.

Instead, Bluesky and atproto use DNS domain names as handles. If a user already owns a domain name, they can claim it as their Bluesky handle by adding a DNS record or by hosting a file under a `/.well-known/` HTTPS URL on that domain [37]. Users can also buy a new domain name within Bluesky, via a partnership with a domain registrar [15]. Alternatively, users can sign up for a subdomain of `.bsky.social` for free.

Using DNS domain names as handles has several advantages:

- We leverage the existing infrastructure of ICANN, registrars, and name servers, including for example the dispute resolution procedures for trademarks.
- Domain names are a well-known concept even among non-technical users, and they are short and simple.
- A user can move to a different server without changing their handle (see Section 3.5).
- Users do not need to host their own server to use their own domain name; a DNS record requires only a one-time setup and no ongoing maintenance.
- For organizations and people that already have a well-known domain name, using that name makes it easy for users to check that their Bluesky account is genuine. For example, the New York Times' handle is `@nytimes.com`.
- An organization can easily allow their staff to demonstrate their affiliation by granting them handles that are subdomains of the organization's main domain name (comparable to institutional email addresses). For example, a journalist's handle may indicate that they are at a particular news organization.
- Providers wanting to offer free subdomains can do so at very little cost.

2.3 Custom Feeds and Algorithmic Choice

Several decentralized social networks choose to offer only a reverse-chronological feed of posts from accounts the user is following – a backlash against the opaque content recommendation algorithms employed by mainstream centralized social networks. For example, Mastodon advertises itself as having "no algorithms or ads to waste your time" [39].

Our belief is that the problem lies not with algorithms *per se*, but rather with centrally controlled, opaque algorithms that remove user agency and prioritize user engagement over all else, e.g. by promoting controversial posts. Good recommendation algorithms can help users discover content that is relevant to them and find new accounts to follow – especially important for new users who are not yet following many accounts. They are also helpful for surfacing content on a particular topic, whereas following a user means seeing all of their posts, which might be on a mixture of topics, not all necessarily interesting to all followers. Giving users the ability to choose their algorithms lets them control what they want to see, rather than having the platform decide for them.

Bluesky Social PBC offers a selection of feed algorithms of its own, and also allows anybody to create their own feed generator [9]. Tens of thousands of custom feeds have already been created. Our goal is to offer an open and diverse marketplace of algorithms in which communities can adapt the system to suit their needs, and users have more agency over how they spend their time and attention [26]. Section 3.4 explains how feed generators work.

In Figure 1, a selection of bookmarked feeds is given at the top of the screen; in this example, the selected "Following" feed is the default reverse-chronological timeline, while "Week Peak Feed" (network-wide posts with many likes from the last week) and "Birds!" (photos and posts from birdwatchers) are third-party feeds. A feed generator can use arbitrary criteria to select its content. For example, the birdwatching feed uses a manually curated list of accounts, and selects posts from those accounts that contain a #birds hashtag or a feather emoji character. Alternative approaches, such as machine learning algorithms, are equally possible.

3 THE AT PROTOCOL ARCHITECTURE

Bluesky is the social app with the features explained in Section 2, while the AT Protocol is the underlying decentralized foundation. We maintain this separation because the AT Protocol is designed to support multiple *social modes*, not just Bluesky. For example, besides a Twitter-style microblogging app, atproto could also be used to implement Reddit-style forums, long-form blogs with comments, or domain-specific social applications such as link sharing or book reviews. The same user identity, social graph, and user data storage servers can be shared between all of these apps.

The data types and concepts for a particular social mode are defined by a *lexicon*, which specifies the schema of the data and the request endpoints involved in providing that social mode [7]. At the moment, the `com.atproto` lexicon defines the core AT Protocol concepts such as user identity (Section 3.5), and the `app.bsky` lexicon defines the microblogging mode. Anyone can define a new lexicon, allowing new social modes to coexist alongside the Bluesky social app on a shared infrastructure. The purpose of a lexicon is to provide documentation, to allow code generation and type-checking in applications, and to facilitate the process of specifying and versioning an interoperable protocol.

The biggest constraint for new social modes is that atproto is currently designed for content that users want to make publicly available. In particular, Bluesky user profiles, posts, follows, and likes are all public. Blocking actions are also currently public; however, we are investigating mechanisms for making these private [16, 41]. At present, Bluesky does not support private communications, such as direct messages, though we plan to add this in the future. Only a small amount of user state is currently private: any muted accounts and threads, notifications and their read/unread status, and user preferences such as pinned feeds and content filtering settings.

3.1 User Data Repositories

All data that a user wishes to publish is added to their *repository*, which stores a collection of *records*. Whenever a user performs some action – making a post, liking another user's post, following another

