Adversarial information retrieval in distributed systems

Jelle Licht, Johan Pouwelse

***Abstract*—Trust is fundamental to any human interaction. Any web page, tweet, blog post, wikipedia edits, news article can be fake or real. Technology can assist in determining trustworthiness of information shared by strangers. A plentiful amount of techniques have been developed to do exactly this. However, many existing systems delegate ultimate trust to a predatory company, or assume that there is some out of band information available on which to base initial estimates of trustworthiness. A system without these drawbacks that consistently classifies peers as honest or dishonest does not currently exist. The goal of this paper is to evaluate and analyze the existing systems and their shortcomings, as well as look at ongoing efforts to make use of the properties of distributed ledger technology for a way forward.**

I. Introduction

Human interactions are based on trust between individuals. The willingness to trust others can be diminished by continuous dealings with incompetent or abusive behavior, and technological systems are not exempt from this. Indeed, whereas advances in predictive models and behavioral analytics can advise us regarding the trustworthiness of certain individuals, the value judgment of whether to trust is still a essentially human one. This problem is exacerbated when assessing the trustworthiness of hundreds, thousands or even millions of individuals, as is often the case in decentralized content sharing systems.

These systems rely on trust for their core operations once the amount of content is intractable for an individual to verify and/or authenticate. Some sort of gatekeeping or filtering is required. For systems like FaceBook, this responsibility is delegated to a central authority. A drawback of allowing FaceBook, or any other central authority for that matter, to make these decisions is that it can lead to inadvertently taking part in a emotion study [1]. A general lack of transparency on the inner workings of such a system [2] make it difficult to identify biases in the selection of content as presented. A decentralized and transparent solution is needed with accountability and trust built-in. More specifically, we take a view that focuses on dealing with spam in these systems.

This work means to provide insight into the existing designs and techniques in this area, as well as provide a framework with which to analyze and compare the shortcomings of existing and proposed solutions. In this paper we demonstrate the various ways in which existing systems deal with dishonest peers, specifically by making use of votes cast by peers. Our major contributions are as follows:

- A survey of different dealings with the potential maliciousness of peer-supplied information in decentralized systems.
- A case study showing how a real-world file-sharing systems makes use of votes cast by users to improve relevance of search results.

The rest of this paper is organized as follows: Section II introduces the problems as tackled by this paper, as well describing the general challenges plaguing decentralized systems. Section III describes mitigation strategies which apply only in specific cases and a more general approach to building trust. Section IV-D regards a case study using Tribler, a real-world fully distributed content sharing and streaming system Section V gives the concluding remarks of this paper.

II. Problem description

It is hard to divide the world in good and evil, even more so for a machine. When exchanging information with a multitude of individuals, the distinction has to be made between honest and dishonest peers in order to make informed decisions.

A key feature of decentralized systems is information sharing. In any but the most trivial systems, peers can not feasibly share all information available in the system: Peers need to use information retrieval techniques to find relevant information. Information retrieval refers to the activity of finding relevant information with regards to a certain query.

A dishonest peer can influence the process of information retrieval depending on the specific system architecture. By gaining control of trusted parts of the system, one can choose to ignore, subvert or simply monitor peer interaction with the decentralized system. This can be effectively equivalent to denying certain or all peers service or threatening legal actions to users of such a system. Deciding which peers to trust is therefore paramount to the functioning of the system.

Decentralized system architectures were originally used and actively researched as a means to make systems more robust against censorship. Properly designed decentralized system exhibit attributes such as scalability, trustworthiness and reliability as well. Since 1999 the usage of p2p file-sharing systems has steadily increased, bringing with it an increased amount of interest in how these systems function. This has been a double edged sword, as interest come from both well-meaning users as well as more adversarial parties aiming to sabotage these systems. A modern decentralized file-sharing system needs. to take the motivations of adversaries and risks for legitimate users of the system into account. Tribler is one

jlicht@fsfe.org, j.a.pouwelse@ewi.tudelft.nl

[1] https://www.theguardian.com/technology/2014/jun/30/facebook-news-feed-filters-emotion-study

[2] http://raley.english.ucsb.edu/wp-content/Engl800/Pasquale-blackbox.pdf

of such decentralized file-sharing systems that aims to provide users with a robust and censorship-proof service.

Modern examples of widely used decentralized systems are the BitCoin blockchain [3], BitTorrent [4] and the Internet itself. Decentralization trends that have been ongoing since the inception of the Internet, combined with bursts of intense research activity have contributed to a sort of arms race between designers of decentralized systems and those perceiving harm by the successful development and deployment of decentralized systems. This has led to the creation increasingly complex decentralization schemes, while these adversaries have come up with social, legal and technical means to prevent designers and users of these systems from being successful in going about their business.

