The Dogwood Protocol

A new protocol for federated, self-moderating social media.

Abstract

Dogwood is an open protocol for building interoperable social media software. Built in response to the hegemony of "web 2 walled gardens", the primary goal is to simultaneously eliminate centralized censorship and enable the creation of enjoyable spaces for network participants. Dogwood's approach combines proven "web 1" technologies with client-owned, cryptographically secured data, a multi-server architecture, and bitcoin payments with the goal of aligning the topology of the network with the social graph.

Introduction

The failures and successes of mainstream social networks like Twitter, Facebook, Reddit, YouTube, and the like, are equal parts self-evident and complex. I will not spend time enumerating them here, except to differentiate which problems Dogwood attempts to solve, and which it does not.

On October 14th, 2020 the New York Post published a controversial article relating to the upcoming US presidential election. This article was quickly censored by both Facebook and Twitter. While other "misinformation" had been suppressed by these platforms in the past, this was a wake-up call for many. It was evident that two of the most powerful platforms for free speech in the history of the world could not be trusted to act in the interests of their users.

The classification of social media companies as publishers which are responsible for the content they disseminate implicitly granted them the privilege of curation of content. This privilege has slowly expanded into the ability to shape the current political narrative with near impunity.

Add to this the fact that when a user is banned from these platforms he may lose not only all the content he has produced, but also all his followers, and it's easy to see how users are held hostage by these platforms. They provide unmatched reach and convenience, but they can also take it away.

The problems of censorship and ownership of data are the two primary problems Dogwood aims to solve. However, the real challenge of such a solution is that the traditional strategies for scaling, moderation, and monetization all require centralization of both platform and data. This is where things get interesting.

While it is my hope that dogwood will have a positive influence on the social media experience itself by restructuring the social graph to emphasize locality and relevance, and enabling the creation of third-party clients that can provide a different user experience, it is not my goal to address the very real problems of social media addiction and its corrosive effects on society. One thing at a time.

What came before

I am not the first to tackle this problem. There exist or are in development several interesting solutions that seem to me to have certain flaws, but which may succeed where Dogwood fails. That is ok by me, all I want is for a good open protocol to emerge that solve the problems outlined above. What follows is my criticism of those solutions, and a justification for why I am not dedicating my effort to contributing to them.

ActivityPub

Best known by its most successful application, Mastodon, ActivityPub is the OG federated social protocol, powering several different applications. Its limitations are pretty well known, the biggest complaint being that users do not really own their data. While ActivityPub does include the ability to migrate from one server to another, it requires the cooperation of the server admin, which isn't always a given. In this sense, ActivityPub is too restrictive, and so is still open to asymmetric interference by node operators.

The flip side of ActivityPub is that in other ways it is too permissive. Content is replicated from one server to another based on the node operator's preferences, not the users'. Likewise, moderation is implemented using a patchwork of techniques that fail to protect the user from content they'd rather not see. Users can mute content in various ways, but they can't rely on anyone to help them curate their experience for them.

Secure Scuttlebutt

In my opinion, the best implementation of a decentralized social media protocol is secure scuttlebutt. Their documentation is excellent, their community vibrant, and full of good ideas. SSB is the source of many of the ideas contained in Dogwood, particularly regarding distributed moderation.

SSB has one fatal flaw however — each user's content is represented as a cryptographically linked list, meaning a user's entire activity history must be downloaded in order to validate the authenticity of their final post. Add to this the fact that media is embedded in the social graph, and the result is an extremely heavy mandatory download, which cannot scale to a large number of users.

This problem is compounded by the fact that while SSB is (beautifully) engineered to work in a p2p network, unmediated by middlemen, in practice most users join "pubs" in order to discover new content. This results in a spray-and-pray approach to content replication similar to ActivityPub.

Nostr

Dogwood is most similar to the Nostr project. I agree with almost everything said on their readme about the problems with current social networks and how the solution should work, on a technical level at least. The two projects are similar enough that it may be possible to switch applications built on Dogwood over to Nostr if Nostr proves to find the level of adoption a social network needs to succeed.

Both protocols focus on a single linear feed of events, cryptographically signed by a private key owned by the originating account, and replicated on demand. Both protocols also support a multi-server architecture, making it possible for users to sidestep the trust in node operators common with other federated architectures.

The primary difference between the two protocols is that while Dogwood is designed to focus on solving the problems specifically related to a social-media use-case, Nostr is a more naive event-stream protocol, where social is just one use-case of many (see Citadel Dispatch episode #63).

This lack of focus, coupled with the terse nature of the Nostr protocol, is likely to result in scaling problems and a broken user experience stemming from ambiguity in how clients might interpret event data. More opinionated validation and a well-demarcated plugin architecture like what Dogwood defines can help sidestep this limitation.

