# Peer-to-Peer Permissionless Consensus via Authoring Reputation

**Anonymous**

[1]Anonymous

***Abstract.*** *Public Internet forums suffer from excess and abuse, such as SPAM and fake news. Centralized platforms employ filtering and anti-abuse policies, but imply full trust from users. We propose a permissionless Sybil-resistant peer-to-peer protocol for content sharing. Our main contribution is a reputation system that moderates content and, at the same time, delivers network consensus. We can trace a parallel with Bitcoin as follows: consolidated posts create reputation (vs proof-of-work), likes and dislikes transfer reputation (vs transactions), and aggregate reputation determines consensus (vs longest chain). The reputation mechanism depends exclusively on the human authoring ability (*proof-of-authoring*), which is slow and scarce, thus suitable to establish consensus.*

***Resumo.*** *Fórums públicos de Internet sofrem de excesso e abuso, tais como SPAM e notícias falsas. Plataformas centralizadas adotam políticas de filtragem e anti abuso, mas exigem confiança dos seus usuários. Nós propomos um protocolo peer-to-peer não permissionado para compartilhamento de conteúdo que é resistente a ataques Sybil. Nossa principal contribuição é um sistema de reputação que serve para moderar o conteúdo e ao mesmo tempo garantir consenso na rede. Nós traçamos um paralelo com o Bitcoin da seguinte forma: postagens consolidadas criam reputação (vs* proof-of-work*),* likes *e* dislikes *transferem reputação (vs transações), e a reputação agregada determina o consenso (vs cadeia mais longa). O mecanismo de reputação depende exclusivamente da habilidade de autoria humana (*proof-of-authoring*), que é lenta e escassa, e portanto adqueada para estabelecer consenso.*

## 1. Introduction

Content publishing in Internet forums and social media is increasingly more centralized in a few companies (e.g. Facebook and Twitter) [27, 11, 10]. On the one hand, these companies offer free storage, friendly user interfaces, and robust access. On the other hand, they concentrate power by collecting users' data and "algorithmizing" consumption. Peer-to-peer alternatives [23] eliminate intermediaries, but strive to achieve consistency while dealing with malicious users.

In an ideal Internet forum, all messages or posts (i) reach all users, (ii) are delivered in a consistent order, and (iii) are respectful and on topic. In a centralized system, items (i) and (ii) are trivially achieved assuming availability and delivery order in the service, while for item (iii), users have to trust the service to moderate content. In a decentralized setting, however, none of these demands are easily accomplished. A common approach in gossiping protocols is to proactively replicate and disseminate posts among peers until they reach all users [23, 8]. However, this approach does not guarantee consensus since posts can be received in conflicting orders [21, 14]. Consensus is key to

eradicate Sybil attacks [5], which are the major threat to decentralized applications in general: without consensus, it is not possible, *at the protocol level*, to distinguish between correct and malicious users in order to satisfy item (iii).

Bitcoin [13] is the first permissionless protocol to resist Sybils through consensus. Its key insight is to rely on a scarce resource—the *proof-of-work*—to establish consensus. The protocol is Sybil resistant because it is expensive to write to its unique timeline (either via proof-of-work or transaction fees). However, Bitcoin and cryptocurrencies in general are not suitable for content sharing because (i) they enforce a unique timeline to preserve value and immunity to attacks; (ii) they lean towards concentration of power due to scaling effects; and (iii) they impose an external economic cost to use the protocol. These issues threaten our original decentralization goals. In particular, a unique timeline implies that all Internet content should be subject to the same consensus rules, which neglects all subjectivity that is inherent to social content. Another limitation of cryptocurrencies is that it is not possible to revoke content in the middle of a blockchain, which is inadmissible considering illegal content (e.g., hate speech).

In this work, we adapt the idea of scarce resources to reach consensus, but in the context of social content sharing. Our first contribution is to recognize the actual published contents as the protocol scarce resources, since they require human work. Work is manifested as new posts, which if approved by others, reward authors with reputation tokens, which are used to evaluate other posts with likes and dislikes. With such *proof-of-authoring* mechanism, token generation is expensive, while verification is cheap and made by multiple users. Due to decentralization, posts in a timeline form a causal graph with only partial order, which we promote to a total order based on the reputation of authors. The consensus order is fundamental to detect conflicting operations, such as likes with insufficient reputation (akin to Bitcoin's double spending). Our second contribution is to allow that users create diversified forums of interest (instead of a singleton blockchain), each counting as an independent timeline with its own subjective consensus etiquette. Our third contribution is to support content removal without compromising the integrity of the decentralized blockchain. Users have the power to revoke posts with dislikes, and peers are forced to remove payloads, only forwarding associated metadata. We integrated the proposed consensus algorithm into Freechains [19], a practical peer-to-peer content dissemination protocol that provides strong eventual consistency [20, 6]. To show the practicability of the consensus mechanism, we simulated months of activity of a chat channel and years of a newsgroup forum, both extracted from publicly available Internet archives.