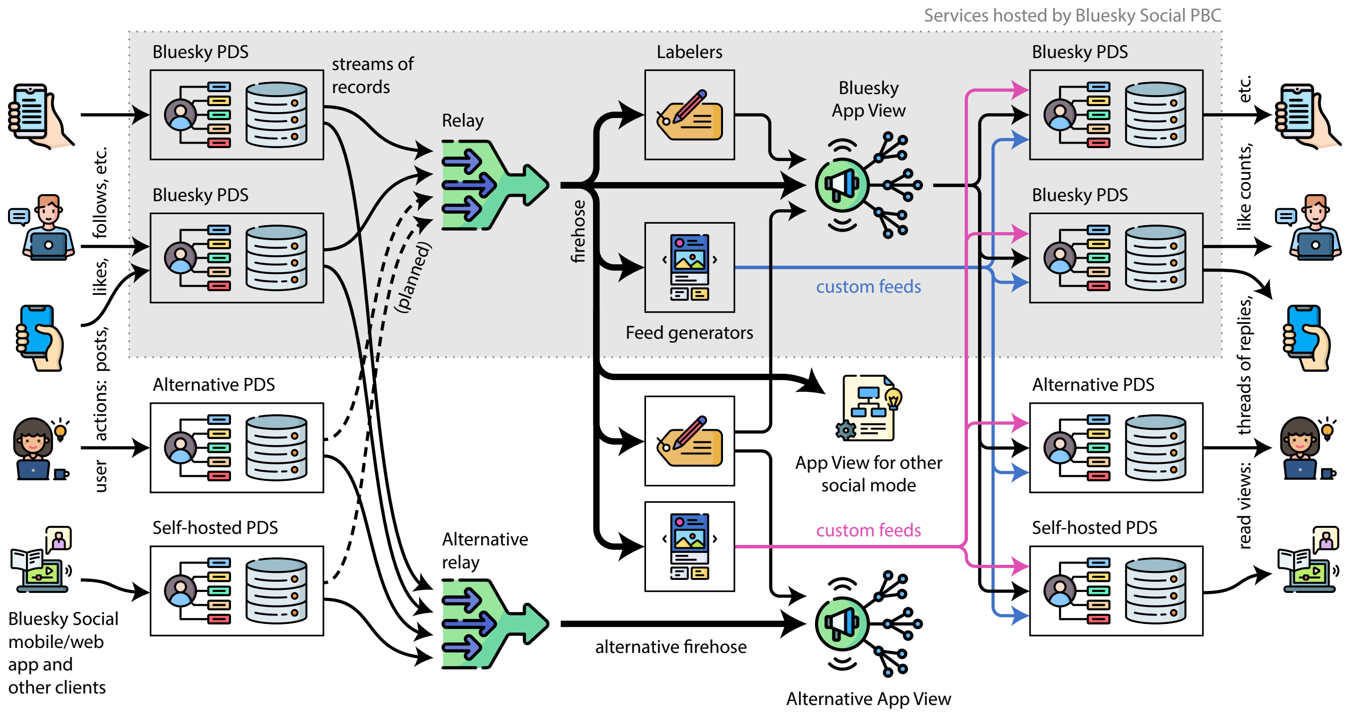


Figure 3: The main services involved in providing Bluesky, and data flows between them. Icons from Flaticon.com.

user, etc. – that action becomes a record in their repository. Records are encoded in DAG-CBOR [45], a restricted form of CBOR [17], a compact binary data format. The schema of records is defined by the lexicon, and a repository may contain a mixture of records from several different lexicons, representing user actions in different social modes. Media files (e.g. images) are stored outside of the user's repository, but referenced by their CID [32] (essentially a cryptographic hash) from a record in the repository. Similarly, a reference to a record in another repository (e.g. identifying a post being liked) also includes its CID.

Each user account has one repository, and it contains all of the actions they have ever performed, minus any records they have explicitly deleted. A *Personal Data Server (PDS)* hosts the user's repository and makes it publicly available as a web service; we discuss PDSes in more detail in Section 3.2.

A user only updates their own repository; for example, if user *A* follows user *B*, this results only in a follow record in user *A*'s repository, and no change to *B*'s repository. To find all followers of user *B* requires indexing the content of all repositories. This design decision is similar to the way hyperlinks work on the web: it is easy to find all the outbound links from a web page at a given URL, but to find all the inbound links to a page requires an index of the entire web, which is maintained by web search engines.

The *AT* in atproto stands for *Authenticated Transfer*, which reflects the fact that repositories are cryptographically authenticated. The records in a repository are organized into a *Merkle Search Tree* (MST), a type of Merkle tree that remains balanced, even as records are inserted or deleted in arbitrary order [3]. After every change to a repository, the root hash of the MST is signed; the public verification key for this signature is part of the user identity described in Section 3.5. This enables an efficient cryptographic proof that a given record appears within a given user's repository. Moreover, when a user updates or deletes a record, the MST enables a proof that the old record no longer appears in the repository.

3.2 Personal Data Servers (PDS)

A PDS stores repositories and associated media files, and allows anybody to query the data it hosts via a HTTP API. Moreover, a PDS provides a real-time stream of updates for the repositories it hosts via a WebSocket. Indexers (see Section 3.3) subscribe to this stream in order to find out about new or deleted records (posts, likes, follows, etc.) with low latency. This architecture is illustrated in Figure 3.

Hosting a PDS for a small number of users requires only small computing resources, even if those users have a large number of followers. Users who wish to self-host their own PDS can therefore do so on a cheap virtual machine in the cloud, or even on a Raspberry Pi connected to their home internet router. However, we expect that most users will sign up for an account on a shared PDS run by a professional hosting provider – either Bluesky Social PBC, or another company.

