plebbit

Plebbit: A Serverless, Adminless, Decentralized Reddit Alternative

(Whitepaper v0.2.0)

Written by Esteban Abaroa

Published 16th September 2021

Abstract

A decentralized social media has 2 problems: How to store the entire world's data on a blockchain, and how to prevent spam while being feeless. We propose solving the data problem by not using a blockchain, but rather "public key based addressing" and a peer-to-peer pubsub network. A blockchain or even a DAG is unnecessary because unlike cryptocurrencies that must know the order of each transaction to prevent double spends, social media does not care about the order of posts, nor about the availability of old posts. We propose solving the spam problem by having each subplebbit owner run a "captcha service" node over peer-to-peer pubsub. Peers who fail too many captchas are blocked from pubsub.

Public Key Based Addressing

In Bittorrent, you have "content based addressing". The hash of a file becomes its address. With "public key based addressing", the hash of a public key becomes the address of the subplebbit. Network peers perform a DHT query of this address to retrieve the content of the subplebbit. Each time the content gets updated, the nonce of the content increases. The network only keeps the latest nonce.

Peer-to-Peer Pubsub

Pubsub is an architecture where you subscribe to a "topic", like "cats", then whenever someone publishes a message of topic "cats", you receive it. A peer-to-peer pubsub network means that anyone can publish, and anyone can subscribe. To publish a post to a subplebbit, a user would publish a message with a "topic" equal to the subplebbit public key (its public key based addressing).

Captcha Service over Peer-to-Peer Pubsub

An open peer-to-peer pubsub network is susceptible to spam attacks that would DDOS it, as well as makes it impossible for moderators to manually moderate an infinite amount of bot spam. We solve this problem by requiring publishers to first request a captcha challenge from the subplebbit owner's peer. If a peer or IP address relays too many captcha challenge requests without providing enough correct

captcha challenge answers, it gets blocked from the pubsub. This requires the subplebbit owner's peer to broadcast the result of all captcha challenge answers, and for each peer to keep this information for some time.

Note: The captcha implementation is completely up to the subplebbit owner. He can decide to prompt all users, first time users only, or no users at all. He can use 3rd party services like Google captchas.

Subplebbit Creation Process

Subplebbit owner starts a Plebbit client "node" on his desktop or server. It must be always online to serve content to his users.

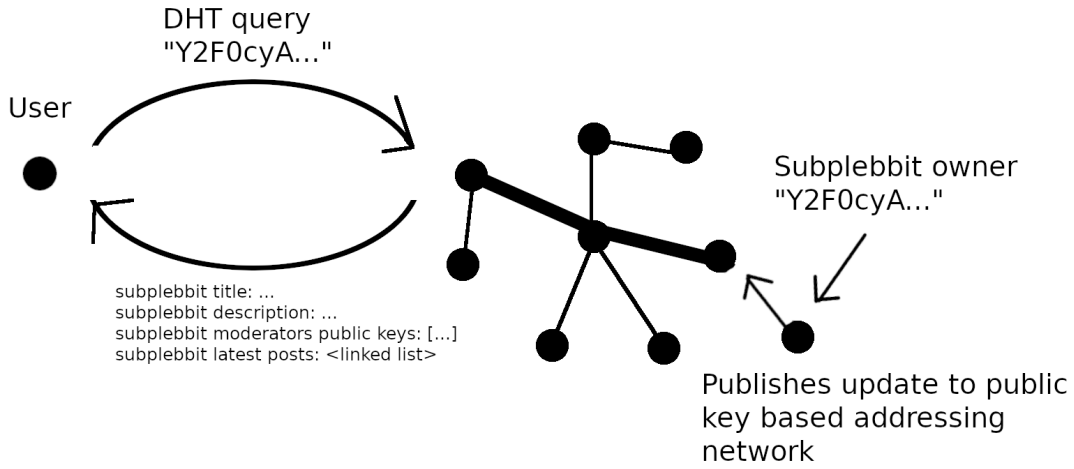
-He generates a public key pair, which will be the "address" of his subplebbit.

-He configures captcha options, like how often and what kind of captchas to show.