Interactions with honest peers can then be trusted, as well as the corollary of this statement. Voting systems can alleviate some of the more serious weaknesses seen in most file-sharing systems. A potential issue with basing spam prevention measures on active user participation is that users might not be properly incentivized to act for the greater good of the system. In the most extreme cases, entities can try to shut down all services by launching a DDoS attack conducted either in the open or anonymously [19].

Most of the theoretical models rely on two central assumptions:

1) Cooperative users vote in a similar way, properly classifying spam vs authentic content.
2) Cooperative users vote.

Prior research challenges both of these assumptions in an empirical study [16]. As the first assumption can be seen as a stronger one than the second, disproving the second also allows the first to shown as optimistic in realistic scenarios.

In some way, determining the authenticity of content based on the statements of potentially untrustworthy users is similar to a Byzantine Fault Tolerance problem [15]. As a corollary, this also means that cooperative users often inadvertently help polluters spread misinformation and spam in the network without being aware of this. Any proposed solution for dividing the world in good and evil has to take into account that even honest peers can commit to abusive behavior because of ignorance.

We aim to give an overview of the abuse types adversaries have taken to prevent users from finding relevant content by spamming some part of the service.

[1]

A. Index poisoning and routing table poisoning

A relatively simple way of preventing a user from downloading certain content is making sure the user has no way of finding said content. As described in [30], index poisoning can take place when users depend on other, potentially malicious users for locating content. Index poisoning takes place when users have no way to distinguish content advertised by malicious vs cooperative users. Index poisoning is effective because an adversary only has to advertise non-existing or corrupt locations of content, after which a naive user will start the expensive process of following up on finding this advertised content.

If an adversary is able to advertise the misinformation in a superior way, this can lead to users repeatedly trying to make use of the corrupted index information before eventually stumbling on a correctly advertised piece of content by happenstance. By then, the damage is done, as most proper decentralized file-sharing systems rely on other users dealing with the same piece of content before being able to download it.

Another issue starts to rear its head when an adversary is able to contribute and sustain large enough of ostensibly cooperative peers to the network. As long as no malicious behavior is undertaken or detected, adversary-controlled nodes start to become entwined in the distributed routing tables of normal users. An adversary can employ this position of power to monitor traffic, or even deny services to any subset of users, or event deny services related to a specific piece of content [21].

A way to deal with index- and routing table poisoning is to routinely label misleading information, and purge it from local information stores. The advantages of this approach are two-fold. First of all, the peer will no longer make use of misleading information, and secondly the peer will no longer contribute to the problem by distributing the misleading information. The challenge then becomes to identify with reasonable certainty which pieces of information are misleading.

B. Content poisoning

An orthogonal method for the adversary to deny users of p2p networks services is to actively flood the network with mislabeled content. Compared to the index poisoning and routing table poisoning method, this method does actually lead to content being available on the network [18]. If the content has to be downloaded in its entirety before a user is able to determine its authenticity, this can quickly lead to entire swarms of peers downloading and in turn sharing spam. One way of dealing with content poisoning is by estimating the probability of the content being authentic.

To estimate the pollution of a file-sharing network, the authors of [18] differentiate between natural and intentional pollution, but conclude that natural pollution is usually limited to a negligible amount. A crawler collects metadata an availability information on the content in a certain network in a best-effort to create a snapshot.

C. Stream poisoning

The application of live p2p streaming of content gives an extra set of constraints which make it especially sensitive to stream poisoning. Dishonest peers can collude to periodically upload corrupt data to honest peers. Depending on how robust the transfers are, honest peers might have to re-download blocks, chunks or even entire files, thereby slowing down the network with unnecessary work. These same considerations also exist for non-live streaming systems, but in that case a system can already be considered usable if honest peers are

[3] https://www.bitcoin.com/
[4] http://www.bittorrent.com

able to use the system within some reasonable amount of time. On the other hand, even a two minute delay could already be considered too much for a live streaming system.

D. Tag spam

Tagging is the process of annotating content with a tag. P2p systems can benefit from tagging by providing a self-regulating fine-grained filter. When tags are properly applied to content, search performance should go up by more clearly describing and distinguishing the content available in a network.

E. votes-spam

If the assumption holds that colluders act in detectable patterns, techniques employed for identifying Web link farm spam pages can also be employed on partial views of a network [29], [5]

SumUp [26] is one the systems designed to be resilient against large swaths of colluding malicious users. It limits the amount of influence colluders have by introducing a bastardized version of the MaxFlow problem.

TorrentTrust [24] extends upon the Credence system by taking into account user trust. The authors also incorrectly state that Credence is by definition a centralized scheme with a centralized certificate issuer. This is arguably the case for the implementation of Credence, but [27] clearly states that any different gate-keeping scheme can be used.

