Discoverability and Moderation for SSB frontends

Second Distributed Systems 2 Project

Massimiliano Girardi 211999

I. Abstract

The rise of social networks was dominated by the centralized approach that is now a defacto standard that is used by all big platforms. There are alternatives to this the model, that although right now less prominent, may in the future play a role in a free-er environment for creators and consumers. In this paper, the provided implementation of such a system is discussed, along with some considerations and a description of the methodology used to both visualize and test the system. This product is developed with the aid of the RePast Eclipse plugin. The work discussed in this paper is based on [1].

II. Introduction

The major problem of censorship that has stroke social media, along with several changes that were made to keep the platforms economically viable to maintain, like non-chronologically ordered feeds, has led many to the research of a free-er (as in freedom) content delivery system. The problem with popular centralized systems is cost and the nature of centralization, that in a world where social networks are becoming the spaces where the public opinion is formed, are too vulnerable to an often opaque hierarchy. One way to solve this problem, and the one that was explored in the following paper, is based on the work of [1] that proposes a secure decentralized gossipping based algorithm that tries to give a strong foundation for an append-only log-based social platform. In the following, a simulation environment was created to test several properties of the algorithm and suggest improvements to it.

III. Code Structure

The project is designed to be extendable and modifiable, therefore in the following chapter, a description of each component is provided along with its purpose. The design principle is typical of the Object-Oriented (OO) paradigm. For each of the following classes is provided with an example implementation that is then used to develop the rest of the discussion it this document. Regardless of this fact, the design of each component can, and should, depending on the scenario that has to be simulated, be altered, even considerably.

A. Messages

The communication layer is based on messages that are exchanged by the hosts to communicate. Most messages are taken from the actions defined in the original paper, nevertheless here is an introduction to each item.

- **Message** A generic implementation of a Message is provided to establish a basic version of the datatype that can be extended by application-level subclasses without having to be bothered to learn how the specific implementation of the algorithm works. The idea is that a subclass calls the constructor of this class by encoding in some way the content that has to be shared so that it can be retrieved by a receiving peer.
- **BlockMessage** The message that communicates the block event to the peers
- **FollowMessage** The message that communicates the following event to the peers
- **PostMessage** The message that communicates the creation of a local post and the content
- **UnblockMessage** The message that communicates the unblock event to the peers
- **UnfollowMessage** The message that communicates the unfollow event to the peers

B. Styles

The content of this directory is used to visualize the internal status of the simulator on the GUI. There are many options to customize the product from the configuration file, but editing or overriding these components offers an unrivalled capability in customization.

- **Network Relay Layout** This component is used to present a network of relays by visualizing also their relative position
- **Relay Network Style** This component is used to visualize a relay on the map.
- **Colored Edge** This component is used to visualize a connection on the network component, it can be changed in colour.

C. Actors

This directory contains all of the active participants of the network.

- **Relay** This is the Relay superclass, that implements most of the common functionalities
- **Concrete Relay** This component is the Relay implementation that implements most of the algorithm. This is also used by the GUI to simulate an execution.
- **Test Relay** This component is a Relay implmentation used for automatic testing.
- **Log** This component is rappresents a write-only log.

IV. Moderation

An excuse that is usually taken by many of the social media administrators to justify removing content and censoring personalities on the platforms, is to serve the greater good and