The rest of the paper is organized as follows: In Section 2, we describe the design of the reputation and consensus mechanism for public forums. In Section 3, we describe the concrete reputation rules we implemented for Freechains. In Section 4, we evaluate the performance of the protocol in real-world public forums. In Section 5, we compare our system with publish-subscribe protocols, federated applications, and fully peer-to-peer systems. In Section 6, we conclude this work.

| Operation | Effect | Goal |
|---|---|---|
| Emission | Old posts award *reps* to authors. | Encourage content authoring. |
| Expense | New posts deduct *reps* from authors temporarily. | Discourage excess of content. |
| Transfer | Likes & dislikes transfer *reps* between authors. | Highlight content of quality. Combat abusive content. |

**Table 1. General reputation operations in public forums.**

## 2. The Reputation and Consensus Mechanism

In the absence of moderation, permissionless peer-to-peer forums are impractical, mostly because of Sybils abusing the system. For instance, it should take a few seconds to generate thousands of fake identities and SPAM millions of messages into the system. For this reason, we propose a reputation system that works together with a consensus algorithm to resist Sybil attacks.

We propose a reputation system in which users can spend tokens named *reps* to post and rate content in the forums: a `post` initially penalizes authors until it consolidates and counts positively; a `like` is a positive feedback that helps subscribers to distinguish content amid excess; a `dislike` is a negative feedback that revokes content when crossing a threshold. Table 1 summarizes the reputation operations and their goals. To prevent Sybils, users with no *reps* cannot perform these operations, requiring a welcoming like from any other user already in the system. The fact that likes are zero-sum operations, which only transfer reputation, eliminates the incentives from malicious users to invite Sybils into the system. The only way to generate new *reps* is to post content that other users approve, which demands non-trivial work resistant to automation.[1]

Bitcoin employs proof-of-work to mitigate Sybil attacks. However, CPU or other extrinsic resources are not evenly distributed among humans, specially in communications using battery-powered devices. Instead, considering the context of public forums, we can take advantage of the human authoring ability as an intrinsic resource. Creating new content is hard and takes time, but is comparatively easy to verify and rate. Therefore, in order to impose scarcity, we determine that only content authoring generates *reps*, while likes and dislikes just transfer *reps* between users. Nevertheless, scarce operations are not yet sufficient because they demand consensus to establish an order in time across the network to prevent inconsistent operations. As an example, consider a malicious author with a single unit of *reps* posting new messages using multiple peers at the same time. According to the *Expense* rule of Table 1, only one of these messages should be accepted. However, without consensus, it is not possible to globally determine which message to accept, since each peer would supposedly accept the first message it sees. On the one hand, in order to validate operations consistently, we need the same message ordering across all peers in the network. On the other hand, we typically can only represent peer-to-peer public forums as DAGs with causal relationships between messages, which provides at most partial order.

Therefore, our key idea to stablish consensus in forum DAGs is to favor forks

[1]Note that to some extent work from AI may be indistinguishable from humans, but its frequency and relevance are still subject to human evaluation, which is subjective and evolves over time.

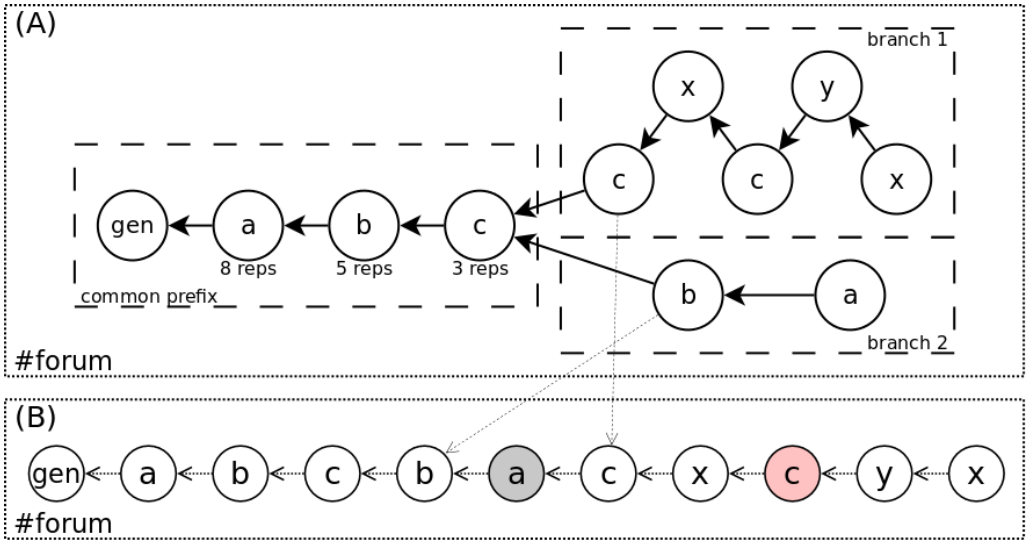


**Figure 1. (A) A public forum DAG with a common prefix and two branches. (B) Total order between blocks of the DAG after consensus.**

with posts from users that constitute the majority of the reputation in the network. These forks have more associated work from active users and are analogous to longest chains in Bitcoin. In technical terms, we can adapt a topological sorting algorithm to favor reputation when deciding between branches in a forum DAG.

