Exploring Content Moderation in the Decentralised Web: The Pleroma Case

Anaobi Ishaku Hassan
Queen Mary University of London
i.h.anaobi@qmul.ac.uk

Aravindh Raman
Telefonica Research
aravindh.raman@telefonica.com

Ignacio Castro
Queen Mary University of London
i.castro@qmul.ac.uk

Haris Bin Zia
Queen Mary University of London
h.b.zia@qmul.ac.uk

Emiliano De Cristofaro
University College London
e.decristofaro@ucl.ac.uk

Nishanth Sastry
University of Surrey
n.sastry@surrey.ac.uk

Gareth Tyson
Queen Mary University of London
g.tyson@qmul.ac.uk

ABSTRACT

Decentralising the Web is a desirable but challenging goal. One particular challenge is achieving decentralised content moderation in the face of various adversaries (*e.g.* trolls). To overcome this challenge, many Decentralised Web (DW) implementations rely on *federation policies*. Administrators use these policies to create rules that ban or modify content that matches certain rules. This, however, can have unintended consequences for many users. This paper presents a first study of federation policies on the DW, their in-the-wild usage and the impact they have on users. We identify how this policies may negatively impact "innocent" users and outline strawman solutions to avoid this problem in the future.

ACM Reference Format:
Anaobi Ishaku Hassan, Aravindh Raman, Ignacio Castro, Haris Bin Zia, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. 2021. Exploring Content Moderation in the Decentralised Web: The Pleroma Case. In *Proceedings of CoNEXT '21.* ACM, New York, NY, USA, 8 pages. https://doi.org/TBA

1 INTRODUCTION

The "Decentralised Web" (DW) is an evolving concept, which encompasses technologies aimed at providing greater transparency, openness, and democracy on the web [5]. Today, well-known social DW platforms include Pleroma, Mastodon (microblogging services), Hubzilla (cyberlocker) and PeerTube (video sharing platform).

In the above examples, individuals or organisations are able to install, own and manage their own servers, usually referred to as **instances** [4, 31]. Instances are independent and participants in the community must register with specific instances. For example, in the case of Pleroma (a microblogging service), an instance could be created within a company to provide a platform for employees to interact. To enable users across such instances to interoperate, **federation** protocols allow information and interactions to flow across DW instances to create a larger interconnected community This creates a physically distinct set of servers, yet allows users to follow each other regardless of which instance they register with.

A key selling point of the DW is the promotion of free speech, outside of the remit of large tech companies. Although appealing, this decentralised form of management creates new challenges [18]. For example, as popular centralised social platforms like Facebook and Twitter continue to clampdown on hateful and violent communities, some of these communities have migrated [28] to DW instances where moderation and regulation is more difficult to enforce (*e.g.* Gab [29]).

In contrast to centralised services (*e.g.* Twitter), DW moderation is usually performed on a per-instance basis. Specifically, instance administrators enforce policies within their own instance to moderate the content coming from other federated instances. For example, administrators of one instance can **reject** (*i.e.* block) any material from other instances that match certain criteria. This instance-based approach shifts the moderation responsibility to administrators who need to answer questions such as: What policies should be applied and to which instances? How much effort should be put in moderation? What is the collateral damage of the policies (*i.e.* while a minority of users might cause a policy, this will affect the rest of the "innocent" users of that same instance)?

To explore these questions and propose solutions, we focus on one prominent DW platform: **Pleroma**. In contrast to other DW microblogging platforms, Pleroma instances make their moderation policies public through an API. Exploiting this, we collect a large-scale dataset covering a period of 5 months. This includes 1298 instances, 111k users, 24.5m posts, associated metadata and, importantly, the 46 different policies imposed by the instances.

Using this data, we explore the types of policies imposed by administrators. We find that that moderation affects the overwhelming majority of the users: 97.7% users and 97.8% posts are impacted by policies. The `reject` action is most popular, affecting 86.2% users and 88.5% posts (Section 4). This brute-force policy blocks entire instances, even though only a subset of users might be misbehaving. To investigate the collateral damage of this, we study user's toxicity using Google's Perspective API (Section 5). While toxic users are the likely target of instance blocking, we find that 95.8% of the users

CoNEXT '21, December 7–10, 2021, Virtual Event, Germany

ACM ISBN TBA…$TBA
https://doi.org/TBA

blocked are not toxic. We conclude by proposing some strawman solutions to reduce collateral damage.

2 PLEROMA OVERVIEW

Pleroma Primer. Pleroma is a lightweight decentralised microblogging server implementation, whose user-facing functionality is similar to Twitter. In contrast to a centralised social network, Pleroma is a federation of servers (aka **instances**), which interlink to share content. Through these instances, users are able to register accounts and publish posts, which will appear on follower timelines. These followers can either be on the same instance or another (federated) instance.

Federation. We refer to users registered on the same instance as **local**, and users registered on different instances as **remote**. A user on one instance is able to follow another user on a different instance. Note that a user registered on their local instance does not need to register with the remote instance to follow the remote user. Instead, a user creates a single account with their local instance. When the user wants to follow a user on a remote instance, the local instance subscribes to the remote user on behalf of the local user, thereby federating with the remote instance. This process is implemented using an underlying subscription protocol (ActivityPub [1]) that allows instances to federate with each other.