BlueSky

BlueSky is an interesting newcomer to the space. As such, I don't have much to say about them, except that their work so far is promising. Many of their solutions seem to be higher-quality versions of what Dogwood is intended to provide: their implementations of DIDs, content replication, and moderation all contain more detail than would be practical to build as a solo developer or small team, but will perform much better at scale. They have done extensive research, and have recently open-sourced their efforts. I'm optimistic they could create a well-funded, technically solid foundation for a new social media substrate.

Enter Dogwood

The Dogwood protocol is an attempt to solve the problems identified above with existing social networks by taking a relatively low-tech approach to social media — essentially an RSS feed for people rather than websites.

Many attempts have been made in the past to create a read/write internet, but have failed because social media is already exactly that — all that's left is to remove the walled gardens in order to enable its application to new problems by creative people.

Dogwood breaks this problem down into a few key areas:

- Cryptographic, pseudonymous identity
- A general purpose schema for social content, applicable to many types of social media
- Multi-master architecture for federated content hosting, and a user-interaction-driven gossip protocol which includes distributed moderation and spam prevention
- Monetization of server resources and content using the Bitcoin Lightning network

Additional details can be found in the protocol repository.

Cryptographic Identity

The base layer of any society is people. However, whether or not a person chooses to reveal their identity in any given context has certain trade-offs. Revealing your identity allows you to demonstrate trust through existing reputation. In contrast, withholding your identity allows you to speak freely, without regard for your existing reputation or personal safety.

There is interesting work being done in the area of zero-knowledge proofs that allow for a partial reconciliation of these two options, but I will simplify for the purposes of our discussion.

Because a hidden thing can be revealed but existing knowledge cannot be revoked, the default approach to identity management is to require as little of a user as possible: a simple public/private key pair. This can be generated from a source of randomness when creating an account, or derived from an existing source. An example of this would be to re-use a lightning wallet key pair by means of the WebLN standard.

Every message originating from a user can then be signed by their private key, and identified by their public key. This creates a continuity of identity across multiple messages, and proves that the author is who he says he is. No additional verification step is needed (see below for how spam prevention might be implemented).

That's not to say that users may not make additional claims about themselves, but the burden of proof is on them — impersonation is possible in this system, and so users should be wary. This is not a novel problem in social media, and may be mitigated by the reputation system described below.

In contrast to Secure Scuttlebutt, messages need not be linked in any way, meaning only a partial record of an account's full history need be retrieved. Old messages may be deleted at a server's discretion to reduce storage requirements, and missing content (identified by a serial identifier embedded in messages) can be requested from other peers in the network.

Credential rotation in the event that a private key is compromised is not yet designed, but is an area of active research in the bitcoin community. Multi-signature keys might be an option. Another option would be to request account deletion, and re-publish under a new key. Well-behaved network peers would then ignore subsequent events signed by the original key. One interesting effect of this is that an impersonation attack could include an account's full event history. This, however, could be detected by a sophisticated peer and ignored. This is an area of active research.

Modeling Social Media

Dogwood's approach to defining a general-purpose model for social media attempts to be minimalistic, while still being useful. The model described below can be applied to many different types of social media including chat, marketplaces, bulletin boards, media sharing, and more. The differentiators of existing social media platforms mostly consist in the curation of the user experience to a particular end, but their data models show a striking convergence. Because Dogwood is an open protocol, different experiences may be built on a common open substrate.

Dogwood defines the following terms:

- Account — represents a user and can be annotated to include metadata with claims about them
- Event — the basic representation of a user's interaction with the network. All other objects in the system are generated as a result of an event being created.
- Vote — a way for a user to register their approval or disapproval of a certain piece of content.
- Blob — a reference to some heavier-weight content like a document, an image, or a video.
- Post — the basic representation of a piece of content. Different content types can be defined

determining how a post is to be interpreted, including referencing topics, mentions, and other content.
- Annotation — because of the problems inherent with content deletion, the use of annotations to edit or qualify a post are encouraged instead.
- Group — a namespace and brokering mechanism for creating a private thread to which new content can be privately posted. This can cover the use cases of private messaging as well as private and public groups to which new parties can be invited.
- Wallet — a way for a user to send and receive payments related to participation in network activity.

Dogwood also provides a plugin architecture which allows servers to define extensions and propagate their definitions through the system, as well as a spec for HTTP and Websocket APIs which allows for a basic level of interoperability between different client and server software.

Architecture & Gossip

Dogwood uses a multi-master architecture similar to that of Nostr, which allows a client to keep a copy of its user's data and selectively publish it to multiple server nodes in the system.

A "client" is a software application that is trusted by and acting on behalf of a user. This may be a website, a mobile app, a desktop app, or a command line tool. This may include any additional features desired, including message signing delegation using something like WebLN, trusted content curation sources, and other layered features. A "server" is a regular web server that follows the Dogwood protocol, accepting, storing, and re-publishing content on request.