Figure 1.A illustrates the consensus criteria. A public forum DAG has a common prefix with signed posts from users $a$, $b$, and $c$. Let's assume that within the prefix, users $a$ and $b$ have contributed with better content and have more reputation combined than $c$ has alone (i.e., $8 + 5 > 3$). After the prefix, the forum forks in two branches: in `branch-1`, only user $c$ remains active and we see that new users $x$ and $y$ (with no previous reputation in the common prefix) generate a lot of new content; in `branch-2`, only users $a$ and $b$ participate but with less activity. Nevertheless, `branch-2` would be ordered first in this case because, before the forking point, $a$ and $b$ have more reputation than $c$, $x$, and $y$ combined. User $c$ here might represent a malicious user trying to cultivate fake identities $x$ and $y$ in separate of the network during weeks to accumulate *reps*.

Figure 1.B indicates the resulting consensus order between blocks in the forum. All operations in `branch-2` appear before any operation in `branch-1`. At any point in the consensus timeline, if an operation fails, all remaining blocks in the offending branch are removed from the primary DAG. As an example, suppose that the last post by $a$ (in gray) is a dislike to user $c$. Then, it's possible that the last post by $c$ (in red), now (suppose) with *0 reps*, is rejected together with all posts by $y$ and $x$ in sequence. The removal rule for the remaining blocks is a direct consequence of tamper proof Merkle DAGs [4, 13] used in permissionless protocols, which cannot support graph modifications.

Note that the consensus order exists only for accountability purposes, and is a view of the primary DAG structure. Note also that users in the branch with more reputation can react to attacks even after the fact. For instance, users $a$ and $b$ can pretend that they did not yet see `branch-1` and post extra dislikes to user $c$ from `branch-2` so that a further merge removes all blocks of `branch-1` from the DAG. There are some other relevant considerations about forks and merges: Peers that first received branches with

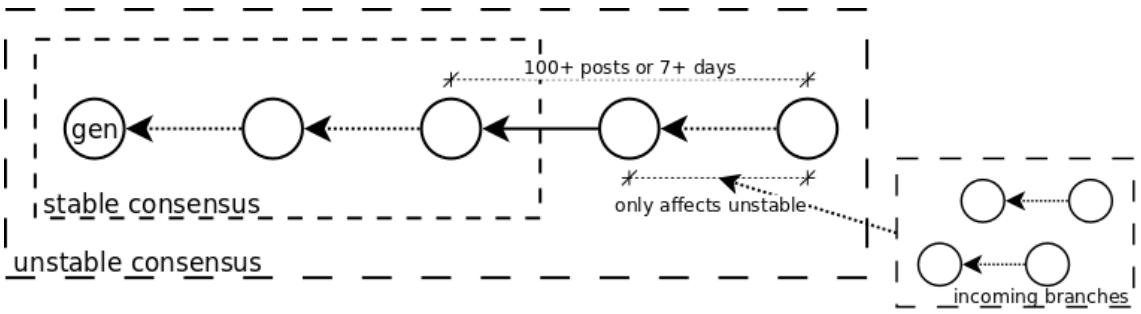


**Figure 2. Stable consensus freezes the order of blocks once they cross the threshold. The unstable order may still be affected by incoming branches.**

less reputation will need to reorder all blocks starting at the forking point. This might involve removing content in the end-user software. This behavior is similar to Bitcoin's blockchain reorganization, when a peer detects a new longest chain. Likewise, peers that first saw branches with more reputation just need to put the other branch in sequence with no reordering at all. This behavior is expected to happen in the majority of the network. Note that unlike Bitcoin, forks are not only permitted but encouraged due to the local-first software principle [9], in which networked applications can work locally while offline. However, the longer a peer remains disconnected, the more conflicting operations it may see, and the higher are the chances of rejection when rejoining.

As a counterpoint to the consensus order in Figure 1.B, maybe users $a$ and $b$ have abandoned the forum for months, and thus `branch-1` is actually legit. In this case, users $a$ and $b$ might be the ones trying to take over the forum. Yet another possibility is that both branches are legit but became disconnected for a long period of time. In any case, it is unacceptable that a very old remote branch affects a long active local forum. For this reason, the consensus algorithm includes an extra constraint that prevents long-lasting local branches to merge, creating *hard forks* in the network. A hard fork occurs when a local branch crosses a predetermined and irreversible threshold of *7 days* or *100 posts* of activity. In this case, regardless of the remote branch reputation, the local branch takes priority and is ordered first. This situation is analogous to a hard fork in Bitcoin and the branches will never synchronize again. More than simply numeric disputes, hard forks represent social conflicts in which reconciling branches is no longer possible. Figure 2 illustrates hard forks by distinguishing *stable consensus*, which cannot be reordered, from *unstable consensus*, which may still be affected by incoming branches. The activity threshold counts backwards, from the latest local block in the unstable consensus.

In summary, the rules to merge a branch from a remote machine `j` into a branch from a local machine `i` are as follows:

- `i` is first if it crosses the activity threshold of *7 days* or *100 posts*, regardless of `j`.
- `i` or `j` is first, whichever has more reputation in the common prefix.
- otherwise, branches are ordered by an arbitrary criteria, such as lexicographical order of the block hashes immediately after the common prefix.