III. Solutions

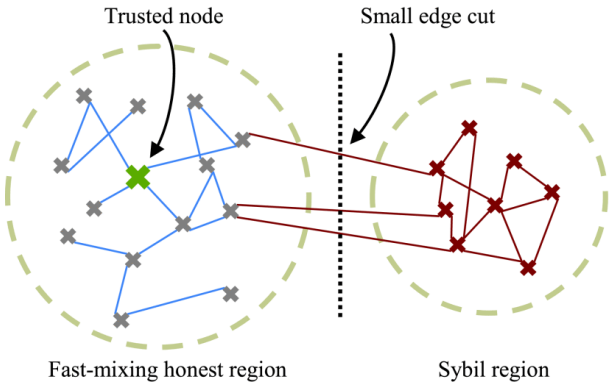
This section provides an overview of mitigation strategies to deter or limit the effect an adversary can have on the quality of service of the file-sharing system. Ways to prevent dishonest peers from being successful can be labeled in one of two types. The first one is to prevent or disincentivize an adversary from exhibiting the malicious behavior. The second type mitigates or isolates the effects of the malicious behavior such that honest peers can make use of the service unhindered.

Having defined the plethora of ways in which an decentralized file-sharing system can be abuses and secured in Section II, the decision has to be made on how to categorize gathered information according to trustworthiness.

Building trust among peers is a well researched topic, and one way of looking at the problem is to see interactions between peers as edges in a graph, with vertices denoting peers. Verifying an interaction implies trusting another peer to some extent. This way of defining trust in a network leads to Web of Trust.

A. Cold start based trust building

Credence [27] introduces the concept of object-based reputation as a way to estimate content authenticity. A simple voting protocol is used where a vote can be cryptographically traced back to the voter. In principle, an adversary can employ a Sybil attack [9] and quickly generate lots of misleading votes. Figure III-A outlines the network topology for a Sybil attack. Any technique that makes the edge cut between the honest region and Sybil region smaller can be effective in practice A gate-keeper mechanism that disincentivizes joining the network multiple times in rapid succession would alleviate this issue, such as a cryptographic challenge comparable to the proof of work mechanism as employed on certain blockchain structures and similar the scheme proposed in [1].



Semantically, Credence assumes a positive vote to be a vote of confidence that the content is authentic; they do not have to be indicative of popularity, although in practice users may complect the two. Cooperative peers will generally consistently vote to correctly classify authentic and non-authentic content. Credence uses a weighted average scheme to estimate the authenticity of each content item based on the collected votes. Using the set of content items on which two peers both voted, a local weight is assigned as follows: Assuming that all cooperative peers vote in a similar fashion, looking at the correlation between voting behavior allows a peer to quickly asses the honesty of a different peer.

SpamResist [36] follows a scheme similar to Credence and Sorcery [33], albeit applied to tag clouds. By dividing peers in two groups, the *unfamiliar peers* and the *interacted peers*, two different heuristics with widely different characteristics can be applied. SpamClean [35] is another way to combat tag spam. Again the authors make use of the insight that users with similar voting behavior can provide more reliable information in general, while also leading to degraded performance of the system for deviants and dishonest peers.

SpamLimit [2]...[6].

Scrubber introduces a scheme which allows for swift punishment of malicious users, while still allowing redemption as a result of continuous honest behavior [6]. It operates on the assumption that at least 25% of users react to punishment by removing the polluted content from the network.

The authors of [12] propose a quorum-based approach, where peers can create a quorum to assess the honesty of certain peers based on ad-hoc voting procedure. As long as honest peers participate in these voting schemes this can be effect, but as the authors of [32] noted, user participation in voting is usually nonexistent or wildly inaccurate in the best of cases. This leads us to believe that earlier mentioned schemes,

[5] TODO: Describe [17]. (Collusion detection in completely-known systems "works", should/could be applicable to online systems. Not sure whether knowing you are being detected changes collusion behaviour.

[6] TODO: Compare SpamLimit vs SpamResist vs SpamClean

while performing admirably in a simulation or experiment, give no guarantees for real world deployments.

Some special considerations have to be made for preventing dishonest peers from sharing corrupted data with honest peers in live streaming systems; honest peers repeatedly trying to get an uncorrupted copy of the data can shut down the system if left unchallenged. A trivial solution would be to create a published checksum for each unit of data, although this leads to a problematic amount of metadata quickly. The authors of [10] introduce a system for honest peers to make a converging assessment regarding the trustworthiness of peer groups by determining and refining probabilities of a specific peer being complicit in repeatedly delivering corrupted data. By taking into account that not each peer is responsible for the same piece of data, this can strike an effective balance between the amount of metadata required and the sensitivity to abuse by dishonest peers.