-He publishes the metadata of his subplebbit to his public key based addressing. This includes subpebblit title, description, rules, list of public keys of moderators, etc.

Note: It is possible to delegate running a client to a centralized service, without providing the private key, which makes user experience easier, without sacrificing censorship resistance.

THE PROCESS OF READING THE LATEST POSTS ON A SUBPLEBBIT



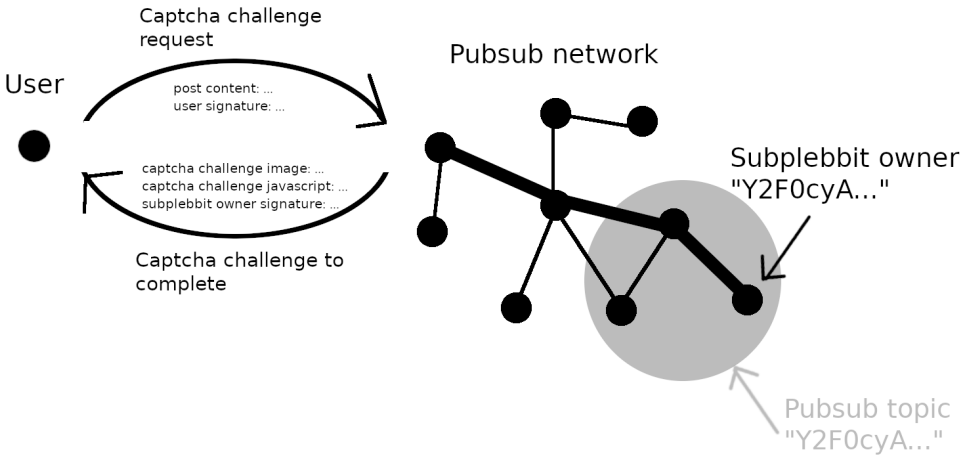
1. User opens the Plebbit app in a browser or desktop client, and sees an interface similar to Reddit.

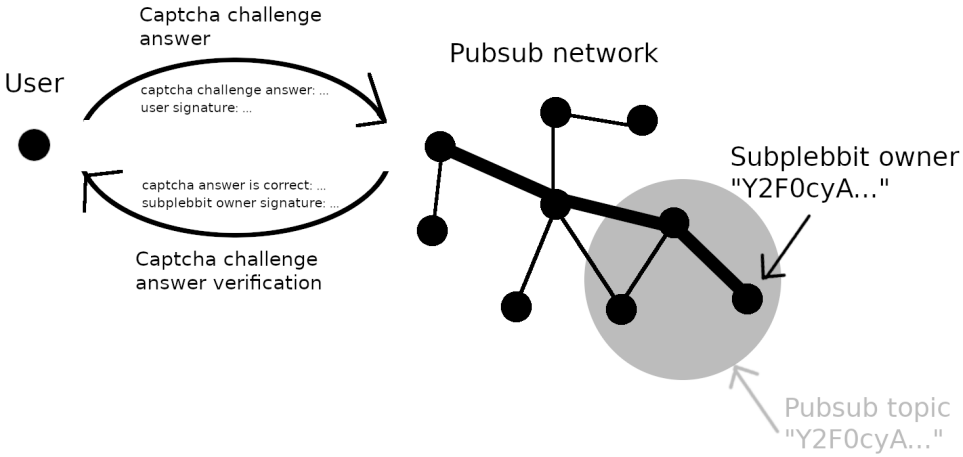
2. His client joins the public key addressing network as a peer and makes a DHT query for each address of each subplebbit he is a member of. The queries each take several seconds but can be performed concurrently.

3. The query returns the latest posts of each subplebbit, as well as their metadata such as title, description, moderator list and captcha server URL.

4. His client arranges the content received in an interface similar to Reddit.

THE PROCESS OF PUBLISHING A POST ON A SUBPLEBBIT





1. User opens the Plebbit app in a browser or desktop client, and sees an interface similar to Reddit.

2. The app automatically generates a public key pair if the user doesn't already have one.

3. He publishes a cat post for a subplebbit called "Cats" with the public key "Y2F0cyA..."

4. His client joins the pubsub network for "Y2F0cyA..."

5. His client makes a request for a captcha challenge over pubsub.

6. His client receives a captcha challenge over pubsub (relayed from the subplebbit owner's peer).

7. The app displays the captcha challenge to the user in an iframe.

8. The user completes the captcha challenge and publishes his post and captcha challenge answer over pubsub.

9. The subplebbit owner's client gets notified that the user published to his pubsub, the post is not ignored because it contains a correct captcha challenge answer.