Compared to choosing a Mastodon server, the user's choice of PDS hosting provider is fairly inconsequential. The PDS URL is internal to the system, and is not normally visible to users. It makes no difference whether two users are on the same PDS or different

PDSes, since interaction between users goes via the indexing infrastructure in any case. A user can migrate from one PDS to another by simply copying their repository and media files to the new PDS, and pointing their account ID at the new PDS URL (see Section 3.5). Even if a PDS shuts down without warning, users can upload a backup of their repository to a new PDS, and thus recover their account without losing any of their posts or their social graph.

PDS operators will generally want to perform some basic moderation by deleting any illegal content hosted on their servers. However, PDS-level moderation is much less important than server-level moderation in Mastodon, because in atproto, the primary moderation role is taken on by seperate actors in the system – the labelers and feed generators (see Section 3.4). This allows different sets of people to offer server hosting and moderation services, respectively; we believe this separation is valuable since operating a server and moderating a community require largely disjoint sets of skills [46].

At the time of writing, Bluesky's indexing infrastructure (see Section 3.3) only indexes repositories on PDS instances hosted by Bluesky Social PBC itself; this limitation exists to limit infrastructure load and abuse problems during the beta period. In that sense, Bluesky is not yet fully decentralized. Support for third-party PDS operators is already implemented and enabled in Bluesky's *sandbox* (testing) environment, and a PDS implementation suitable for self-hosting is already open source [8]. We plan for the Bluesky indexing infrastructure to begin indexing repositories on other PDS operators (indicated by dashed arrows in Figure 3) in early 2024.

3.3 Indexing Infrastructure

On the web, websites are crawled and indexed by search engines, which then provide web-wide search and discovery features that the websites alone cannot provide. The AT Protocol is inspired by this architecture: the repositories hosted by PDSes are analogous to websites, and the indexing infrastructure is analogous to a search engine. User repositories are primary data (the "source of truth"), and the indexes are derived from the content of the repositories.

At the time of writing, most of Bluesky's indexing infrastructure is operated by Bluesky Social PBC (indicated by a shaded area in Figure 3). However, the company does not have any privileged access: since repositories are public, anybody can crawl and index them using the same protocols as our systems use. Client apps can switch to reading from a different index, or use a combination of multiple indexes.

While operating a small PDS is designed to be cheap, operating an indexer that ingests the entire network requires greater computing resources. We therefore expect that there will be fewer hobbyist indexers than self-hosted PDSes. Nevertheless, as Bluesky grows, there are likely to be multiple professionally-run indexers for various purposes. For example, a company that performs sentiment analysis on social media activity about brands could easily create a whole-network index that provides insights to their clients. Web search engines can incorporate Bluesky activity into their indexes, and archivists such as the Internet Archive can preserve the activity for posterity.

The indexing infrastructure operated by Bluesky Social PBC is illustrated in Figure 3. It is composed of multiple services that have integration points for external services.

3.3.1 *The Relay.*

The first component is the *Relay*, which crawls the user repositories on all known PDSes and consumes the streams of updates that they produce. The Relay checks the signatures and Merkle tree proofs on updates, and maintains its own replica of each repository. From this information, the Relay creates the *firehose*: an aggregated stream of updates that notifies subscribers whenever records are added or deleted in any of the known repositories.

The firehose is publicly available. Consuming the firehose is an easier way of building an index over the whole network, compared to directly subscribing to the source PDSes, since the Relay performs some initial data cleaning such as discarding malformed updates and filtering out high-volume spam. The firehose can optionally include Merkle proofs and signatures along with records, allowing subscribers to check that they are authentic.

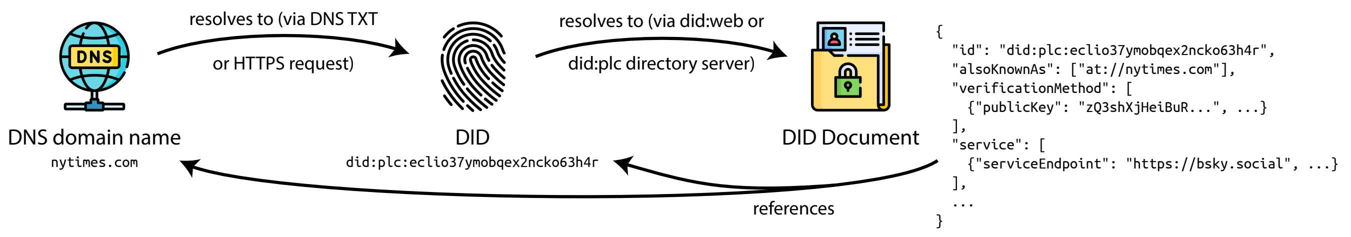
The Relay does not interpret or index the records in repositories, but simply stores and forwards them. Any developers wanting to create a new social mode on top of atproto can define a new lexicon with new record types, and these records can be stored in existing repositories and aggregated in the firehose without requiring any changes to the Relay.

3.3.2 *The App View.*

The App View is a service that consumes the firehose, and processes the records that are relevant to the Bluesky social app (records in the `com.atproto` and `app.bsky` lexicons). For example, the App View counts the number of likes on every post, and it collates the thread of replies to each post. The App View also maintains the set of followers for each user, and constructs the timeline containing the posts by the accounts that each user is following. It then offers a web service through which this information can be queried. When a record contains references to images, the App View fetches those files from the original PDS, resizes them if necessary to reduce the file size, and makes them available via a content delivery network (CDN).