Finally, we consider that revoking posts is fundamental in the context of social content publishing, and thus, an important contribution of this work. As described in Figure 3, a forum post has three possible states: `BLOCKED`, `ACCEPTED`, or `REVOKED`. If the author has reputation, a new post is immediately `ACCEPTED` in the forum. Otherwise, it is

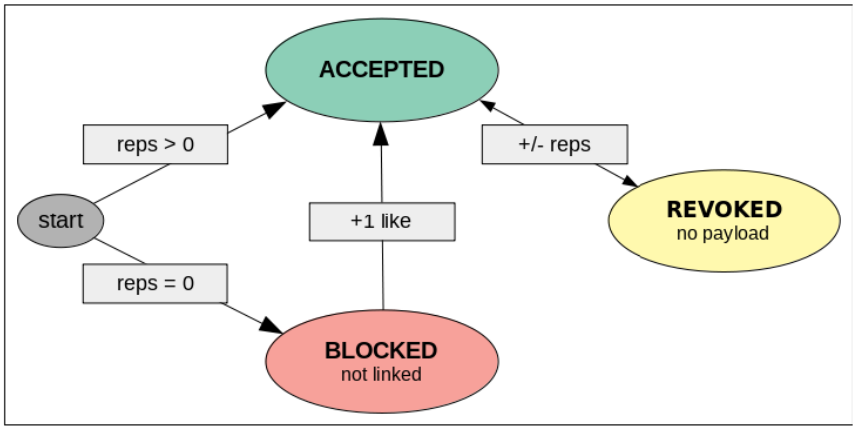


**Figure 3. State machine of posts: `BLOCKED` posts are not linked in the DAG. `ACCEPTED` posts are linked and retransmitted. The payload of `REVOKED` posts are not retransmitted.**

`BLOCKED` and requires a like from another user. Blocked posts are not considered part of the forum DAG in the sense that new posts do not link back to it. In addition, peers are not required to hold blocked posts and neither retransmit them to other peers. However, if blocked posts are not disseminated, new users will never have the chance to be welcomed with a like. A reasonable policy is to hold blocked posts in a temporary bag and retransmit them for some visibility in the network. Once accepted, a post becomes part of the forum and can never be removed again, since Merkle DAGs are immutable by design. However, if the number of dislikes exceeds an arbitrary threshold (e.g., more dislike than likes), the block becomes `REVOKED` and its payload is not retransmitted to other peers. Note that a block hash does not depend on its associated payload, but only on the payload hash. Hence, it is safe to remove the payload as long as one can prove its revoked state. Later, if the post receives new likes, it means that the payload is still known somewhere and peers can request it when synchronizing again.

## 3. Public Forums in Freechains

Freechains [19] is an unstructured peer-to-peer topic-based publish-subscribe protocol, in which each *chain* is a replicated Merkle-DAG representing the causal relationships between the messages. The protocol operation is typical of publish-subscribe systems: an author publishes a post to a chain, and subscribed users eventually receive the message.

We integrated the proposed reputation and consensus mechanism with Freechains to support content moderation and mitigate abuse. Table 2 details the rules we conceived, which are discussed as follows through a concrete example. Authors have to sign posts in order to be accounted by the reputation system and operate in the chains. We start by creating an identity whose public key is assigned as the pioneer in a chain named *forum*:

```
> freechains keys pubpvt 'pioneer-password'
4B56AD.. DA3B5F..  <-- public and private keys
> freechains 'forum' join '4B56AD..'
10AE3E..           <-- hash representing the chain
> freechains 'forum' post --sign='DA3B5F..' \
   'The purpose of this chain is...'
1_CC2184..         <-- hash representing the post
```

| Operation | Rule num | Rule name | Description | Observations |
|---|---|---|---|---|
| Emission | 1.a | pioneers | Chain join counts *+30 reps* equally distributed between the pioneers. | `[1.b]` A post takes 24 hours to consolidate. New posts during this period will not be rewarded later. Only after this period, the next post starts to count 24 hours. |
| | 1.b | old post | Consolidated post counts *+1 rep* to author. | |
| Expense | 2 | new post | New post counts *-1 rep* to author temporarily. | The discount period varies from 0 to 12 hours and is proportional to the sum of authors' reps in subsequent |
| Transfer | 3.a | like | Like counts *-1 rep* to origin and *+1 rep* to target | The targets are the referred post and its corresponding author. If a post has at least 3 dislikes and more dislikes than likes, then its contents are hidden. `[3.b]` Users can revoke their own posts with a single |
| | 3.b | dislike | Dislike counts *-1 rep* to origin and *-1 rep* to targ | |
| Constraints | 4.a | min | Author requires at least *+1 rep* to post. | |
| | 4.b | max | Author is limited to at most +30 reps. | |
| | 4.c | size | Post size is limited to at most 128Kb. | |

**Table 2. Reputation rules for public forum chains in Freechains. The chosen constants ($30\ reps$, $24h$, etc) are arbitrary and target typical Internet forums.**

The `join` command in rule `1.a` bootstraps a public chain, assigning *30 reps* equally distributed between an arbitrary number of pioneers indicated through their public keys. The pioneers shape the initial culture of the chain with the first posts and likes, while they gradually transfer *reps* to other authors, which also transfer to other authors, expanding the community. The `post` command in sequence is signed by the single pioneer (in this example) and indicates the purpose of the chain for future users.