10. The subplebbit owner's client publishes a message over pubsub indicating that the captcha answer is correct or incorrect. Peers relaying too many messages with incorrect or no captcha answers get blocked to avoid DDOS of the pubsub.

11. The subplebbit owner's client updates the content of his subplebbit's public key based addressing automatically.

12. A few minutes later, each user reading the subplebbit receives the update in their app.

(If the user's post violates the subplebbit's rules, a moderator can delete it, using a similar process the user used to publish.)

Note: Browser users cannot join peer-to-peer networks directly, but they can use an HTTP provider or gateway that relays data for them. This service can exist for free without users having to do or pay anything.

What is a "Post"?

Post content is not retrieved directly by querying a subplebbit's public key. What is retrieved is list of "content based addressing" fields. Example: latest post: "bGF0ZXN0...", metadata: "bWV0YWRhdGE...". The client will then perform a DHT query to retrieve the content. At least one peer should have the data: the subplebbit's owner client node. If a subplebbit is popular, many other peers will have it and the load will be distributed, like on Bittorrent.

Anti-Spam Strategy beside Captcha Service

The captcha service can be replaced by other "anti-spam strategies", such proof of balance of a certain cryptocurrency. For example, a subplebbit owner might require that posts be signed by users holding at least 1 ETH, or at least 1 token of his choice. Another strategy could be a proof of payment, each post must be accompanied by a minimum payment to the owner of the subplebbit. This might be fitting for celebrities wanting to use their subplebbit as a form of "onlyfan", where fans pay to interact with them. Both these scenarios would not eliminate spam, but they would bring them down from an infinite amount of spam, to an amount that does not overwhelm the pubsub network, and that a group of human moderators can manage. Proof of balance/payment are deterministic so the P2P pubsub network can block spam attacks deterministically. Even more strategies can be added to fit the need of different communities if found, but at this time the captcha service remains the most versatile strategy.

Improving Efficacy of Public Key Based Addressing

A public key based addressing network query is much slower than a content addressing based one, because even after you find a peer that has the content, you must keep searching, in case another peer has content with a later nonce (more up to date content). In content based addressing, you stop as soon as you find a single peer, because the content is always the same. It is possible to achieve the same speed in Plebbit, by having public key based addressing content expire after X minutes, and having the subplebbit owner republish the content after the same X minutes. Using this strategy, there is only ever one valid content floating around the network, and as soon as you find one peer that has it, you can deterministically stop your search.

Unlinking Authors and IP Addresses

In Bittorrent, an attacker can discover all the IP addresses that are seeding a torrent, but he can't discover the IP address of the originator of that torrent. In Bitcoin, an attacker can directly connect to all peers in the network, and assume that the first peer to relay a transaction to him is the originator of that transaction. In Plebbit, this type of attack is mitigated by having the author encrypt his comment or vote with the subplebbit owner's public key, which means that while the attacker can know a peer published something, he doesn't know what or from what author.

Conclusion

We believe that the design above would solve the problems of a serverless, adminless decentralized Reddit alternative. It would allow unlimited amounts of subplebbits, users, posts, comments and votes. This is achieved by not caring about the order or availability of old data. It would allow users to post for free using an identical Reddit interface. It would allow subplebbit owners to moderate spam semi-automatically using their own captcha service over peer-to-peer pubsub. It would allow for all features that make Reddit addictive: upvotes, replies, notifications, awards, and a chance to make the "front page". Finally, it would allow the Plebbit client developers to serve an unlimited amount of users, without any server, legal, advertising or moderation infrastructure.

If you would like to get involved in the development of Plebbit, we are currently hiring JS Devs, please contact me either on:

Telegram: @estebanabaroa

Discord: estebanabaroa#2853