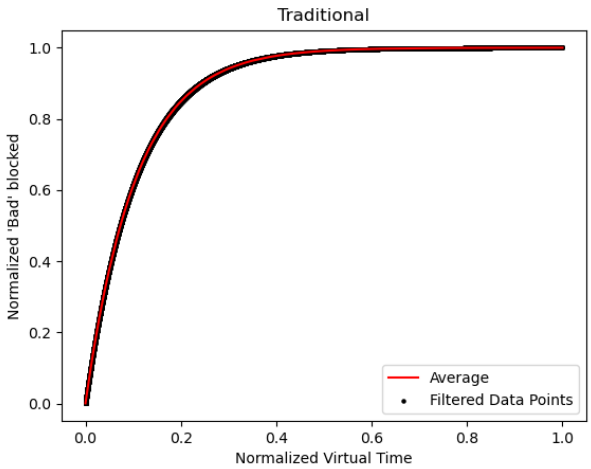

Fig. 1. Traditional blocking method performance

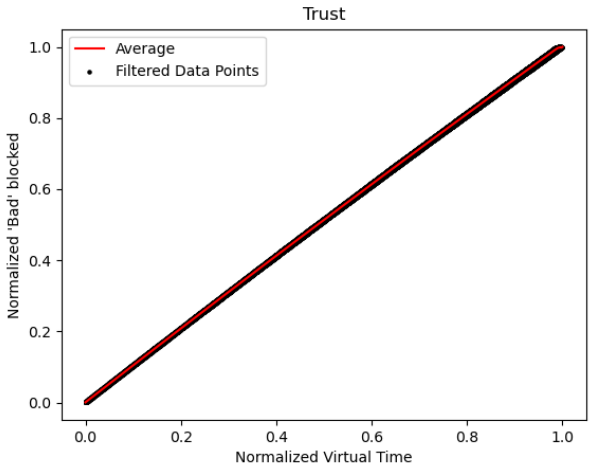

Fig. 2. Trust chain blocking

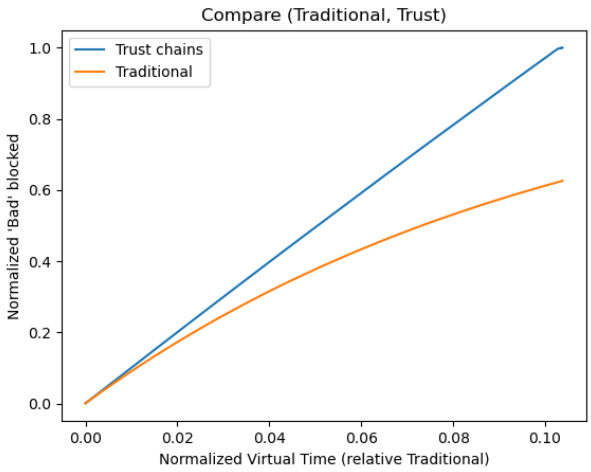

Fig. 3. Trust chain blocking VS Traditional blocking

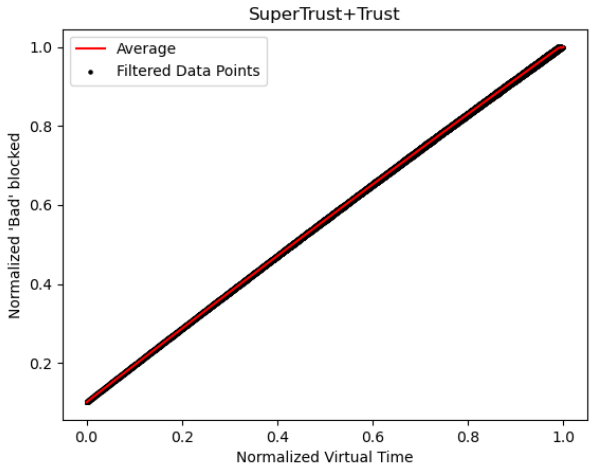

Fig. 4. SuperTruster performance impact

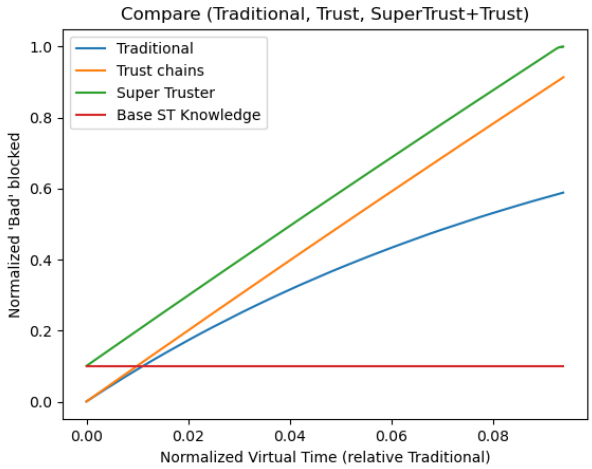

Fig. 5. SuperTruster VS Trust chains VS Traditional

isolate the ideas that are deemed outside of the social norm of the country the platform user base is mostly localized. The process of limiting the freedom of expression on the sites is usually carried out by massive operations of manual reviewal of the reports filed by the users themselves, usually aided by ML algorithms. The problem with this process, along with the ML algorithm, is that the platforms utilize them to also propose content to the users. This has been documented in the last years to just create a phenomenon well known as ”bubbles”. A bubble is a set of thinking alike individuals, that gather themselves together and are quite opaque for the outside world. This nature of the bubble makes reports few, and further apart, limiting the probability of censorship. This would probably kill off a community in a normal environment, where content propagation is organic, and therefore, closeness is not an evolutionary advantage. Sadly here the platforms have made another assist to this bubbles, that is ML-aided content propagation. This decision tends to favour the growth of these bubbles by showing them only to people that are unlikely to

disagree with the content.

An interesting aspect of the protocol at hand that we studied was how the gossip-based propagation and block lists could affect and reduce the formation of this phenomena. The interesting finding that will be further discussed in the following is that, without resorting to a centralized approach, the formation of bubbles is really difficult. The problem arises from the fact that a node can only communicate with the one that is logically, or in our example implementation, geographically close to it. There is no difference between these two approaches, but the former is easier to visualize than the latter. For a bubble to form, two nodes that are interested in forming it have to be in proximity (range) to each other. If they are not, then there must be a chain of nodes that connect the two broadcast domains that they are a member of. Blocklists, and even more block lists subscription, break these chains, isolating and fragmenting their domain.