Fediverse. The resulting network of federated instances is referred to as the fediverse. The fediverse includes instances from Pleroma as well as instances from other platforms that Pleroma can federate with (*e.g.* Mastodon) because they support the same subscription protocol (*i.e.* ActivityPub). Accordingly, Pleroma instances can federate and target its policies at non-Pleroma instances (*e.g.* Gab from Mastodon).

Policies. Instances in the fediverse federate with each other and federated instances can target each other with policies. Policies affect how instances federate with each other through different rule-action pairs. These allow certain actions to be executed when a post, user or instance match pre-specified criteria. We refer to each of these rule-action pairs within a policy as **actions** (*e.g.* the `SimplePolicy` has multiple actions such as `media removal` and `reject`).

Note, some policies are in-built to the Pleroma software package. Instance administrators can enable ("switch on") one or more policies. Some of these policies are enabled by default when a new Pleroma instance bootstraps. Additionally, administrators can craft new policies if they require specific functionalities that are not covered by the in-built policies.

3 DATA COLLECTION

Instances. Our measurement campaign covers the period between the 16 December 2020 and 24 April 2021. We first compile a list of Pleroma instances by crawling the directory of instances from distsn.org and the-federation.info. We then capture the list of instances that each Pleroma instance has ever federated with using the Peers API.[1] This includes both Pleroma and non-Pleroma instances (Pleroma can federate with any instance of the fediverse, see Section 2). In total, we identify 9969 instances, out of which 1534 are Pleroma and 8435 are non-Pleroma (*e.g.* Mastodon).

We then collect metadata for each Pleroma instance every 4 hours via their public APIs.[2] We obtain the number of users on the instance, the number of their followers, the number of posts, the version of Pleroma, whether the instance is accepting new registrations, the enabled policies, the applied policies as well as the instances targeted by these policies, and other meta information. From the 1534 Pleroma instances, we are able to gather data from 1298 instances (84.6%). For the remaining 236 instances: 110 are not found (404 status code), 84 instances require authorisation for timeline viewing (403), 24 result in bad gateway (502), 11 result in service unavailable (503) and 7 return gone (410).

User Timelines. Users have three timelines: (*i*) a *home* timeline, with posts published by the accounts that the user follows (local and remote); (*ii*) a *public* timeline, with all the posts generated within the local instance; and (*iii*) the *whole known network*, with *all* posts that have been retrieved from remote instances that the local users follow. Note, the *whole known network* is not limited to remote posts that a particular user follows; instead, it is the union of remote posts retrieved by all users on the instance. The whole known network timeline is an innovation driven by the decentralised nature of instances: it allows users to observe and discover posts by remote users.

Out of the 1298 Pleroma instances, we gather all posts from 796 instances (119 instances had no posts and the public timeline of the remaining 38.7% instances were not reachable). We gather data using the public Timeline API.[3] This timeline covers all posts shared on each instance. This allows us to collect 14.5M (including federated posts) public posts out of 24.5M posts, covering 91.7K users. Note, from the 1,298 instances we are able to crawl, we discover a total of 111K unique users. 48.7% of users published at least one post.

Harmful Classifications. Generally, instance administrators target moderation policies/actions against other instances that have been perceived to post harmful content or violate their "Terms of Service". For any instance that has at least one `reject` action targeted against it (see Section 4.1), we annotate all of its posts with harmful/non harmful labels (15.8% of all instances). We annotate the posts using Google's Perspective API [7, 24]. The Perspective API scores text based on the perceived impact it might have on a conversation [20]. The scores represent the probability that a human annotator would reach the same conclusion and is represented between 0 to 1 for a range of attributes [8, 21]. In this paper, we classify posts across three attributes: toxicity, profanity and sexually explicit content. Perspective results have been found to be similar to human annotators [9, 17, 27, 28], it is widely used in production environments (*e.g.* New York Times) [22] and it is continually maintained and updated [13].

We label a *post* as harmful if it receives a score of 0.8 or above in any of the three attributes (toxicity, profanity and sexually explicit). This threshold is based on recommendations from the developers of the Perspective API [23] and related literature [3]. Finally, we classify a *user* as harmful when the average of all the user's posts

[1] <instance.uri>/api/v1/instance/peers

[2] <instance.uri>/api/v1/instance/

[3] <instance.uri>/api/v1/timelines/public?local=true

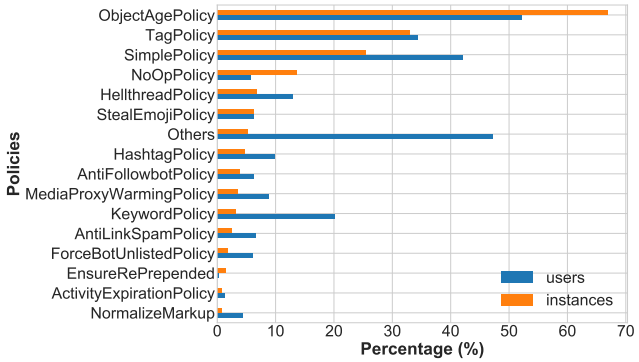

Figure 1: The top 15 policy types and the percentage of instances that use each policy (sorted by the percentage of instances). We also include the percentage of the global user population on the instances that use each policy. We represent the percentage of instances and users for all the less popular policies as "Others"

