Radio

no_independe@danwin1210.de

May 17, 2022

1 Introduction

It's shown a P2P social media called radio, based on IPV6 technology. It's called radio since every user will have his own "station" to send files, by using a randomly generated multicast channel. In the first sections I will try to be the least techinical as possible so that regular users can understand the beginning, but later I will show a more detailed view of the algorithm.

2 Informal Description

The algorithm works in the following way: Every user has an specific multicast address where he sends posts, people who follow can see the posts or post comments by receiving and sending trasmissions to this address respectively.

That by its own, won't need P2P, but as we will run on other problems, those will need to be dealt by relying on peers. For example, if some user doesn't receive the files of a person he follows because his machine is not plugged in when the files were sent, how does the person retrieve it later? He will need to "ask" to peers (people that follow the same user) to give all actions (data that encode the things that happened in the network) during the time the user was off.

Other thing that needs to be thought about is attacks by hackers to the network. In the case of an attacker who sends wrong files to a P2P network, the algorithm must have some way to deal with it, and the most common choice would be by taking a majority vote (the files that are sent the most must be true). This type of algorithm doesn't work in the case of an attacker who is able to gain control over the majority of the network (most of the peers belong to him). When that happens we have a 51% attack.

To remedy this type of an attack, some users who have been seeding for more time, will have receive more "trust" of its peers, that is, they will favour files sent by this user. In addition to that, people when signing up to an account will need a proof-of-work (POW) to be validated as a user (like minting).

Another problem, is how to identify each user? It's common practice for a web page to instead of storing a password of some user on a file, to store hashes of the password (a mapping that is hard to reverse), and the company knows that a password sent corresponds to the password of the user because the key sent matches the hash. We will adopt a similar idea and use hashes to represent passwords. Those passwords will be 128 characters autogenerated, therefore the user won't need to care about it. Those passwords and hashes will change everytime the user sends something to the network.

The next sections will show in more detail how each of those components were designed and how they can be combined.

3 IPV6 and multicast

The original idea of the IPV4 was to have an unique address to every computer, but since there are 4 billion addresses, in a world with 8 billion people and much more machines, it wasn't enough.

Since, it wasn't possible, it was adapted IPV4 to IPV6. The IPV6 address consists of 8 groups hexadecimal numbers, each having 4 digits. Therefore having

1

$$2^{4*8*4} = 2^{128}$$

Possible binary representations of an address. So for example, "abcd : abcd : abcd : abcd : abcd : abcd : abcd : abcd" would be a valid IPV6.

Some of the addresses are reserved for specific purposes. One of those types of address are designated to multicast channels. Multicasts channels are addresess devoted to streaming of data to multiple locations. Every person who listens on one of those addresses (hosts on that address) receives a message sent by someone on that address. The first two bytes of the address are ff, the third byte represents the lifetime of the address (0 for permanent and 1 for temporary), the fourth the scope (is it hosted only on the local network, is it global?). The multicast addresses, are all going to be with unlimited time and global, therefore all multicast addresses of the algorithm start with "ff0e".

4 Safe message

To identify the action of each user on the network, it was created the idea of safe message. A safe message, works as follows, every person who belongs to the same network, has for every user the hash of his key, and when a user sends a message, he sends together with his message the key of the old hash and a new hash. The key of the old hash certifies that this

user is the the only user who has the key. The new hash will replace the old hash, and the user who sent the message will have the corresponding key to that hash.

To be more specific the first 128 characters of the safe message would be the key of the old hash, the next 128 characters correspond to the new hash, the next character the protocol (this will be explained later on), and the next sequence of characters would correspond to the username and the account where the user is sending messages.

Notice that those safe messages will be sent through multicast channel, which means if some user who follows the account is not present, he won't receive the message, nor the new hash. That's why we need peers to send the activity of the network when the user logs back. This will be explained on section 8.