This approach to moderation, that makes each node the master of its view of the system is indeed limited by the effort that the users are willing to put into pruning the domain. The true boost to the approach is the subscription to blocklists. The approach that was explored in the simulator is similar to the concept of a shared block list. A user might decide to trust another user. If this happens, the blocking and unblocking actions are mimicked among trusted peers. Of course, the trust relationship is not always mutual, but along trust chains, the exclusion of users is spread fast across the network. This system is vulnerable to social engineering, a malicious host might try to infiltrate a trust chain and poison the block list. For this reason, users should be discouraged to give their trust to each other, warning them of this possibility. This is though a discussion to be left to the GUI designer though, along with the probable need for super trusted users.

A super trusted user is a relay that is either controlled by the GUI designer, or it is controlled by some very reputable users among the community. The role of such a user is to have a small block list that is reserved for the worst offenders. The idea is that by default a super trusted user can be trusted, therefore excluding the worst members from reaching anyone by default. A super trusted user should not trust anyone.

A. Evaluation

Evaluating moderation is a tricky business, that could lead to many philosophical questions that will not be addressed in the following. To evaluate the system the test scenario is described in the pseudo-code VII. Using the provided parameters, each of the following methods was evaluated. For the purpose of this tests, 'bad' actors are pre-determined.

1) Traditional User Discover for itself: This is the worse technique because it is a lack of a shared methodology, but it was tested to provide a comparison therm for the other propositions. As it can be seen in Fig.1 the convergence time for the blocklists, in a statical network is long, users are exposed to 'bad' messages for many network cycles. This could scare users away from the community so it has to be avoided.