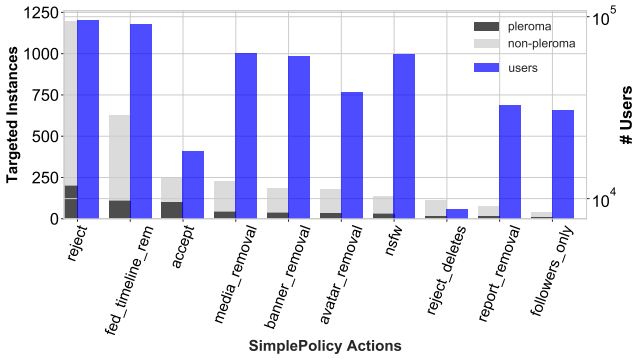

Figure 2: Number of instances targeted by `SimplePolicy` actions (Y-1). Instances are split into Pleroma and non-Pleroma instances. We also plot the number of users on the associated Pleroma instances (Y-2).

for any of the attributes (toxicity, profanity, sexually explicit) is greater or equal to 0.8.

4 EXPLORING POLICIES

We begin by briefly characterising the types of polices used within Pleroma as well as the instances targeted by these policies.

4.1 Characterising Policy Settings

Overview of Policies. We are able to retrieve policy information from 91.9% of Pleroma instances. The remaining 8.1% of Pleroma instances do not expose their policy information. These cover 46 unique policy types: 26 of these policies are included in the Pleroma software package, instance administrators have created the other 20. In general, administrators need to enable policies before they target them towards specific instances. However, we find the `ObjectAgePolicy` and `NoOpPolicy` enabled by default in the software package.

The policies we retrieve affect 97.7% of the total users and 97.8% of all posts. Figure 1 shows the distribution of the top 15 policy types we find enabled by the administrators across instances and the percentage of users signed-up within those instances.

Popular Policies. Most common amongst the popular policies is the `ObjectAgePolicy` (on 66.9% of instances). This policy allows admins to apply an action based on the age of a post regardless of the post's harmful/non harmful nature. The default age threshold is 7 days but administrators are able to configure this as they choose. Possible actions under this policy includes (*i*) delist: removes the post from the public timelines; (*ii*) strip followers: removes followers from the recipient list; and (*iii*) reject: rejects the message entirely. As a default policy, this is enabled on any new installations of Pleroma starting from version 2.1.0.

The `TagPolicy`, applies policies to individual users based on tags but does not completely stop the flow of any material between instances. For example it allows marking posts from individual users as Not Safe For Work (NSFW). This policy is enabled on 33% of instances. Finally, the `SimplePolicy` is enabled on 25.4% of instances. The `SimplePolicy` is the most flexible policy, allowing admins to configure a range of actions on posts or instances that match a certain criteria *e.g.* the `reject` action blocks all connections from a given instance.

The remaining policies are less commonly encountered. For completeness, we briefly discuss a few of them here. The `HellthreadPolicy` is enabled on 6.7% of the instances. This de-lists/rejects a post when the number of user mentions exceeds a set threshold. The `StealEmojiPolicy` whitelists instances to automatically download emojis from; this is enabled on 6.2% of the instances. Other less common policies we encounter include the `HashtagPolicy`, `AntiFollowBotPolicy`, `MediaProxyWarmingPolicy`, `KeywordPolicy` (see Appendix A for a list of Pleroma in-built policies).

SimplyPolicy Breakdown. Due to the diversity of features available within the `SimplePolicy`, its reach as well as its relevance in content moderation, we next inspect the most popular actions associated with the `SimplyPolicy`. Figure 2 presents a breakdown of the various actions used by instances with the `SimplePolicy` against Pleroma instances, as well as instances from other platforms of the fediverse (*e.g.* Gab from Mastodon). The figure also shows the number of users signed-up on these instances. In contrast, Figure 3 shows the number of Pleroma instances that have targeted other instances with the `SimplePolicy` actions and the number of users on these instances.

The figures reveal a rich variety of policy actions. For example, the `media removal` action (which removes any media coming from targeted instances) is applied by 5.4% of the instances, and this impacts 23.3% of users. The most popular and stringent is, however, the `reject` action. Figure 3 shows that the 73% of instances that have the `SimplePolicy` enabled, apply the reject action.

We also notice 86.2% of users and 88.7% of posts are on instances that have been rejected by at least one other instance, and these rejected instances make up 80% of all moderated instances. On a finer granularity, we see the `reject` action making up 62.8% of all moderation events while the sum of all the other (9) `SimplePolicy` actions make-up the remaining 37.2%. As the `reject` action is the

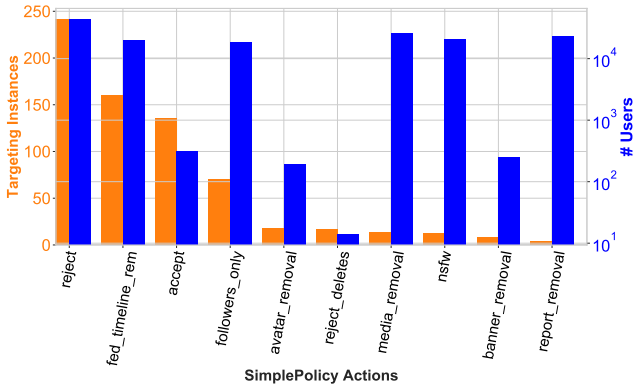

Figure 3: Bar plot showing the number of instances that have targeted other instances with the `SimplePolicy` actions. We also plot the number of users associated with the targeted instances.