The most basic concern in public forums is to resist Sybils abusing the chains. Fully peer-to-peer systems cannot rely on logins or CAPTCHAs due to the lack of a central authority. Viable alternatives include (i) building social trust graphs, in which users already in the community vouch for new users, or (ii) imposing explicit costs for new posts, such as proof of work. We propose a mix between trust graphs and economic costs. Rule `4.a` imposes that authors require at least *1 rep* to post, effectively blocking Sybil actions. To vouch for new users, rule `3.a` allows an existing user to like a newbie's post to unblock it, but at the cost of *1 rep*. This cost prevents that malicious members unblock new users indiscriminately, which would be a breach for Sybils. For the same reason, rule `2` imposes a temporary cost of *1 rep* for each new post. Note that the pioneers rule `1.a` solves the chicken-and-egg problem imposed by rule `4.a`: if new authors start with no *reps*, but require *reps* to operate, it is necessary that some authors have initial *reps* to boot the chains.

In the next sequence of commands, a new remote user joins the same public chain and posts a message, which is welcomed with a like signed by the pioneer:

```
> freechains keys pubpvt 'newbie-password'
503AB5.. 41DDF1..  <-- public and private keys
> freechains 'forum' join '4B56AD..'
10AE3E..           <-- same pioneer as before
> freechains 'forum' post 'Im a newbie...' \
   --sign='41DDF1..'
2_C3A40F..         <-- blocked post
> freechains 'forum' like '2_C3A40F..' \
   --sign='DA3B5F..'
3_59F3E1..         <-- hash representing the like
```

Note that chains with the same name but different pioneers would be incompatible because the hash of genesis blocks depend on the pioneers' public keys.

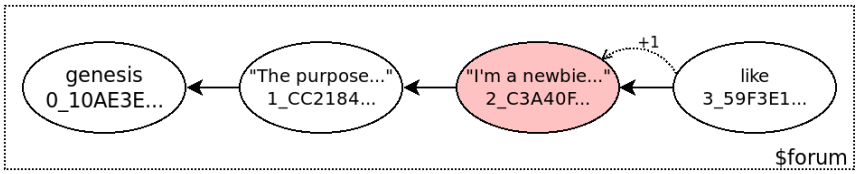


**Figure 4. The `like` approves the newbie message into the `#forum` DAG.**

Figure 4 illustrates the chain DAG up to the like operation. The pioneer starts with *30 reps* (rule `1.a`) and posts the initial message. New posts penalize authors with *-1 reps* during at most 12 hours (rule `2`), which depends on the activity succeeding (and including) the new post. The more activity from reputed authors, the less time the discount persists. In the example, since the post is from the pioneer controlling all *reps* in the chain, the penalty falls immediately and she remains with *30 reps*. This mechanism limits the excess of posts in chains dynamically. For instance, in slow technical mailing lists, it is more expensive to post messages in sequence. However, in chats with a lot of active users, the penalty can decrease to zero quickly.

Back to Figure 4, a new user with *0 reps* tries to post a message (hash `C3A40F..`) and is blocked (rule `4.a`), as the red background highlights. But the pioneer liked the blocked message, decreasing herself to *29 reps* and increasing new user to *1 rep* (rule `3.a`). Note that the newbie post is not penalized (rule `2`) because it is followed by the pioneer like, which still controls all *reps* in the chain.

With no additional rules to generate *reps*, the initial *30 reps* would constitute the whole "chain economy" forever. For this reason, rule `1.b` rewards authors of new posts with *1 rep*, but only after 24 hours. This rule stimulates content creation and grows the economy of chains. The 24-hour period gives sufficient time for other users to judge the post before rewarding the author. It also regulates the growth speed of the chain. In Figure 4, after 1 day, the pioneer would now accumulate *30 reps* and the new user *2 reps*, growing the economy in *2 reps* as result of the two consolidated posts. Note that rule `1.b` rewards at most one post of each author at a time. Hence, new posts during the 24-hour period will not reward each of them with extra *reps*. Note also that rule `4.b` limits authors to at most *30 reps*, which provides incentives to spend likes and thus decentralize the network, while rule `4.c` limits the size of posts to at most *128kB* to prevent DDoS attacks using gigantic blocked posts.

Likes and dislikes (rules `3.a` and `3.b`) serve three purposes in the chains: (i) welcoming new users, (ii) measuring the quality of posts, and (iii) revoking abusive posts (SPAM, fake news, illegal content, etc). The quality of posts is subjective and is up to users to judge them with likes, dislikes, or simply abstaining. This way, access to chains is permissionless in the sense that the actual peers and identities behind posts are not directly assessed by the protocol, but instead by the other users in the system. The reputation of a given post is the difference between its likes and dislikes, which can be used in end-user software for filtering and highlighting purposes. On the one hand, since *reps* are finite, users need to ponder to avoid indiscriminate expenditure. On the other hand, since *reps* are limited to at most *30 reps* per author (rule `4.b`), users also have incentives to rate content. Hence, these upper and lower limits work together towards the quality of the chains. Note that a dislike shrinks the chain economy since it removes *reps* from both the

origin and target. Finally, the actual contents of a post may be revoked if it has at least 3 dislikes, and more dislikes than likes (rule `3`). However, considering that *reps* are scarce, dislikes are encouraged to combat abusive behavior, but not to eliminate divergences of opinion.