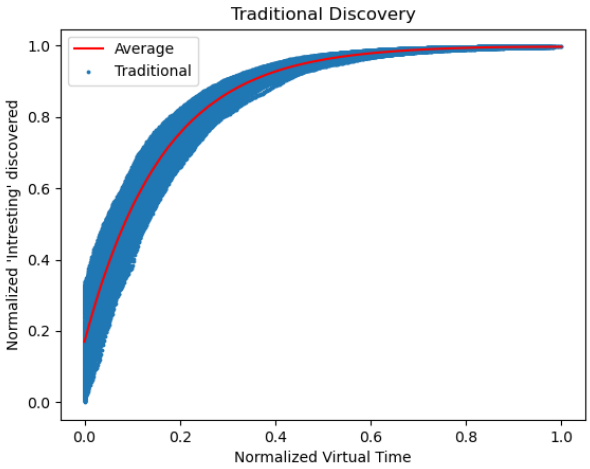

Fig. 6. Traditional discovery

2) Trust chains: As it can be seen in Fig.2, on average the trust chain reduces severely the convergence time and this depends linearly on the maximum chain size, this is not surprising for two facts: Longer trust chains are propagating different blocks faster, uniforming themselves. Secondly, the spreading effect does also influence the leaves. Of course, the longer the chain is, the easier it is for someone to break the trust and ruin the chain altogether. Comparing the average convergence time of a small size chain and the traditional method the difference is evident, as proposed in Fig.3

3) SuperTruster: The super trusted user gives a constant boost to the discovery of bad actors in the community, as it can be seen in Fig.4. This is not surprising, the super truster was kept with a constant knowledge during the tests and it should be similar to what we are envisioning in reality. The blocking from a super trustee should be a rare event, and as such, the blocklist can be seen as (almost) constant coming from this host.

Even if the impact is similar to what was expected before conducting the tests, there is an argument to be made that this actor has his place in the protocol. There could be several super trusted users, that each specifically targets users that discuss several topics that might be controversial or sensible for some users, this would surely make for a way better experience in the system.

Another important fact that arises from the comparison in Fig.5 is that the user is exposed for a fraction of the time to whom is deemed a worse offender in the network, therefore severely limiting their discoverability and possibility to interact with others.

It must be underlined that this is not a centralized solution to the problem, a super trusted user is nothing more than a normal user, therefore it participates in the network similarly to many others.

V. Discoverability

This section will introduce a suggestion to aid discoverability for content catered mainly to new users, that could be an

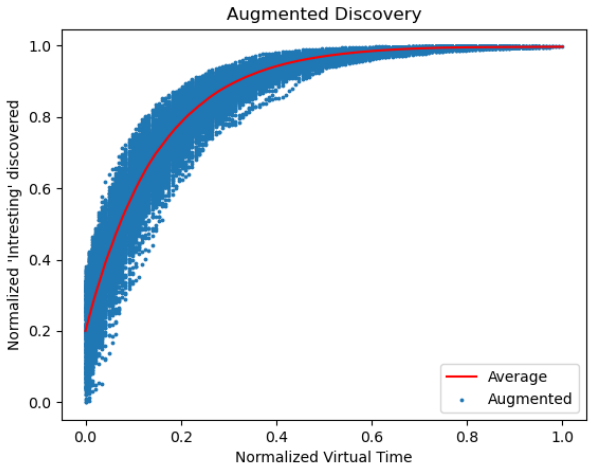

Fig. 7. Augmented discovery

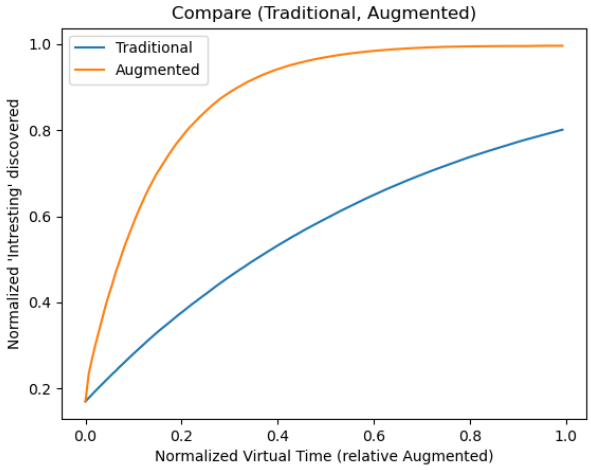

Fig. 8. Traditional compared to Augmented

interesting addition to the SSB protocol, and was therefore tested. The idea is that, given that messages are not cast in this implementation using a multicast, but broadcast, a relay is destined to receive multiple messages that are not of its interest. This messages are not appended to the local log and are basically ignored. This fact is mainly due to the design of append-only logs, where a message is not verifiable if the predecessor is not known.