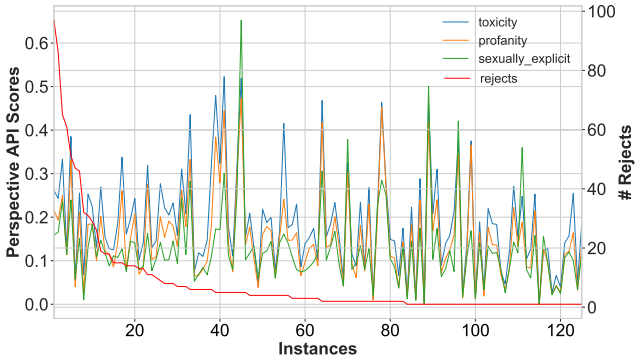

Figure 4: A plot of rejected Pleroma instances with the number of times they are rejected, their average toxicity, profanity and sexually explicit scores across all users on the instances (sorted by the number of rejects).

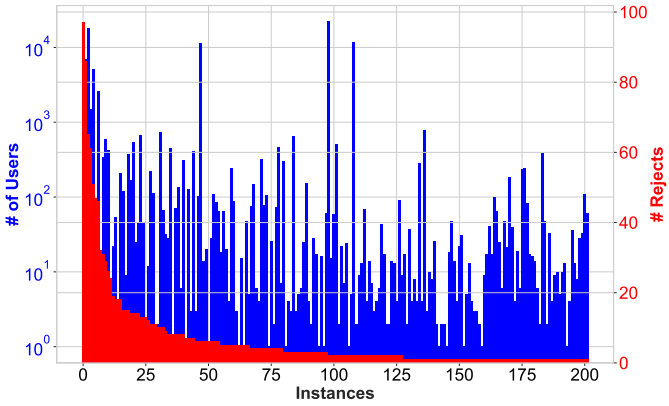

Figure 5: A bar plot showing all rejected Pleroma instances (X-axis) with their number of users and the number of Pleroma instances that have rejected them (sorted by the number of rejects).

most stringent, the most popular and impacting a larger number of users, we focus our analysis on the `reject` action. Hence, we spend the next section exploring the instances that these `reject` actions are targeted against.

4.2 Characterising Rejected Instances

Distribution of Reject Policies. Figure 5 shows the number of `reject` actions targeting each Pleroma rejected instances. These instances represent only 15.5% of Pleroma instances, however, they accumulate 86.2% and 88.7% of the total users and posts, respectively. Instances with more posts tend to receive a larger number of rejects: we find a weak correlation between the number of posts on an instance and the number of rejects (Spearman of 0.38). Overall, we find 1,200 unique instances have been rejected at least once (202 Pleroma and 998 non-Pleroma). The majority of these are targeted by only a small subset of instances though: 86.8% of these are rejected by fewer than 10 instances. However, we do see an "elite" set (5.4%) of controversial Pleroma instances that gain in excess of 20 `reject` actions, led by `Freespeechextremist.com` (a proponent of free speech with 97 rejects). A variety of other types of instances also make up the top rejected list *e.g.* `kiwifarms.cc` (well known for trolling, with 86 rejects), `spinster.xyz` (a feminist instance rejected 65 times) and `neckbeard.xyz` (blocked by another instance linking it to the LGBT community with 61 rejects). This "elite" set of rejected instances accounts for 33.6% and 23.4% of the total Pleroma users and posts, respectively.

For context, Table 1 lists the 5 most rejected Pleroma instances along with their number of users, posts, as well as their average scores in toxicity, profanity and sexually explicit content. Similarly, Figure 4 shows a normalised plot with the Google Perspective API attribute features in Table 1 for all the rejected Pleroma instances in our dataset. The score of each Perspective API attribute is the average score of all posts by users on an instance. Although the instance with the most `reject` actions against it is `gab.com` (a Mastodon instance), Pleroma instances make-up 3 of the top 5. Amongst the top 10 overall, just 40% are from the Pleroma platform. This suggests a larger percentage of illicit material is imported into Pleroma from larger platforms such as `Mastodon` (probably due to the size of their user base).

Do rejected Pleroma instances retaliate? We next assess whether rejected Pleroma instances also tend to use the `reject` action themselves. To answer this, we compute the Spearman's correlation coefficient for rejects applied by *vs.* rejects received for all rejected Pleroma instances (-0.033). This means that the reverse is actually the case. In fact, we notice that the most rejected Pleroma instances barely apply the `reject` action against other instances (pleroma or non-Pleroma). For example, the most rejected Pleroma instance, `freespeechextremist.com`, does not reject a single other instance (Pleroma or non-Pleroma). We conjecture that their openness to any kind of material may contribute to them being rejected. Of the top 10, only 1 Pleroma instance (`spinster.xyz`, a woman-centric instance) has rejected over 2 instances (Pleroma and non-Pleroma) with 45 rejects. A manual check reveals non-tolerance

Instance	#rejects	#users	#user posts	Toxicity Sc	Profanity Sc	Sexually Explicit Sc
freespeech-extremist.com	97	1.8k	1.13M	0.26	0.22	0.16
kiwifarms.cc	86	6.8k	391k	0.24	0.19	0.16
spinster.xyz	65	17.9k	1.34M	NA	NA	NA
neckbeard.xyz	61	15.1k	816k	0.13	0.11	0.11
poa.st	51	5.1k	344k	0.27	0.25	0.18

Table 1: Top 5 Pleroma rejected instances with the number of times they are rejected, users, posts, the averages of their toxicity, profanity and sexually explicit scores.