To display this information in the user's client app, the client queries the user's own PDS, which then fetches the neccessary data from an App View. The App View is also responsible for enforcing moderation controls: 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 [61], 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 [16, 41]. If users are unhappy with the moderation rules applied by the App View operated by Bluesky Social PBC, it is always possible for third parties to operate alternative App Views that index the same firehose and present the data in a different way.

If the AT Protocol is used to implement another social mode besides microblogging, that application will most likely require an App View service of its own, which can be hosted by anyone. This service can then interpret and index the records in users' repositories in whatever way is required for that application.


resolves to (via DNS TXT or HTTPS request)
resolves to (via did:web or did:plc directory server)
DNS domain name
nytimes.com
DID
did:plc:eclio37ymobqex2ncko63h4r
DID Document
references

```
{
  "id": "did:plc:eclio37ymobqex2ncko63h4r",
  "alsoKnownAs": ["at://nytimes.com"],
  "verificationMethod": [
    {"publicKey": "zQ3shXjHeiBuR...", ...}
  ],
  "service": [
    {"serviceEndpoint": "https://bsky.social", ...}
  ],
  ...
}
```

Figure 4: A handle resolves to a DID, and a DID resolves to a DID document, which in turn references the handle, DID, and the user's public key. Icons from Flaticon.com.

3.4 Labelers and Feed Generators

Relay and App View aim to provide a mostly "unopinionated" service: they compute indexes over repositories in a neutral way, without attempting to rank or classify content. However, a good user experience also requires "opinionated" judgements for the purposes of content filtering (e.g. detecting sexually explicit images or spam) and curation (e.g. selecting posts on a particular topic).

The AT Protocol seperates out the "opinionated" aspects of the system into separate services: *labelers* and *feed generators*. These services typically take the firehose as their input. Labelers produce a stream of judgements about content (e.g. "this post is spam"), whereas feed generators return a list of post IDs they have selected for inclusion in a custom feed, as described in Section 2.3. Users can choose in their client app which feeds and which labelers they want to use. The output of labelers is consumed by App Views or PDSes in order to apply content filtering [12]. For a feed generator, an App View expands the post IDs into full posts before sending them to the client app of users who have subscribed to that feed.

Having labeler and feed generator services that are separate from App Views has several advantages:

- Anyone can run such services, which enables a pluralistic ecosystem in which different parties may make different judgements about the same piece of content. Users, as well as the operators of App Views and PDSes, can decide whose judgements they want to trust, and it is easy for them to switch to alternative labeling and feed generation services if their current providers fail to meet their expectations.
- It becomes easier to set up alternative App View providers: since any App View can consume the publicly available output from labelers and feed generators, there is less pressure for each App View to develop its own content filtering infrastructure. Having alternative App Views is important for a healthy, decentralized marketplace.

Feed generators can be implemented in code using our starter kit [10], or created with a third-party service such as Skyfeed [13].

3.5 User Identity

As explained in Section 2.2, user handles in Bluesky and atproto are DNS domain names. Any number of identity providers can coexist in the system: Bluesky Social PBC allows users to register subdomains of `.bsky.social`, but the indexing infrastructure does not treat users differently based on their handle.

We want a user to be able to change their handle without affecting their social graph. Therefore, when a record in user *A*'s repository indicates that *A* is following *B*, that record must identify *B* in a way that is more long-lived than specifying *B*'s handle. For this reason, every Bluesky/atproto account has an immutable, unique identifier: a *decentralized ID* or *DID*, which is a URI starting with the prefix `did:`. The record that *A* follows *B* then contains *B*'s DID. DIDs are a recent W3C standard [50].

Moreover, we want a user to be able to migrate to a different PDS without changing either their DID or their handle. The DID specification provides a mechanism for *resolving* a DID into a *DID document*, a JSON document containing information about the user identified by that DID, as illustrated in Figure 4. In atproto, a DID document specifies (among other things) the handle of the user, the URL of their PDS, and the public key that is used to sign the Merkle tree root of their repository every time they add or delete a record. To change their handle or their PDS, the user needs to update their DID document to the new value.

For a user to successfully claim a particular handle, they must have a bidirectional link between their DID and their domain name handle, as shown in Figure 4:

- A link from the handle to the DID is established either by storing the DID in a DNS TXT record on that domain name, or by returning the DID in response to a HTTPS request to a `/.well-known/` URL on that domain name [37].
- A link from the DID to the handle is established by including the handle in the DID document that is returned when the DID is resolved.

3.5.1 Resolving DID documents. The W3C DID specification [50] does not directly specify the mechanism for resolving a DID into a DID document. Rather, the first substring after `did:` in a DID indicates the *DID method*, and the specification of the DID method defines the protocol for obtaining the DID document. Hundreds of DID methods have been defined [54], many of which are dependent on specific blockchains or other external systems. To avoid atproto implementations having to support so many resolution methods, our services currently only accept DIDs based on either `did:web` (defined by the the W3C Credentials Community Group [27]) or `did:plc` (defined by ourselves for atproto [31]). Support for more DID methods might be added in the future.

The `did:web` method is very simple: the part of the DID after `did:web:` is a domain name, and the DID document is resolved by making a HTTPS request to a `/.well-known/` URL on that domain name (a path can optionally be included). The security of a `did:web` identity therefore assumes that the web hosting provider for that domain is trusted, and also relies on trusting the TLS certificate authorities that may authenticate the HTTPS request.