## 4. Experiments with Public Forums

In this section, we perform experiments to evaluate the performance and practicability of the protocol. As detailed further, we measure the following evaluation parameters: (a) metadata overhead, (b) consensus runtime, (c) graph forks, and (d) blocked messages.

We simulate the behavior of two publicly available forums as if they were using Freechains: a chat channel from the Wikimedia Foundation[2], and the *comp.compilers* newsgroup[3]. Chats and newsgroups represent typical public forums with faster interactions with shorter payloads (chats), and slower interactions with larger payloads (newsgroups). We only simulate the first *10.000* messages of the forums, which represent 3 months of activity in the chat and 9 years in the newsgroup.

The simulation spawns $N$ peers, each joining the same chain with the same arguments. For each message in the original forum, we (i) set the timestamp of all peers to match the date, (ii) create a pair of keys if the author is new, (iii) post the message from a random peer in $N$, (iv) like the post if the author has no reputation, and (v) synchronize the chain with $M$ random peers. Since all messages are part of the original archive, we always perform step (iv) to unblock messages from newbies, which we use to measure the evaluation parameter (d). Step (v) will inevitably create forks in the chain for any `M<N`, which we measure in evaluation parameter (c).

For the newsgroup, we use `N=15` and `M=5`, which represents a larger number of peers with few interconnections to stress the local-first nature of the protocol. For the chat, we use `N=5` and `M=3`, which represents a smaller number of peers with more interconnections. We executed each simulation `4` times in a conventional desktop PC (`i7` CPU, `8GB` RAM, `512GB` SSD). Since the variations were negligible, we always discuss the median measures.

Evaluation item (a) measures the protocol overhead due to blockchain metadata, which consists of a timestamp, author signature, and hashes (block id, payload, backlinks, and likes). The original chat archive is `800kB` in size, or `80B` for each message, which includes a timestamp, a username, and the actual payload. The simulated chat chain is `8MB` in size, which indicates a $10x$ overhead. The original newsgroup is `30MB` in size, or `3kB` for each message, which includes a timestamp, a sender, a subject, and the actual payload (typically much longer). The simulated newsgroup chain is `42MB` in size, which indicates a $50\%$ overhead. It is clear that the metatada overhead is not negligible, specially for short chat messages, but decreases as the payload increases.

Evaluation item (b) accounts the consensus algorithm applied locally at each chain. We measured the time to sort blocks in a local DAG both for the first time (without any caches), and incrementally (from stable consensus caches). For the chat chain, it takes `125s` and `50ms` for the initial and incremental sorts respectively, while for the

[2]Chat: `https://archive.org/download/WikimediaIrcLogs/`
[3]Newsgroup: `https://archive.org/download/usenet-comp`

newsgroup chain, it takes `100s` and `70ms`. The incremental sort is limited to 7 days or 100 posts, regardless of the size of the chain, which conveniently settles an upper bound on the input size of the consensus algorithm. Given that we use plain JSON files in the file system, we consider an incremental consensus under `100ms` to be a practical upper bound.

Evaluation item (c) counts the number of forks in chain DAGs, which indicates the level of asynchrony between peers following the local-first principle (supposedly). We calculate the ratio of forks over the total number of messages, e.g., a DAG with $100$ messages and $10$ forks has a ratio of $10\%$. We found a ratio of $18\%$ for the chat and $14\%$ for the newsgroup, which confirms that the simulation achieves a reasonable level of asynchrony.

Evaluation item (d) measures how much bookkeeping is required to sustain active users in the forums. Even though the newbie rule `4.a` in Table 2 is key to combat Sybils, ideally it should not deny access to active users with low reputation recurrently. The evaluation counts the number of blocked messages requiring extra likes after the welcoming likes (which are disconsidered) and calculates the ratio over the total number of messages in the chain. As an example, if $10$ users posted $110$ messages requiring $20$ likes, we discount the $10$ initial messages and welcoming likes and find a ratio of $10\%$ (`(20-10)/(110-10)`). For the chat with $80$ users, we found a ratio of $3.7\%$. For the newsgroup with $5000$ users, we found a ratio of $3.5\%$. Considering that users were not aware of the reputation rules, the low ratios indicate that their "natural" posting behavior matches the constraints of the rules. We assume that users would use the revoke mechanism to combat abusive content, having no further effects on our evaluation.

## 5. Related Work

Many other systems have been proposed for decentralized content sharing [23, 7]. Here we consider the classes of publish-subscribe, federated, and peer-to-peer protocols.