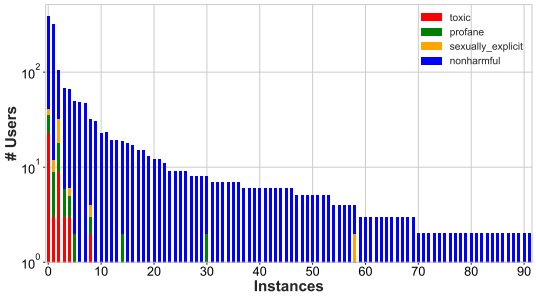

Figure 6: Number of toxic, profane, sexually explicit and non-harmful users on each rejected Pleroma instance.

for pornography, hate, violence or harassment in its "Terms of Service".

Why are instances blocked?. To examine this, we manually annotate the 92 rejected Pleroma instances by going through their post content and also visiting each site. Note, these are the rejected Pleroma instances we have post data for, and we exclude single user instances (see Section 5). We label the rejected Pleroma instances as (*i*) Toxic (hate speech): for content with identity attacks, threats, insults and other hateful material; (*ii*) Sexually explicit (pornography): for adult content; (*iii*) Profane: for material with swear/curse words; and (*iv*) General: for Pleroma instances we are unable to categorise. We are able to annotate 88.4% of the rejected Pleroma instances. For the Pleroma instances we are able to annotate, we find that the sexually explicit, toxic and profane instances make up 90.6%. The remaining 9.4% are labeled as general.

5 IS THERE COLLATERAL DAMAGE?

We conjecture that the activities of individual users may result in an entire instance being rejected. Thus, many "innocent" users may also be rejected by association. We refer to this as **collateral damage**. To shed preliminary light on this question, we explore what fraction of users on rejected Pleroma instances share harmful material. We flag that there may be multiple reasons why an instance is rejected, and emphasise this limitation in our analysis.

In total, we have posts for 61.9% rejected Pleroma instances, with 26.4% of them being single user instances. As we are interested in what percentage of "innocent" users are affected by these policies, we filter out the single user Pleroma instances. We find 1.62k users on these rejected Pleroma instances have publicly accessible content (59.3k posts). Using the Perspective labels (see Section 3), we find

Threshold	0.5	0.6	0.7	0.8	0.9
Non Harmful (%)	86.4	91.8	94.1	95.8	97.3

Table 2: Percentages of harmful and non harmful Pleroma users with varying Google Perspective API thresholds.

that 4.2% of these users on rejected Pleroma instances have a score of >=0.8 in at least one of the three attributes [3]. With our threshold of 0.8, we notice a harmful-to-non-harmful posts ratio of 1:11. We also find that the Pleroma instances with posts averaging a Perspective API score >=0.8 actually make up 7 of the top 10 most rejected Pleroma instances.

Figure 6 shows a stacked bar plot, with the number of users that have been classified as toxic, profane or sexually explicit on each rejected Pleroma instance. We also plot the number of non-harmful users on each Pleroma instance. For the users with an average Perspective API scores >=0.8, we see a distribution of 69.7% toxicity, 57.6% profanity and 43.9% as sexually explicit. Note here that a user can be classified as all 3. Based on this method, 95.8% of the users on rejected Pleroma instances are affected, even though none of their own posts are flagged as harmful. This further strengthens our earlier hypothesis: it is likely that just a few posts from a few users users trigger the rejections.

For completeness, we finally show the percentage of harmful and non-harmful users when we use other Perspective thresholds in Table 2. We find that regardless of the threshold, a high percentage of non harmful users are rejected alongside the small percentage of harmful users.

6 IMPLICATIONS

Collateral Damage. DW instance-based moderation provides useful tools for admins. However, our findings indicate that they are not granular enough. `Reject` actions are applied to entire instances and all associated users are blocked. Despite this, we find that only 4.2% of the users on rejected instances share harmful posts. Although such users may be undesirable for other reasons, this does suggest that a large majority of users may be "collateral damage". This raises questions as to whether rejecting entire instances is the most appropriate approach.

Dissatisfied Users. Generally, users tend to join social platforms where their friends already are, *i.e.* the network effect [2, 16]. This could possibly be one of the major reasons why some decentralised networks struggle to accumulate a strong user base. Being rejected

could result in growing user dissatisfaction, especially if they have friends on another instance which they are banned from following. We argue that such experiences could lead to an exodus of users. Thus, addressing collateral damage is important.

Federation Graph. In a centralised network, a social graph shows the connections between users. In the DW, this can go further to show the connections between instances. The `reject` action tends to have far reaching effects on the social graph. For example, if an instance relies on another instance to reach a segment of the social graph, and due to the actions of a few users it gets rejected, that instance could be cut off from the wider network. This will adversely affect the federation. Exploring the wider impact of this is an interesting line of future work.