`did:web` identities are therefore similar to domain name handles, with the difference that the name cannot be changed, since a DID is an immutable identifier. This makes `did:web` appropriate for the identity of organizations that are already strongly linked to a particular domain name. For most users, `did:plc` is more appropriate, since it uses a domain name only as a handle that can be changed.

3.5.2 The did:plc DID method. When a user creates an account on the Bluesky social app, they are by default assigned a DID of the form `did:plc:eclio37ymobqex2ncko63h4r`, where the string after the prefix `did:plc:` is the SHA256 hash of the initial DID document, truncated to 120 bits and encoded using base32 [31]. A DID of this form can be resolved to the corresponding DID document by querying a server at https://plc.directory/, which is currently operated by Bluesky Social PBC; in the future we plan to establish a consortium of independent operators that collectively provide the PLC directory service.

The PLC directory server plays an authoritative role similar to the DNS root servers, but it is mostly untrusted because PLC DID documents are self-certifying. If the DID document has not changed since its initial creation, it is easy to verify that a DID has been correctly resolved to a DID document by recomputing its hash. To support changes to the DID document, the initial version of a user's DID document contains a public key that is authorized to sign a new version of the DID document. Any new version of the DID document is only valid if it has been signed by the key in the previous version. The directory server returns all DID document versions for a given DID, allowing anybody to check the chain of signatures.

If the directory server were to be malicious, it would not be able to modify any DID documents – it could only omit valid DID document versions from its responses, or fail to respond at all. Moreover, if there were to be a fork in DID document history such that two correctly signed successor versions for some DID document exist, the directory server could choose which one of these forks to serve. To mitigate the risk of such attacks, we anticipate that a future version of the PLC directory will use techniques from certificate transparency [33] to ensure that DID document updates form an append-only log.

3.5.3 Authentication. 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 authenticate themselves 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, and other PDS operators are free to use different authentication methods.

4 RELATED WORK

Several other decentralized social networks are in development. We believe that there is no single optimal design: different systems make different trade-offs, and are therefore suitable for different purposes. Bluesky and the AT Protocol aim to provide a good user experience by making moderation a first-class concern, by having clients that are lightweight and fast, and by providing a global view over the whole network. For example, conversation threads include all replies (unless removed by moderation), regardless of the server on which they were posted. To achieve this goal we rely on an indexing infrastructure that is more centralized than some other designs. However, we emphasize that there can be multiple competing indexers, and third-party client apps are free to show data from whichever indexers they wish.

In 2021 some of our team published a review of the decentralized social ecosystem [25]. In this section we summarize some recent developments that have happened since, and we refer to the review for a more comprehensive comparison of protocols and systems.

Many decentralized social networking projects have ideas in common. For example, the idea of using DNS domain names as usernames also appears in Nostr [23]. An atproto PDS has similarities to Git repository hosting (e.g. GitHub/Gitlab) or a Solid pod [49]. There are also federated chat systems such as Matrix [40], IRC [43], and XMPP [47], but we focus on systems that provide a Twitter-like model where users follow each other.

4.1 Scuttlebutt

Secure Scuttlebutt (SSB) is a peer-to-peer social networking protocol [1]; Manyverse [57] and Planetary [59] are social applications built upon the SSB protocol. It optionally uses relay servers called *pubs* to store messages from peers that are offline, and to enable user discovery. The client software downloads the feeds from accounts that the user is explicitly following, and from accounts followed by followed accounts (up to three hops by default). This can require significant amounts of storage and bandwidth on the client.

Any messages from users outside of the third-degree network are not shown, which effectively limits the set of people who can mention or reply to a user to the third-degree network. This deliberate design decision is intended to reduce moderation problems by prioritizing conversation between people who already know each other [53]. In contrast, Bluesky/atproto are designed to allow anybody to talk to anybody else. This requires more explicit moderation to manage unwanted content, but we believe it also enables serendipity and is a prerequisite for any "digital town square".

Since SSB is built upon append-only logs and gossip replication, it is not possible to delete content once it has been posted [56]. User identity is tied to a cryptographic key on the user's device, requiring manual key management for moving to another device. Posting from multiple devices is not possible, as sharing the same key between devices can make an account unrecoverable [55]. A work-in-progress successor protocol to SSB, called PPPPP, is designed to address these issues [52].

4.2 Nostr

Nostr also began as a revision of SSB, replacing the append-only logs with individual signed messages [30]. It leans more heavily on relay servers instead of peer-to-peer communication: clients publish and fetch messages on relays of their choice, and there is no federation among relays [21]. The protocol is deliberately simple, and it prioritizes censorship resistance over moderation: relays can block users, but users can always move to a new relay, and use multiple relays at the same time. Communication (e.g. reply threads) is only possible between users who have at least one relay in common. Although some services index the whole Nostr network, these indexes are not used for user-to-user interaction. As a result, it is unpredictable who will see which message. The creator of Nostr writes: "there are no guarantees of anything [. . .] to use Nostr one must embrace the inherent chaos" [22]. Key management is manual in Nostr, and facilities for key rotation are still under discussion [4].