It would be interesting if sometimes the posts, even if it is impossible to verify their origin were shown to the user anyway. This mechanism would increase the number of posts that are shown to a user, with the possibility of following the author. This feature should be optional and does not really modify the protocol, just the way a relay acts when receiving a message. Of course, this should only apply to Post Messages.

The idea is that there are basically two scenarios that can arise from this feature:

- **The user is interested in the content** This is the best outcome, the user discovers a new peer that produces interesting posts, adding him to the following list. Eventually, all posts from this new peer will be loaded, and the protocol will continue as usual.
- **The user is not interested in the content** The user will just keep on scrolling the feed.

Of course, this 'suggested' posts should be clearly marked as such, and a tool-tip should clearly state that the actual origin of a suggested post is not verifiable. This concept could be exploited by bad actors, but the point is that, by preventing any non-positive action to be taken on a suggested post, there is basically nothing to lose from the scenario. The only exploit that is possible at this point is to post something in the feed impersonating someone else, for that single post alone. This is, of course, possible but it is not a big deal, given that firstly the post is clearly marked as suggested, secondly, this is quite an unlikely event to happen. Given that suggested posts are not propagated by those who visualize them and all the others propagate them (and therefore verify them) only if they already follow the author, for the described exploit to work the bad actor must be in the same broadcast domain of the victim and should produce a considerable quantity of messages to increase its odds.

As shown in Fig.6, the traditional tends to converge to a maximum interest coefficient, defined as a relative number of follows marked as interesting by the user, after some pruning, from initial knowledge of the network. The convergence is steady and monotonic because in this case no new follow are issued by the actors involved, therefore the user has no way to discover new peers.

Similarly, in Fig.7, the discovery rate seems to converge to the maximum value, but it has to be noted that the growth, in this case, seems less regular. This is because even in a 'steady' situation follow wise, there is still a chance to discover new peers in the community that may be of interest. This of course also means that sometimes someone who was first considered as interesting is in fact not, so the growth is not monotonic.

The real revelation comes from the comparison of the two methods. As proposed in Fig.8 the convergence of the augmented method is way faster than the traditional one. This is surprising and should be investigated further. It seems, although no further analysis was conducted, that the discovering of new peers does not only lead to a faster convergence, and to a relatively higher number of interesting users, but the variation is very notable. It is difficult at the point of writing to determine how much this has to do with the parameters of the network and how much has to do with the proposition itself but this preliminary result seems promising.

VI. Testing

Testing was carried out using a proprietary engine that allows defining a testing script and some metrics to evaluate a system giving it the startup parameters. The basic idea is that given a tuple of parameters there is a hidden parameter in every simulation that involves random number generation, that is the seed. The seed is a fundamental input for the simulation that is often overlooked but that could lead the research to get some wrong conclusion from a set of output. The problem is

that, once a simulation is carried out, the outcome might be influenced by a negative (or positive) sequence of randomly fired events. To avoid this possibility each of the simulations is carried out multiple times forcing different seeds. The outputs are then averaged among the different seeds to have the best representation of the expected result. This results also in a shadow beyond the graph line, that is the average, that represents the range of the results in the specific x point. The cloud is not always proposed in the graphs because it sometimes gets in the way of comprehension.

A. Moderation

The graphs in the moderation section were extracted from the data extracted from multiple simulations.

The data, as it can be seen by the lack of outlier, was filtered to exclude points that were too far away from the average.

The Y-axis proposed the 'Normalized Bad blocked' metric, that is calculated by averaging the number of blocked peers in the network over the number of actually 'bad' actors present. It has to be underlined that it was assumed that there are no false positives in the detection mechanism.