7 SOLUTION SPACE

We now briefly explore some solutions to the above challenges. One obvious solution would be for administrators to adopt other less stringent in-built Pleroma policies, *e.g.* tagging posts as NSFW. With this policy, messages from targeted instances are tagged with warnings rather than blocked.

Some of the most rejected Pleroma instances happen to be those with sexually explicit content and these materials are mostly in media form. With the `media removal` facility, multimedia content is removed, leaving only the text. That way, the harmful material loses its meaning while the non harmful users are still able to have their posts delivered across the federation. For DW platforms that carry out moderation in a similar fashion to Pleroma (*e.g.* Mastodon), we concretely propose three steps that could be taken to improve moderation in the DW:

(1) New generic policies could be designed that rely on a reliable and curated list of well-known instances in the fediverse that may need to be blocked. For example, policies called "`NoHate`" or "`NoPorn`" could have instances like `Gab`, `freespeechextremist.com` and `social.myfreecams.com`, and `baraag.net` listed as part of a community effort. Thus, an administrator could simply select the relevant lists. We expect that these listings are periodically updated by professionals who ensure that the instances have limited collateral damage.
(2) New user-driven policies could be designed that enable administrators to moderate on a per-user basis. Notable among these policies is the `TagPolicy`, which is able to apply a policy to a user based on a tag applied. Hence, streamline moderation interfaces could be devised to make the process of tagging individual users straightforward (potentially assisted by automated classifiers).
(3) New policies could enable administrators to automatically implement policies/actions for users who repeatedly violate the "Terms of Service". For example, policies/actions could be automatically applied (*e.g.* NSFW, media removal) to a user when they have been reported n times, or when the user post goes above a certain threshold (*e.g.* in Google Perspective API). Again, such actions could be assisted by automatic classification of behaviour.

8 RELATED WORK

There has been a range of work on content moderation in social media. Halevy et al. [6] looked into striking a balance between free speech and safety, considering diverse cultural and political climates. Fortuna et al. [12] used 6 publicly available datasets and compared the labeling of each dataset for attributes such as sexism, toxicity and racism. They find that definitions, datasets and conflicting annotations can all affect the performance of classifiers. Ribeiro et al [26] looked at differences between hateful Twitter users and normal users with respect to their activities, vocabulary and network centrality. Other studies have profiled social media users based on their dissemination of hate material [14, 15]. Our work differs in that we have focused on the implementation of policies by administrators on Pleroma, rather than the behaviour of hateful users.

There have also been a set of studies looking specifically at DW services. Raman et al. [25] measured the challenges in deploying DW applications, particularly related to network issues [10]. Zignani et al. [31] studied the growth of Mastodon while comparing its structure with Twitter. Similarly, La Cava et al. [11] explored the evolution of Mastodon at an instance level, as well as the connectivity between instances. Zignani et al. [19] further investigated the interrelationship between the Mastodon system design and the social network. Another closely related work [18] looked at how Mastodon users tag their own posts as NSFW. Doan et al. [30] investigated the performance of a decentralized video streaming platform (DTube) by developing an app that streams from both centralized and decentralized services. We differ from these works in that we focus on content moderation activities.

9 CONCLUSION

In this paper we have presented the first study of Pleroma and its associated policies. We find that policies are widely used, impacting 97.7% of users and 97.8% of posts. Using the Perspective API, we have found that 95.8% of the users on rejected Pleroma instances do not share posts classified as harmful. This leaves just 4.2% of harmful users that are likely responsible for the rejects. This implies significant "collateral damage". This has led us to sketch a set of strawman policies that may reduce this damage. Our proposed solutions would generally be applicable to platforms that carry out moderation at a per instance granularity (*e.g.* DW) rather than at a per user granularity (*e.g.* Twitter). In our future work, we plan to implement and evaluate these policies, as well as continue to explore how other DW platforms perform moderation.

There are a number of lines of future work. We wish to further explore the reasons why administrators apply particular policies. This, for instance, could be achieved via user surveys. Using this knowledge, we intend to develop novel policies that can assist administrators. We are particularly interested in designing more techniques that can automatically identify users or instances to apply certain policies too.

ACKNOWLEDGEMENTS

This work is supported EPSRC EP/S033564/1 and EU H2020 grant agreements No 871793 (Accordion), No 871370 (Pimcity), and No 101016509 (Charity)