4.3 Farcaster and blockchain-based systems

Farcaster [60] has some architectural similarities to Bluesky/atproto, although it was developed independently. It has storage servers called *hubs*, which store the state of the entire network similarly to an atproto Relay, and it has a concept of *hosted app servers* that are similar to our App View [51]. Farcaster user IDs are similar to our DIDs, and they are mapped to public keys using a smart contract on the Ethereum Optimism blockchain that is functionally similar to our PLC directory. Usernames can be either ENS names [19], or names within an off-chain namespace managed centrally by Farcaster, similarly to `.bsky.social` subdomains in Bluesky [20].

A difference is that Farcaster has no equivalent to atproto's PDS; instead, client apps publish signed messages directly to a hub, and hubs synchronize messages using a convergent gossip protocol. Users must pay in cryptocurrency to register their public key, and for hub data storage (at the time of writing, Ethereum equivalent to $5 USD/year); when a user exceeds their storage allowance, old messages are deleted. Fees are currently collected centrally by the Farcaster team [29]. In contrast, the AT Protocol does not specify storage limitations, but leaves it to providers of PDS and indexing services to define their own business model and abuse-prevention policies. We also prefer to avoid a dependency on a cryptocurrency.

The Lens protocol [35] is more strongly blockchain-based than Farcaster: it even stores high-volume user actions such as posts and follows on Polygon, a proof-of-stake blockchain. DSNP takes a similar approach [44]. Placing high-volume events directly on a blockchain incurs orders of magnitude higher per-user costs than atproto, and is likely to run into scalability limits as the number of users grows. Lens is adopting a layer-3 blockchain that provides better scalability and lower cost [36], but weaker security properties. Linking social accounts to cryptocurrency wallets and NFTs enables users to monetize their content, but this is not a goal of atproto.

4.4 ActivityPub and Mastodon

ActivityPub [34] is a W3C standard for social networking, and Mastodon [39] is its most popular implementation. We have highlighted aspects of their design in Sections 1, 2.3, and 3.2. Mastodon gives a lot of power to server administrators: for example, a server admin can choose to block another server, preventing all communication between users on those servers. There is a degree of lock-in to a server because moving to another server is intrusive: the username changes, moving posts to the new server currently requires an experimental command-line tool [48, 58], and other users' replies to those posts are lost. These risks can be mitigated by self-hosting; managed providers exist [28], but they still require some expertise and cost money. The AT Protocol separates the roles of moderation and hosting, and aims to make it easier to change providers.

When user A follows user B, A's server asks B's server to send it notifications of B's future posts via ActivityPub. This architecture has the advantage of not requiring a whole-network index. However, replies to a post notify the server of the original poster, but not necessarily every server that has a copy of the original post, leading to inconsistent reply threads on different servers. Notifications can be forwarded, but in the limit this leads to each server having a copy of the whole network, which would make it expensive to run a server. Viral posts can generate a lot of inbound requests to a server from people liking, replying, and boosting (reposting). The Bluesky indexing infrastructure is also fairly expensive, but a PDS is cheap to run. Since users can choose their moderation preferences independently from their indexing provider (App View), we believe that the ecosystem can be healthy with a small number of indexing providers.

5 CONCLUSIONS

Bluesky and the AT Protocol are a new approach to social media. While some decentralized systems prioritize censorship resistance, we believe that a good user experience requires explicitly addressing problematic content such as harassment and misinformation. We therefore make moderation a first-class concern that is handled separately from infrastructure hosting, and we provide strong mechanisms for users to control the content they see.

Our open architecture allows a pluralistic system in which there is no global consensus on what content is acceptable. A user on a self-hosted or loosely moderated PDS may post controversial content, but they are not entitled to the attention of others: App Views may choose not to index the content, and clients may ignore it depending on the user's moderation settings. This philosophy is sometimes described as "free speech, but not free reach" [18].

The AT Protocol provides cryptographically authenticated data, but our implementation pairs it with custodial key management to provide a familiar user experience. The system is open to third-party clients and alternative PDS hosts, and anybody can index the network using real-time streams for low-latency updates. This reduces the dependency on any one provider, since every part of the system can be run by multiple competing providers, and users can switch providers with minimal friction (in particular, without changing username, and without losing any of their content or social graph). The AT Protocol is also extensible to other social modes besides microblogging.

ACKNOWLEDGMENTS

Martin Kleppmann is supported by the Volkswagen Foundation and crowdfunding supporters including Mintter and SoftwareMill.

REFERENCES

[1] [n. d.]. Scuttlebutt Protocol Guide. https://ssbc.github.io/scuttlebutt-protocol-guide/
[2] Ben Adida. 2022. Don't let federation make the experience suck. https://benlog.com/2022/12/28/dont-let-federation-make-the-experience-suck/ Archived at

https://perma.cc/W7CY-TF23.

[3] Alex Auvolat and François Taïani. 2019. Merkle Search Trees: Efficient State-Based CRDTs in Open Networks. In *38th Symposium on Reliable Distributed Systems (SRDS 2019)*. IEEE, 221–230. https://doi.org/10.1109/srds47363.2019.00032

[4] Cat Ball, fiatjaf, Kevin Smith, Vitor Pamplona, et al. 2022. Nostr issue #45: Key distribution, rotation, and recovery. https://github.com/nostr-protocol/nostr/issues/45 Archived at https://perma.cc/26TW-ME48.

[5] Chelsea Barabas, Neha Narula, and Ethan Zuckerman. 2017. *Defending Internet Freedom through Decentralization: Back to the Future?* Technical Report. MIT Media Lab. https://dci.mit.edu/decentralizedweb Archived at https://perma.cc/Q8CJ-D44Y.