Decentralized topic-based publish-subscribe protocols, such as *XMPP* [18], *ActivityPub* [26], and *gossipsub* [25], decouples publishers from subscribers in the network. A key limitation of *pubsubs* is that the brokers that mediate communication still have a special role in the network, such as authenticating and validating posts. Nevertheless, some pubsubs do not rely on server roles, and instead, use peer-to-peer gossip dissemination [3, 15, 12, 16, 25, 15]. Most of these protocols focus on techniques to achieve scalability and performance, such as throughput, load balancing, and real-time relaying. However, these techniques alone are not sufficient to operate permissionless networks with malicious Sybils [24]. Being generic protocols, pubsubs are typically unaware of the applications built on top of them. In contrast, as stated in Section 3, the pubsub of Freechains is conceptually at the application level and is integrated with the semantics of chains, which already verifies blocks at publishing time. For instance, to flood the network with posts, malicious peers need to spend reputation, which takes hours to recharge (rule `2` in Table 2). In addition, blocked posts (Figure 3) are not a concern either, because they have limited reachability. Another advantage of a tighter integration between the application and protocol is that Merkle DAGs simplify synchronization, provide persistence, and prevent duplication of messages. In summary, Freechains and *pubsub* middlewares operate at different network layers, which suggests that Freechains could benefit of the

latter to manage the interconnections between peers.

Federated protocols, such as e-mail, allow users from one domain to seamlessly exchange messages with users of other domains. *Diaspora*, *Matrix*, and *Mastodon* are recent federations for social media, chat, and microblogging [7], respectively. As a drawback, identities in federations are not portable across domains, which may become a problem when servers shutdown or users become unsatisfied with the service [1]. In any of these cases, users have to grab their content, move to another server, and announce a new identity to followers. Moderation is also a major concern in federations [7]. As an example, messages crossing domain boundaries may be subject to different policies that might affect delivery. With no coordinated consensus, it is difficult to make pervasive public forums practical. For this reason, Matrix supports a permissioned moderation system[4], but which applies only within clients, after the messages have already been flooded in the network. As a counterpoint, federated protocols seem to be more appropriate for real-time applications such as large chats rooms. The number of hops and header overhead can be much smaller in client-server architectures compared to peer-to-peer systems, which typically include message signing, hash linking, and extra verification rules.

Regarding peer-to-peer protocols, Bitcoin [13] is probably the most successful permissionless network, but serves specifically for electronic cash. IPFS [4] and Dat [17] are data-centric protocols for hosting large files and applications, respectively. Scuttlebutt [22] and Aether [7] are closer to Freechains goals and cover human-centric $1 \rightarrow N$ and $N \leftrightarrow N$ public communication, respectively [19].

Bitcoin adopts proof-of-work to achieve consensus, which does not solve the centralization issue entirely, given the high costs of equipment and energy. Proof-of-stake is a prominent alternative [2] that acknowledges that centralization is inevitable, and thus uses a function of time and wealth to elect peers to mint new blocks. As an advantage, these proof mechanisms are generic and apply to multiple domains, since they depend on an extrinsic scarce resource. In contrast, we chose an intrinsic resource, which is authored content in the chains themselves. We believe that human work grows more linearly with effort and is not directly portable across chains with different topics. These hypotheses support the intended decentralization of our system. Another distinction is that generic public ledgers require permanent connectivity to avoid forks, which opposes our local-first principle. This is because a token transaction only has value as part of the longest chain. This is not the case for a local message exchange between friends, which has value in itself.

IPFS [4] is centered around immutable content-addressed data, while Dat [17] around mutable pubkey-addressed data. IPFS is more suitable to share large and stable content such as movies and archives, while Dat is more suitable for dynamic content such as web apps. Both IPFS and Dat use DHTs as their underlying architectures, which are optimal to serve large and popular content, but not for search and discovery. In both cases, users need to know in advance what they want, such as the exact link to a movie or a particular identity in the network. On the one hand, DHTs are probably not the best architecture to model decentralized human communication with continuous feed updates. On the other hand, replicating large files across the network in Merkle DAGs is also

[4]Matrix moderation: `https://matrix.org/docs/guides/moderation`

impractical. An alternative is to use DHT links in Merkle payloads to benefit from both architectures.

Scuttlebutt [22] is designed around public identities that follow each other to form a graph of connections. This graph is replicated in the network topology as well as in data storage. For instance, if identity $A$ follows identity $B$, it means that the computer of $A$ connects to $B$'s in a few hops and also that it stores all of his posts locally. Scuttlebutt is aligned to $1{\rightarrow}N$ public identity chains of Freechains. For group $N{\leftrightarrow}N$ communication, Scuttlebutt uses the concept of channels, which are in fact nothing more than hash tags (e.g. *#sports*). Authors can tag posts, which appear not only in their feeds but also in local virtual feeds representing these channels. However, users only see channel posts from authors they already follow. In practice, channels simply merge friends posts and filter them by tags. In theory, to read all posts of a channel, a user would need to follow all users in the network (which also implies storing their feeds). A limitation of this model is that new users struggle to integrate in channel communities because their posts have no visibility at all. As a counterpoint, channels are safe places that do not suffer from abuse.

Aether [7] provides peer-to-peer public communities aligned with $N{\leftrightarrow}N$ public forums of Freechains. A fundamental difference is that Aether is designed for ephemeral, mutable posts with no intention to enforce global consensus across peers. Aether employs a very pragmatic approach to mitigate abuse in forums. It uses established techniques, such as proof-of-work to combat SPAM, and an innovative voting system to moderate forums, but which affects local instances only. In contrast, Freechains relies on its permissionless reputation and consensus mechanisms for moderation.

## 6. Conclusion

In this paper, we propose a permissionless consensus and reputation mechanism for social content sharing in peer-to-peer networks. We enumerate four main contributions: (i) human authored content as a scarce resource (*proof-of-authoring*); (ii) diversified public forums, each as an independent blockchain with subjective moderation rules; and (iii) abusive content removal preserving data integrity.