REFERENCES

[1] ActivityPub. 2018. https://www.w3.org/TR/activitypub/.
[2] Fabrício Benevenuto Tiago Rodrigues Meeyoung Cha Virgílio Almeida. 2009. Characterizing User Behavior in Online Social Networks. In *IMC*.
[3] Corel. 2021. Corel: Toxic Comments|Talk Documentation. https://docs.coralproject.net/talk/toxic-comments/
[4] Megan Farokhmanesh. 2017. A beginner's guide to Mastodon, the hot new open-source Twitter clone. https://www.theverge.com/2017/4/7/15183128/mastodon-open-source-twitter-clone-how-to-use.
[5] Barbara Guidi, Marco Conti, Andrea Passarella, and Laura Ricci. 2018. Managing social contents in Decentralized Online Social Networks: A survey. *Online Social Networks and Media* (2018).
[6] Ves Halevy Alon Canton-Ferrer Cristian Ma Hao Ozertem Umut Pantel Patrick Saeidi Marzieh Silvestri Fabrizio Stoyanov. 2020. Preserving integrity in online social networks. In *arXiv*.
[7] Davidson Thomas Warmsley Dana Macy Michael Weber Ingmar. 2017. Automated hate speech detection and the problem of offensive language. In *ICWSM*.
[8] jigsaw. 2019. https://github.com/conversationai/.
[9] Lucas Dixon John Pavlopoulos, Nithum Thain and Ion Androutsopoulos. 2019. Offensive language identification and categorization with perspective and bert. 571–576.
[10] Sebastian Kaune, Konstantin Pussep, Christof Leng, Aleksandra Kovacevic, Gareth Tyson, and Ralf Steinmetz. 2009. Modelling the internet delay space based on geographical locations. In *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*. IEEE, 301–310.
[11] Greco Sergio La Cava Lucio and Tagarelli Andrea. 2021. Understanding the growth of the Fediverse through the lens of Mastodon. In *Applied Network Science*, Vol. 6. 639–645.
[12] Fortuna Paula Soler-Company Juan Wanner Leo. 2020. Toxic, hateful, offensive or abusive? What are we really classifying? An empirical analysis of hate speech datasets. In *LREC*.
[13] Dixon Lucas. Li John. Sorensen Jeffrey. Thain Nithum. Vasserman Lucy. 2018. Measuring and Mitigating Unintended Bias in Text Classification. In *AIES 2018* . 67–73.
[14] Binny Mathew, Anurag Illendula, Punyajoy Saha, Soumya Sarkar, Pawan Goyal, and Animesh Mukherjee. 2020. Hate begets hate: A temporal study of hate speech. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (2020), 1–24.
[15] Goyal Pawan Mathew Binny, Dutt Ritam and Mukherjee Animesh. 2018. Spread of hate speech in online social media. In *arXiv*. 173–182.
[16] Giuliano Mega, A Montresor, and GP Picco and. 2011. Efficient dissemination in decentralized social networks. IEEE International Conference on Peer-to-Peer Computing, 338–347.
[17] Rajadesingan. A Resnick P. and Budak C. 2020. Quick, Community-Specific Learning: How Distinctive Toxicity Norms Are Maintained in Political Subreddits. In *AAAI Conference*. 557–568.
[18] Zignani Matteo Quadri Christian Galdeman Alessia Gaito Sabrina Rossi Gian Paolo. 2019. Mastodon content warnings: Inappropriate contents in a microblogging platform. In *ICWSM 2019*. 639–645.
[19] Zignani Matteo. Quadri Christian. Gaito Sabrina. Cherifi Hocine. Rossi Gian Paolo. 2019. The Footprints of a 'Mastodon': How a Decentralized Architecture Influences Online Social Relationships. In *INFOCOM 2019*. 472–477.
[20] Perspective. 2021. https://www.perspectiveapi.com/how-it-works/.
[21] Perspective. 2021. https://developers.perspectiveapi.com/s/about-the-api-model-cards.
[22] Perspective. 2021. https://www.perspectiveapi.com/how-it-works/.
[23] Perspective. 2021. https://developers.perspectiveapi.com/s/about-the-api-model-cards.
[24] Budak Ceren Rajadesingan Ashwin, Resnick Paul. 2020. Quick, community-specific learning: How distinctive toxicity norms are maintained in political subreddits. In *ICWSM*.
[25] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. 2019. Challenges in the Decentralised Web: The Mastodon Case. In *ACM IMC*. 217–229.
[26] Manoel Ribeiro, Pedro Calais, Yuri Santos, Virgílio Almeida, and Wagner Meira Jr. 2018. Characterizing and detecting hateful users on twitter. In *ICWSM*.
[27] Ribeiro Manoel Horta. Jhaver Shagun. Zannettou Savvas. Blackburn Jeremy. De Cristofaro Emiliano. Stringhini Gianluca. West Robert. 2020. Does Platform Migration Compromise Content Moderation? Evidence from $r/The_Donaldandr/Incels$.
[28] Beverly Robert Rye Erik, Blackburn Jeremy. 2020. Reading In-Between the Lines: An Analysis of Dissenter. In *arXiv*. ACM SIGCOMM.
[29] The Copia Institute. 2021. https://www.tsf.foundation/blog/decentralized-social-media-platform-mastodon-deals-with-an-influx-of-gab.
[30] Doan Trinh Viet. Pham Tat Dat. Oberprieler Markus. Bajpai Vaibhav. 2020. Measuring Decentralized Video Streaming: A Case Study of DTube. In *IFIP Networking 2020*. 118–126.
[31] Matteo Zignani, Sabrina Galto, and Gian Paolo Rossi. 2018. Follow the "Mastodon": Structure and evolution of a decentralized online social media. In *ICWSM*. 541–550.

A APPENDIX

We summarise the basic functionalities of Pleorma in-built policies in Table 3.