The X-axis proposes 'Normalized Virtual Time'. This metric is calculated by re-synchronizing the activations of each replica (so that all peers in an x point have done the same applicationSteps) that are then normalized (among each simulation).

The comparison graphs feature on the X-axis a slight difference, the X set of each metric is truncated to the first who gets a Y value of (close to) 1. The displayed x value on the end of the scale refers to the time of a simulation on a traditional algorithm.

B. Discoverability

The data for the graphs in this section was generated, due to time constraints, with the aid of a Markov chain that closely mimics the behaviour of the real network.

The graphs proposed are generated using the same methodology and language as the others, therefore the axis has mostly the same names.

VII. Appendix

The test language is based on python, therefore should be easy enough to understand. This tests along with the tests engine will though not be provided along with the simulator because of licensing issues.

```
#lin(low, high):
#  defines a linear space betwenn low and high
#norm(...): normalizes a dataset
#compare(...): compares ... datasets

RELAY_COUNT = ...
PACKETLOST = 0
BAD_RELAY = RELAY_COUNT * lin([.1, .3])
KNOWN_NETWORK = True
PROBABILITY_MARK_AS_BAD = lin([.1, .3])
MAXCHAIN_SIZE = lin([.1, .3])
SUPERTRUSTER_KNOWLEDGE = lin([.1, .3])
INTERESTING = choose(..., set of i : RELAY_COUNT)
      * RELAY_COUNT

BLOCK data = { ... } #dataset java definition
DISC data = { ... } #dataset from csv

for BLOCK plot:
  #standalone graphs
  Traditional :
    norm('Bad' blocked, outcast=remove),
    | norm(Virtual Time)
    .average.dp
  Trust :
    norm('Bad' blocked, outcast=remove),
    | norm(Virtual Time)
    .average.dp
  SuperTrust :
    norm('Bad' blocked, outcast=remove),
    | norm(Virtual Time)
    .average.dp
  #compare graphs
  compare(Traditional, Trust).legend
    .average.truncate
  compare(Traditional, Trust, SuperTrust).legend
    .draw(
      baseline(SuperTrust. "Base ST Knowledge")
    ).average.truncate

for DISC plot:
  "Traditional Discovery" td :
    norm('Intresting' intr),
    | norm(Virtual Time)
    .average.dp.legend
  "Augmented Discovery" ad :
    norm('Intresting' intr),
    | norm(Virtual Time)
    .average.dp.legend
  compare(td, ad).legend
    .average.truncate
```

VIII. Future Work

The following section is subdivided into two of the activities that according to us are more promising and interesting.

A. Discovery

The new discovery option seems very promising, and a good complement to the already proposed solutions (hubs) that are in place in SSB. It would be really interesting to integrate this mechanism in the simulator to see how it behaves and, similarly, in the already existing SSB clients.

B. Simulator GUI

In this research, the main focus was to develop and provide an API for the simulation of the SSB protocol. There are many things that were left out due to time constraints, but one important thing was the simulator GUI. This instrument was not really developed all that much but could be an important asset in order to display the network in real-time. This was not necessary to conduct the data analysis but could be useful to someone approaching the project for the first time or who wants to experiment first.

IX. Conclusions

The work presented in this paper proves the effectiveness of trust chains in fast community moderation. Interestingly, a semi-centralized approach to moderation, that is super trusted

users, was proved to be an important tool for baseline rule enforcement and therefore is highly suggested for SSB frontends that want to be user friendly.

Another important asset that was explored in this research is the possibility to let 'unverified' and unknown posts sometimes permeate the feed in a front end application. This strategy seems to be very promising and should be investigated further.

The developed simulator that was used for most of the study is undoubtedly a complete and strong implementation of the algorithm in [1], although as cited in future work, it might need some further work in the data visualization part.

References

[1] Christian Tschudin Anne-Marie Kermarrec Erick Lavoie. *Gossiping with Append-Only Logs in Secure-Scuttlebutt*.