The key insight of the consensus mechanism is to use the human authoring ability as a scarce resource to determine consensus. This contrasts with extrinsic resources, such as CPU power, which are dispendious and not evenly distributed among people. Consensus is backed by a reputation system in which users can rate posts with likes and dislikes, which transfer reputation between them. The only way to forge reputation is by authoring new content under the judgement of other users. This way, reputation generation is expensive, while verification is cheap and decentralized.

The reputation and consensus mechanism is integrated into Freechains, a peer-to-peer protocol that offers multiple arrangements of public and private communications. Users have the power to create public forums of interest and apply diverse moderation policies. In particular, users can revoke content considered abusive according to the majority, not depending on centralized authorities. We simulate the behavior of existing chat and newsgroup forums as if they were using Freechains to show the practicability of the protocol as a descentralized alternative for public forums.

Finally, we do not claim that the proposed reputation system enforces "good" human behavior in any way. Instead, it provides a transparent and quantitative mechanism to

help users understand the evolution of forums and act accordingly. Human creativity contrasts with plain economic resources (e.g., proof-of-work), which do not appraise social interactions and also tend to concentrate over the time.

## References


[1] A. Auvolat. Making federated networks more distributed. In *2019 38th Symposium on Reliable Distributed Systems (SRDS)*, pages 383–3831. IEEE, 2019.

[2] L. M. Bach, B. Mihaljevic, and M. Zagar. Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550. IEEE, 2018.

[3] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni. TERA: Topic-Based Event Routing for Peer-to-Peer Architectures. In *Proceedings of the International Conference on Distributed Event-Based Systems*, pages 2–13, 2007.

[4] J. Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[5] J. R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[6] V. B. Gomes, M. Kleppmann, D. P. Mulligan, and A. R. Beresford. Verifying strong eventual consistency in distributed systems. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–28, 2017.

[7] J. Graber. Decentralized social ecosystem review. Technical report, BlueSky, 2021.

[8] M. Kleppmann. Making crdts byzantine fault tolerant. In *Proceedings of the 9th Workshop on Principles and Practice of Consistency for Distributed Data*, pages 8–15, 2022.

[9] M. Kleppmann, A. Wiggins, P. van Hardenberg, and M. McGranaghan. Local-first software: you own your data, in spite of the cloud. In *Proceedings of Onward'19*, pages 154–178, 2019.

[10] D. Koll, J. Li, and X. Fu. The good left undone: Advances and challenges in decentralizing online social networks. *Computer Communications*, 108:36–51, 2017.

[11] N. Masinde and K. Graffi. Peer-to-peer-based social networks: A comprehensive survey. *SN Computer Science*, 1(5):1–51, 2020.

[12] M. Matos, A. Nunes, R. Oliveira, and J. Pereira. Stan: exploiting shared interests without disclosing them in gossip-based publish/subscribe. In *IPTPS*, page 9, 2010.

[13] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, 2009.

[14] B. Nasrulin and J. Pouwelse. Decentralized collaborative version control. In *Proceedings of the 2nd International Workshop on Distributed Infrastructure for Common Good*, pages 11–16, 2021.

[15] J. A. Patel, É. Rivière, I. Gupta, and A.-M. Kermarrec. Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*, 53(13):2304–2320, 2009.

[16] F. Rahimian, S. Girdzijauskas, A. H. Payberah, and S. Haridi. Vitis: A gossip-based hybrid overlay for internet-scale publish/subscribe enabling rendezvous routing in unstructured overlay networks. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 746–757. IEEE, 2011.

[17] D. C. Robinson, J. A. Hand, M. B. Madsen, and K. R. McKelvey. The dat project, an open and decentralized research data tool. *Scientific data*, 5(1):1–4, 2018.

[18] P. Saint-Andre, K. Smith, R. Tronçon, and R. Troncon. *XMPP: the definitive guide*. " O'Reilly Media, Inc.", 2009.

[19] F. Sant'Anna, F. Bosisio, and L. Pires. Freechains: Disseminação de conteúdo peer-to-peer. In *Workshop on Tools, SBSeg'20*.

[20] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. In *Symposium on Self-Stabilizing Systems*, pages 386–400. Springer, 2011.

[21] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen. Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 5(1):63–108, 1998.

[22] D. Tarr et al. Secure scuttlebutt: An identity-centric protocol for subjective and decentralized applications. In *Proceedings of ACM ICN'19*, pages 1–11, 2019.

[23] S. A. Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, Dec. 2004.

[24] D. Vyzovitis, Y. Napora, D. McCormick, D. Dias, and Y. Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and eth2. 0 networks. Technical report, Protocol Labs, 2020.

[25] D. Vyzovitis and Y. Psaras. Gossipsub: A secure pubsub protocol for unstructured, decentralised p2p overlays. Technical report, Protocol Labs, 2019.

[26] C. Webber, J. Tallon, and O. Shepherd. Activitypub. *W3C Recommendation, W3C, Jan*, 2018.

[27] J. Zittrain. Fixing the internet. volume 362, pages 871–871. American Association for the Advancement of Science, 2018.