[6] Bluesky Social PBC. [n. d.]. AT Protocol Community Projects. https://atproto.com/community/projects Archived at https://perma.cc/X88A-9XM4.

[7] Bluesky Social PBC. [n. d.]. AT Protocol Specification. https://atproto.com/specs/atp

[8] Bluesky Social PBC. [n. d.]. GitHub repositories. https://github.com/bluesky-social

[9] Bluesky Social PBC. 2023. Algorithmic Choice with Custom Feeds. https://blueskyweb.xyz/blog/7-27-2023-custom-feeds Archived at https://perma.cc/Z6U4-VMY8.

[10] Bluesky Social PBC. 2023. ATProto Feed Generator. https://github.com/bluesky-social/feed-generator

[11] Bluesky Social PBC. 2023. Bluesky Proposal 0001: User Lists, Reply-Gating, and Thread Moderation. https://github.com/bluesky-social/proposals/tree/main/0001-user-lists-replygating-and-thread-moderation

[12] Bluesky Social PBC. 2023. Bluesky Proposal 0002: Labeling and Moderation Controls. https://github.com/bluesky-social/proposals/tree/main/0002-labeling-and-moderation-controls

[13] Bluesky Social PBC. 2023. Featured Community Project: SkyFeed. https://atproto.com/blog/feature-skyfeed Archived at https://perma.cc/AYR8-AY5K.

[14] Bluesky Social PBC. 2023. Moderation in a Public Commons. https://blueskyweb.xyz/blog/6-23-2023-moderation-proposals Archived at https://perma.cc/XFX2-CCFJ.

[15] Bluesky Social PBC. 2023. Purchase and Manage Domains Directly Through Bluesky. https://blueskyweb.xyz/blog/7-05-2023-namecheap Archived at https://perma.cc/QUA7-L8QJ.

[16] Bluesky Social PBC. 2023. Why are blocks on Bluesky public? https://atproto.com/blog/block-implementation Archived at https://perma.cc/2ZQX-KTNJ.

[17] Carsten Bormann and Paul Hoffman. 2020. RFC 8949: Concise Binary Object Representation (CBOR). IETF Standards Track. https://datatracker.ietf.org/doc/html/rfc8949

[18] Renee DiResta. 2018. Free Speech Is Not the Same As Free Reach. WIRED. https://www.wired.com/story/free-speech-is-not-the-same-as-free-reach/ Archived at https://perma.cc/ZF5R-USHM.

[19] ENS Labs Limited. [n. d.]. Ethereum Name Service. https://ens.domains/about/

[20] Farcaster Team. [n. d.]. Farcaster Architecture. https://docs.farcaster.xyz/protocol/architecture.html Archived at https://perma.cc/7PDP-ATTH.

[21] fiatjaf. [n. d.]. nostr - Notes and Other Stuff Transmitted by Relays. https://github.com/nostr-protocol/nostr Archived at https://perma.cc/6YCW-VERW.

[22] fiatjaf. 2023. A vision for content discovery and relay usage for basic social-networking in Nostr. https://fiatjaf.com/3f106d31.html Archived at https://perma.cc/9N8B-DLXW.

[23] fiatjaf and Michael Dilger. 2021. NIP-05: Mapping Nostr keys to DNS-based internet identifiers. https://github.com/nostr-protocol/nips/blob/master/05.md

[24] Gildásio Filho. [n. d.]. deck.blue. https://deck.blue/

[25] Jay Graber. 2021. Ecosystem Review. https://gitlab.com/bluesky-community1/decentralized-ecosystem Archived at https://perma.cc/RJ2Y-H6YT.

[26] Jay Graber. 2023. Algorithmic choice. https://blueskyweb.xyz/blog/3-30-2023-algorithmic-choice Archived at https://perma.cc/WQR6-5QJF.

[27] Christian Gribneau, Michael Prorock, Orie Steele, Oliver Terbu, Mike Xu, and Dmitri Zagidulin. 2023. did:web Method Specification. W3C Credentials Community Group. https://w3c-ccg.github.io/did-method-web/ Archived at https://perma.cc/WB8M-8ECW.

[28] Grow your own services. [n. d.]. Grow your own social network. https://growyourown.services/grow-your-own-social-network/ Archived at https://perma.cc/KS4A-RAEW.

[29] Cassie Heart, horsefacts, and Varun Srinivasan. 2023. FIP-6: Flexible Storage. https://github.com/farcasterxyz/protocol/discussions/98 Archived at https://perma.cc/9JT5-DR3V.

[30] Evan Henshaw-Plath. 2023. Pivoting Protocols, from SSB to Nostr. https://www.nos.social/blog/pivoting-from-ssb-to-nostr Archived at https://perma.cc/9Y63-28YM.

[31] Daniel Holmgren, Bryan Newbold, Devin Ivy, and Jake Gold. 2023. DID PLC Method (did:plc). https://github.com/did-method-plc/did-method-plc

[32] IPFS Documentation. [n. d.]. Content Identifiers (CIDs). https://docs.ipfs.tech/concepts/content-addressing/ Archived at https://perma.cc/65PP-ZRQW.