5 Protocols

To interpret and send the correct response to a message sent to a node, the algorithm needs some sort identification of the type of message. The identification will be called a protocol. Here's the list of what each number mean in a request of the application

1 Sign up request. Done through multicast channel

2 Sends a signal to all the networks that the user follows, that he changed the key of its own network. Done through multicast channel

3 Send a request to peer to become neighbour and receive actions. Done through a server hosted from computer.

4 Send file to all the followers on the network. Done through multicast

5 Sends a message back to peer B that user A received the request of user B to become peers. Done through a server hosted from computer

6 Sends files to the peer who requested. Done through a server hosted from computer

7 Sends comments through multicast.

8 Sends request to receive data of all the users with their hashes and the POWs they used. Done through a server hosted from computer

9 Sends message to the network of the user to block some user. Done through multicast

6 Account

Every account will be assigned an specific multicast channel. In this channel the user can send posts to those that are listening to the channel,

those will be the followers. Also the followers are allowed to comment on posts, and as duty, they need to send the files of the network when requested by a peer.

To avoid conflicting usernames, the algorithm will generate 10 random characters at the end of the username, when creating a new username, since there's no commonly shared database with all the usernames.

7 Sign up

To avoid creating unlimited number of accounts, when sending a request to join a network, the user will also send a proof of work. In this case, will be a key such that when mapped to a sha512 hash, the first 3 characters will be 0. User also generates a key, and send the hash of the key to the network to perform safe messages later.

8 P2P Network and 51% attack

The peers will be responsible for sending the actions that happened on the network as stated before. The algorithm will be a bit similar to those used for proof-of-stake in crypto.

Even though is very hard to deal with such a powerfull attack, it's proposed an algorithm to remedy the problem. The algorithm will favour files sent by peers who seed for more time, but before showing the algorithm, it's solved the following problem and given some highlights: Given that people 1, 2, 3..., n have probability of sending the correct action of

2

$$p_1, p_2, ..., p_n$$

Given the set

3

$$A$$

Of people who agree on some action is the correct one, what's the probability the action is correct? It is the probability of people who agree being correct and people who disagree being incorrect, given that those two groups of people will either be right or wrong about the file. Therefore the probability is

4

$$P_A = \frac{\prod_{x \in A} p_x \prod_{x \notin A} (1 - p_x)}{\prod_{x \in A} p_x \prod_{x \notin A} (1 - p_x) + \prod_{x \in A} (1 - p_x) \prod_{x \notin A} p_x}$$

On the numerator the first product correspond to the probability of people who agree being correct, the second product to the probability of people who disagree being incorrect. From that it follows the formula. To simplify, define

5

$$p := \prod_{x \in A} p_x$$

6

$$p' := \prod_{x \in A} (1 - p_x)$$

7

$$a := \prod_{x} p_x$$

8

$$b := \prod_{x} (1 - p_x)$$

Since

9

$$\prod_{x \in A} p_x \prod_{x \notin A} (1 - p_x) = \frac{\prod_{x \in A} p_x}{\prod_{x \in A} (1 - p_x)} \prod_{x} (1 - p_x) = \frac{p}{p'} b$$

And

10

$$\prod_{x \in A} (1 - p_x) \prod_{x \notin A} p_x = \frac{\prod_{x \in A} (1 - p_x)}{\prod_{x \in A} p_x} \prod_{x} p_x = \frac{p'}{p} a$$

Therefore

11

$$P_A = \frac{(pb/p')}{(pb/p') + (p'a/p)} = \frac{pb}{p'} \frac{pp'}{(p^2 b + p'^2 a)} = \frac{p^2 b}{p^2 b + p'^2 a}$$

Notice the very important fact for later, consider for some person k. Call

12

$$\hat{p} := \prod_{x \in A - \{k\}} p_x$$

13

$$\hat{p}' := \prod_{x \in A - \{k\}} (1 - p_x)$$

14

$$\hat{a} := \prod_{x;x\neq k} p_x$$

15

$$\hat{b} := \prod_{x;x\neq k} (1-p_x)$$

We have

16

$$p = \hat{p}p_k$$

17

$$p' = \hat{p}'(1-p_k)$$

18

$$a = \hat{a}p_k$$

19

$$b = \hat{b}(1-p_k)$$

Therefore

20

$$P_A = \frac{\hat{p}^2 p_k^2 \hat{b}(1-p_k)}{\hat{p}^2 p_k^2 \hat{b}(1-p_k) + \hat{p}'^2(1-p_k)^2 \hat{a}p_k}$$