For completeness, we show in Figure 7 the entire policy spectrum, the percentage of Pleroma instances that enable these policies, as well as the number of users on these instance.

policy	#description	#instances	#users
ObjectAge	Rejects or delists posts based on their age when received	869	57,854
TagPolicy	Applies policies to individual users based on tags	429	38,067
SimplePolicy	Restrict the visibility of activities from certains instances with a suite of actions	330	46,691
NoOpPolicy	Doesn't modify activities (default)	176	6,443
HellthreadPolicy	De-list or reject messages when the set number of mentioned users threshold is exceeded	87	14,401
StealEmojiPolicy	List of hosts to steal emojis from	81	7,003
HashtagPolicy	List of hashtags to mark activities as sensitive (default: nsfw)	62	10,933
AntiFollowbotPolicy	Stop the automatic following of newly discovered users	51	6,918
MediaProxyWarmingPolicy	Crawls attachments using their MediaProxy URLs so that the MediaProxy cache is primed	46	9,851
KeywordPolicy	A list of patterns which result in message being reject/unlisted/replaced	42	22,428
AntiLinkSpamPolicy	Rejects posts from likely spambots by rejecting posts from new users that contain links	32	7,347
ForceBotUnlistedPolicy	Makes all bot posts to disappear from public timelines	23	6,746
EnsureRePrepended	Rewrites posts to ensure that replies to posts with subjects do not have an identical subject and instead begin with re:	18	247
ActivityExpirationPolicy	Sets a default expiration on all posts made by users of the local instance	11	1,420
SubchainPolicy	Selectively runs other MRF policies when messages match	8	81
MentionPolicy	Drops posts mentioning configurable users	6	1,149
VocabularyPolicy	Restricts activities to a configured set of vocabulary	5	121
AntiHellthreadPolicy	Stops the use of the HellthreadPolicy	4	2,106
RejectNonPublic	Whether to allow followers-only/direct posts	3	1,101
FollowBotPolicy	Automatically follows newly discovered users from the specified bot account	2	281
DropPolicy	Drops all activities	1	1,098

Table 3: Description of policies provided by Pleroma and the number of instances that enable them, as well as the number of users on the instances

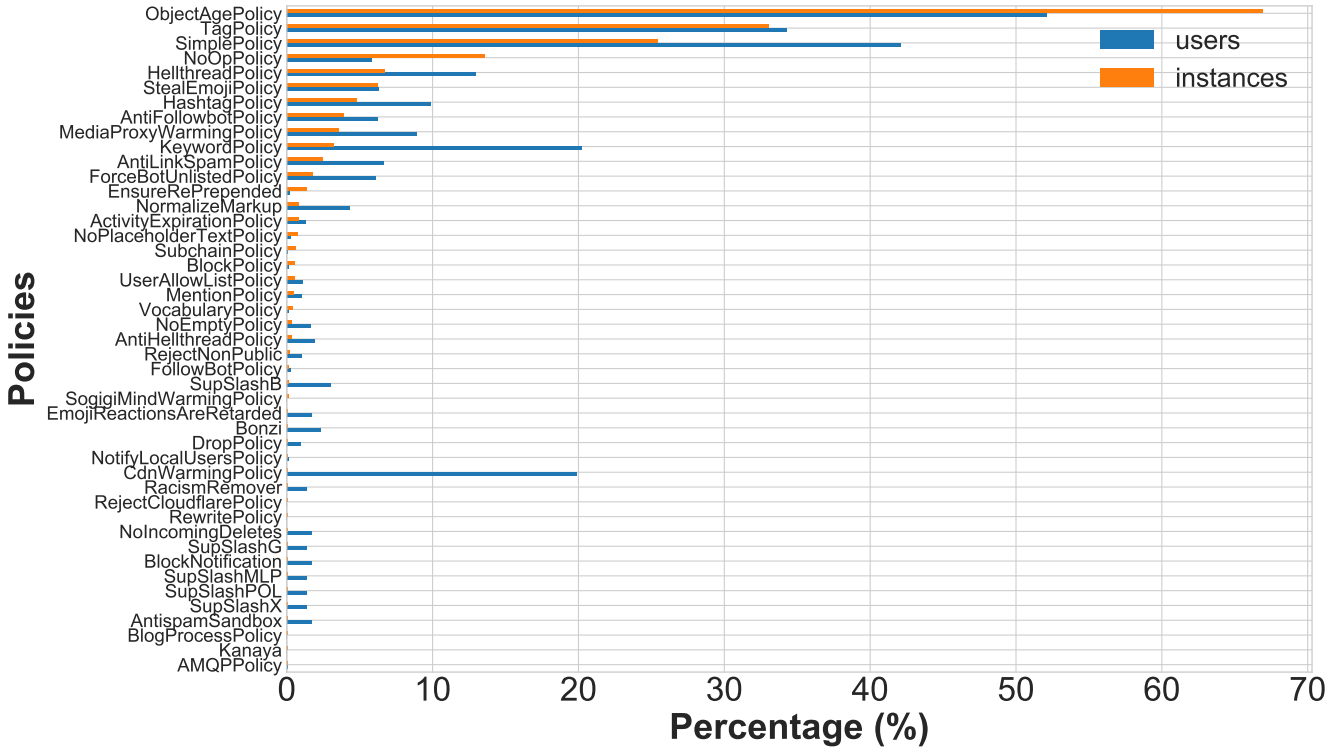

Figure 7: Entire policy spectrum showing policy types and the percentage of instances that use each policy (sorted by the percentage of instances). We also include the percentage of the global user population on the instances that use each policy.