This architecture and how content flows through it is the most important part of the Dogwood protocol. This is a radical departure from all other existing approaches to social media discussed above. It bears repeating: Dogwood's goal is to align the topology of the network with the social graph.

The way this is implemented is using "votes". A user may specify a decimal value ranging from -1 to 1 for any object in the system. How these votes are represented to the user is up to the client software, but for example a positive vote on a post may correspond to a like; a negative vote to a dislike. Accounts, servers, groups, and anything else may also be voted on.

Votes are also the mechanism for sharing content across the network. If a vote is published to a server that does not already know about the content to which the vote refers, it should request that content from the source specified in the vote's message. In this way, content can selectively hop across nodes in the system using user votes as the signaling mechanism.

Votes do not cascade through the entire network, but are limited in effect to the servers to which they are published. This allows for different community standards to emerge based on the set of people who publish

their votes to a given server. A user can then choose communities he wants to participate in, while still having selective access to the wider network. Client software may also allow a user to identify certain other users, or even external sources, whose preferences they wish to weight more heavily when requesting content suggestions.

This accomplishes the common social use case of generating content recommendations. But more importantly, it forms the basis for a system of distributed moderation. A few very strong downvotes may indicate spam or illegal content, and can quickly be suppressed. Votes originating from a very high-reputation account might be weighted even more heavily.

In combination with this decentralized moderation system, outright censorship may also be employed by server admins using all the usual techniques to take down illegal content without compromising the censorship-resistance of the wider network. Finally, a well-designed fee structure might disincentivize spammers. This three-layer approach to moderation creates an environment that is hostile to bad actors.

Of course, this all requires servers to act in a benevolent manner. Since servers are the most trusted actors in the network, they also have the most power to abuse the system. Servers may attempt to attack the network or particular nodes within it in a variety of ways, including by forging requests, spamming, stealing content, lying about reputation and votes, and more. For this reason, a single server should not be trusted to respond to all requests accurately or completely.

Implementers of both server and client applications are encouraged to validate event signatures and payloads, implement algorithms for identifying bad actors, and encourage users to widely publish votes on servers based on quality of service. If a server is identified as hostile to the network, a user can simply choose not to interact with it. Eventually, with no clients publishing to it, and no other servers willing to fulfill requests for content replication, the server will run out of content to publish.

Monetization & Incentives

Server costs can become significant, especially at the scale of a social network. While many alternative social networks are able to get by with independent server operators paying costs out of pocket, that's not a solid foundation to rely upon.

In the past, social media companies have monetized their services by selling ads on their platform. This option would be possible on Dogwood using a plugin and a purpose-built client, but is probably not worth pursuing since ads are unpopular, and so would probably not survive in an environment where other sources can provide the same content with a different model.

The most obvious solution would for servers to simply charge users directly. As the Bitcoin Lightning network matures, this is becoming more feasible. Services like Stacker News already exist that monetize in this way. This can take a variety of forms:

- Charge users to register their account with your server to reduce spam from new accounts
- Charge users for blob storage to reduce image spam and cover costs elsewhere
- Charge users for posts identified as more likely to be spam

- Charge users for a high rate of posting

This of course raises the question of who will pay for social media. The cost of participation will dampen the network effect by discouraging posting and joining. To overcome this, servers may choose to periodically pay a portion of their earnings to users based on the quality of their contribution to the quality of the local network, measured by upvotes or some other metric. This is the approach Stacker News takes as well — it is exciting to get paid for posting quality content. The incentive this creates for users to post quality content will in turn raise the quality of the network as a whole, accelerating the network effect.

Fees and payouts are published by servers in accordance with the Dogwood protocol, which also defines certain error messages a server may return in order to request payment for the action being taken. Instead of requesting payment for every action, servers may also maintain a small balance for user accounts to reduce friction.

Conclusion

Social media is an apparently simple thing — users, posts, likes — but hides a world of complexity underneath. Social media companies have to grapple with this complexity, managing, monetizing, scaling, censoring. In contrast, an open protocol creates an environment in which individual server operators, content producers, consumers, developers, and entrepreneurs collaborate and compete to create emergent solutions.

A hobby server operator might host a server to interact with his friends using a chat-based UI. His friends can cross-post content from their local community or the wider social web via their own reddit-like clients. Elon Musk might abandon Twitter to create a highly-scalable centralized infrastructure that interoperates with your local farmer's market's server. Politically-oriented groups might create a carefully curated "parallel economy", rejecting content that doesn't fit with their mission. Traditional websites might scrape comments from their social media posts and display them on their site. Cypherpunks can layer on peer-to-peer technology to improve the resiliency of the network.

All this can be accomplished one step at a time, resulting in a more interesting, weird, prosperous, healthier, safer internet, and all without reliance on or interference from — but potentially in partnership with — big tech and legacy media.