Consider that using formula (11), but removing the person k from the decision, we have probability of action being correct is

21

$$\hat{P}_A = \frac{\hat{p}^2\hat{b}}{\hat{p}^2\hat{b} + \hat{p}'^2\hat{a}}$$

Check that

22

$$p_k = 50\% \rightarrow P_A = \frac{\frac{1}{8}\hat{p}^2\hat{b}}{\frac{1}{8}(\hat{p}^2\hat{b} + \hat{p}'^2\hat{a})} = \hat{P}_A$$

23

$$p_k \rightarrow 100\% \Rightarrow P_A \rightarrow \frac{\hat{p}^2 p_k^2 \hat{b}(1-p_k)}{\hat{p}^2 p_k^2 \hat{b}(1-p_k) + \hat{p}'^2(1-p_k)^2 \hat{a}p_k} =$$

$$= \frac{\hat{p}^2 p_k^2 \hat{b}}{\hat{p}^2 p_k^2 \hat{b} + \hat{p}'^2(1-p_k)\hat{a}p_k} =$$

$$= \frac{\hat{p}^2 p_k^2 \hat{b}}{\hat{p}^2 p_k^2 \hat{b}} = 1$$

So while the action of person k with probability 50% of telling the truth has no influence in determining if the action is correct, a probability of 100% completely determines that the action is correct. Keep that in mind.

If we knew the probabilities, we could use the formula above to calculate the probability the action being correct, and in the case of an attack choose the one with greatest probability. Moreover, the formula above gives a linear time algorithm on the number of people (first is calculated "a" and "b", then for each action we compute "p" and "p' ").

It's going to be shown a procedure to give an estimate of that probability, such that the people that have more influence on the decision of some user are the ones who shared more files, during more time. The algorithm runs locally on every computer on the network, meaning everyone will most likely have different sets of probabilities for every individual. Every user starts with a 50% of probability and every time a user sends the correct action (determined using the greatest probability of being the correct one), the probability of the user increases according to a map that has maximum at 1, otherwise it decreases with minimum at 50%.

If the user sends the correct file and his probability is

24

$$p_u$$

Then the probability will be later

25

$$(1 + 19p_u)/20$$

Otherwise

26

$$(1 + 18p_u)/20$$

The reason why is the following. Given the map

27

$$(1 + ax_n)/b = x_{n+1}$$

The fixed point of the map will be such that

28

$$(1 + ax)/b = x \to 1 + ax = bx \to 1 = (b - a)x \to b - a = 1/x$$

For fixed point x=50% we would have b-a=2, for a fixed point at 100%, we would have b-a=1. Then it was chosen an arbitrary value for b=20.

Therefore the algorithm works as follows: "People who share the "correct" action will increase their score (probability of choosing the correct

action), people who didn't will decrease. The correct file is determined by the formula 11, based on people's score. In the case of a tie, it's chosen the action that most users sent".

If a person decides to create many users at one specific moment to make the network send wrong files, it will need to deal with the problem that those users have very low influence at the beginning, meaning the attack will need to be sustained for much more time to work properly (the attacker needs to make his score high enough for enough number of users to start the attack).

9 Contributing

Those are areas that I found that are of interest to the development of the program or maybe crucial.

Security The peers send files directly to each others, those could be potentially infected with a virus, therefore, when receiving files, the server needs some type of scan to filter certain type of code.

DDoS Even though it's still security, it needs special attention. The algorithm needs to have some defense against DDoS attacks on server application and multicast channel

Design Needs a better design, if possible that doesn't use javascript and can run locally.

Streaming The multicast is already appropriate for streaming, therefore it would be appreciated ideas of how to add the feature, so would any help.

Crypto-Wallet Addresses This could serve as a revenue for content creators. Any ideas of how to add the feature would be much appreciated, so would any help.

Images Currently the algorithm is not being able to send images, only text files in the multicast channel, therefore needing some adaptation.