[33] Ben Laurie. 2014. Certificate Transparency. *ACM Queue* 12, 8 (Aug. 2014), 10–19. https://doi.org/10.1145/2668152.2668154

[34] Christine Lemmer-Webber, Jessica Tallon, Erin Shepherd, Amy Guy, and Evan Prodromou. 2018. ActivityPub. W3C Recommendation. https://www.w3.org/TR/2018/REC-activitypub-20180123/

[35] Lens Protocol. [n. d.]. Lens Protocol Overview. https://github.com/lens-protocol/core Archived at https://perma.cc/SFA7-7CQ6.

[36] Lens Protocol. 2023. Introducing Momoka to Scale Lens. https://mirror.xyz/lensprotocol.eth/3Hcl0dGE8AOYmnFolzqO6hJuueDHdsaCs3ols2ruc9E Archived at https://perma.cc/5SY9-PCP3.

[37] Emily Liu. 2023. How to set your domain as your handle. https://blueskyweb.xyz/blog/4-28-2023-domain-handle-tutorial Archived at https://perma.cc/4LNR-6YC5.

[38] Mike Masnick. 2019. *Protocols, Not Platforms: A Technological Approach to Free Speech.* Technical Report. Knight First Amendment Institute at Columbia University. https://knightcolumbia.org/content/protocols-not-platforms-a-technological-approach-to-free-speech Archived at https://perma.cc/2V36-FKV3.

[39] Mastodon gGmbH. [n. d.]. Mastodon. https://joinmastodon.org/

[40] Matrix.org Foundation. [n. d.]. Matrix: An open network for secure, decentralised communication. https://matrix.org/

[41] Bryan Newbold. 2023. Mechanisms for private "block" relationships between Bluesky accounts. https://github.com/bluesky-social/atproto/discussions/1131 Archived at https://perma.cc/2FWX-NPAX.

[42] Samuel Newman. [n. d.]. Graysky: Bluesky, like you've never seen it before. https://graysky.app/

[43] Jarkko Oikarinen and Darren Reed. 1993. RFC 1459: Internet Relay Chat Protocol. IETF Network Working Group. https://datatracker.ietf.org/doc/html/rfc1459

[44] Project Liberty. 2020. Decentralized Social Networking Protocol (DSNP). https://dsnp.org/dsnp_whitepaper.pdf Archived at https://perma.cc/RD62-RCKA.

[45] Protocol Labs. [n. d.]. Specification: DAG-CBOR. https://ipld.io/specs/codecs/dag-cbor/spec/ Archived at https://perma.cc/D7UV-EUFL.

[46] Yoel Roth and Samantha Lai. 2023. Collective Security in a Federated World. In *Scaling Trust on the Web.* Atlantic Council, Chapter Annex 5. https://www.atlanticcouncil.org/in-depth-research-reports/report/scaling-trust/ Archived at https://perma.cc/CT3R-DCF5.

[47] Peter Saint-Andre. 2011. RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core. IETF Standards Track. https://datatracker.ietf.org/doc/html/rfc6120

[48] SilverWolf32 et al. 2019. Mastodon issue #12423: Support Post Migration. https://github.com/mastodon/mastodon/issues/12423

[49] Solid Project. [n. d.]. Solid. https://solidproject.org/

[50] Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. 2022. Decentralized Identifiers (DIDs) v1.0: Core architecture, data model, and representations. W3C Recommendation. https://www.w3.org/TR/did-core/

[51] Varun Srinivasan. [n. d.]. Farcaster: A Decentralized Social Network. https://github.com/farcasterxyz/protocol/blob/main/docs/OVERVIEW.md

[52] André Staltz. 2023. Manyverse Blog: June 2023 update. https://www.manyver.se/blog/2023-06-05 Archived at https://perma.cc/D568-KD26.

[53] André Staltz. 2023. Manyverse Blog: May 2023 update. https://www.manyver.se/blog/2023-05-05 Archived at https://perma.cc/9D7E-E8EH.

[54] Orie Steele and Manu Sporny. 2023. DID Specification Registries: The interoperability registry for Decentralized Identifiers. W3C DID Working Group. https://w3c.github.io/did-spec-registries/#did-methods Archived at https://perma.cc/LM4T-JTZ5.

[55] The Manyverse Authors. [n. d.]. FAQ: How can I use my account on many devices? https://www.manyver.se/faq/account-on-many-devices Archived at https://perma.cc/5S9S-NA9U.

[56] The Manyverse Authors. [n. d.]. FAQ: How do I delete content? https://www.manyver.se/faq/permanence Archived at https://perma.cc/DSB4-6H78.

[57] The Manyverse Authors. [n. d.]. Manyverse. https://www.manyver.se/

[58] Tokyo Outsider. 2023. MastodonContentMover. https://mastodoncontentmover.github.io/ Archived at https://perma.cc/EGM8-RM8U.

[59] Verse Communications Inc. [n. d.]. Planetary Social. https://www.planetary.social/

[60] Warpcast. [n. d.]. Farcaster: A protocol for decentralized social apps. https://www.farcaster.xyz/

[61] X Help Center. [n. d.]. How to block accounts on X. https://help.twitter.com/en/using-x/blocking-and-unblocking-accounts Archived at https://perma.cc/VZZ6-CSCM.

[62] Douglas Yeung. 2023. The 'Digital Town Square' Problem. The RAND Blog. https://www.rand.org/blog/2023/01/the-digital-town-square-problem.html Archived at https://perma.cc/3GM7-3VPP.