B. Social enrichment

Most p2p networks have users use information contained within the system to determine which peers to trust. This can be problematic for users who only joined the network recently, as they might not have all the information to correctly categorize peers they interact with. The situation turns into a bootstrap problem, where being initially basing your behavior on information begotten from dishonest peers leads to honest colluding with dishonest peers without by accident.

A different approach allows for using out-of-band information to enrich the information gathered from within the system. One such system is Sorcery as introduced in [33] and [34], adding social network information as a source of baseline truth, thereby addressing the bootstrap problem. Sorcery uses this baseline truth to issue challenges to peers of which no prior knowledge is available. Comparing challenge responses to the baseline truth allows each peer to estimate a relative reliability connected peer in the network.

SybilInfer as proposed in [8] relies on knowledge of all social interaction in a network overlay in order to determine the likelyhood of peers being either honest or dishonest.

IV. Search in existing systems

A glaring flaw in several of the proposed solutions is the fact that most of them have not been deployed and used by real life users. Simulations as run in some of the existing research is can provide valuable insight into the dynamics and steady-state behavior of systems, but mean nothing if some of the initial assumptions made for the system or simulation do not hold in the real world. In this section, we show the black box behavior of several supposedly mature decentralized systems.

A. Information retrieval in Freenet

Freenet [7] is a system which allows its users to make use of web services hosted on the Freenet platform while still providing a modicum of privacy. Development of Freenet has focused on providing the platform for others to offer services on top of. Indices can be created by anyone, with a de facto set of indices being proposed on the starting page, such as the Filtered Index seen in Figure [[fig:freenet] . These indices are a simple service like any other, and are usually curated by a group of individuals. Interacting with resources on the Freenet platform is done by establishing a connection to the network and sending queries for specific resource. Freenet allows for searching its network of content, but this usually leads to significant delays, while depending on the current peers as well as the state of the local cache. Effectively, making use of the curated index services is more reliable when looking for content. Freenet has been development for 17 years and the platform is still in active development. The technical platform leaves users to their own devices on deciding which services and what content to trust, and as such could be seen to leave spam prevention entirely up to the community as a social challenge.

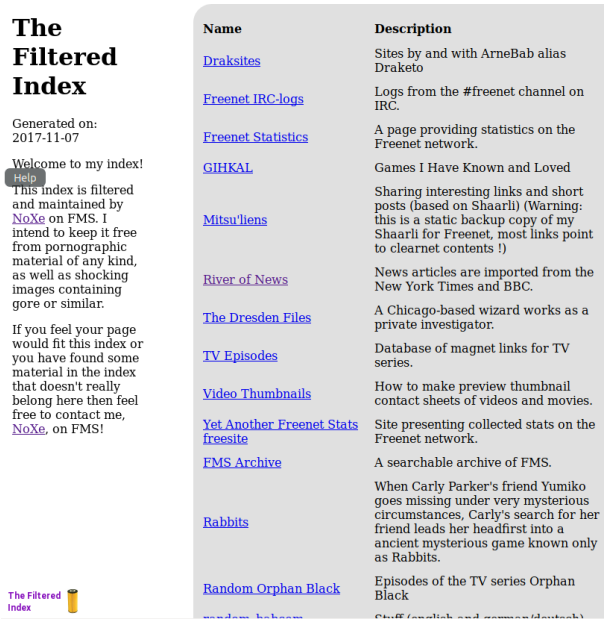


Fig. 1. An example of a Freenet curated index, allowing users to find non-spam content.

B. Information retrieval in gnunet

GNUnet [8] is a reputation-based network with a focus on anonymity. It can either be run on top of an exiting TCP/IP stack, or replace it with the GNUnet stack entirely, although this has mostly been an academic exercise.

The goals of the GNUnet project can seen as quite ambitious, at the time of this writing there are no pieces of software readily available that work with modern operating systems and dependencies. TODO: Contact Christian on why nothing builds.

GNUnet supports anonymous search queries by allowing peers to forward queries, resulting in an inability to discern original queries compared to forwarded ones. Responders to queries have to prove they have actual knowledge about the

[7] https://freenetproject.org/

[8] https://gnunet.org/bibliography?page=2&f[keyword]=2

looked-for content via a layered hashing system, comparable to a Zero-knowledge proof TODO: {Ask Christian for reference regarding ZKP}[11].

C. Information retrieval in YaCy

YaCy aims to be a fully decentralized search engine which can provide an alternative to services such as the Google search engine [9]. Compared to the other systems reviewed here, the scope of the problem YaCy aims to solve is more limited; it is mainly used to index content in *existing* networks, such as the Internet, but is not limited to this use-case. Figure 2 demonstrates the basic interface of YaCy.

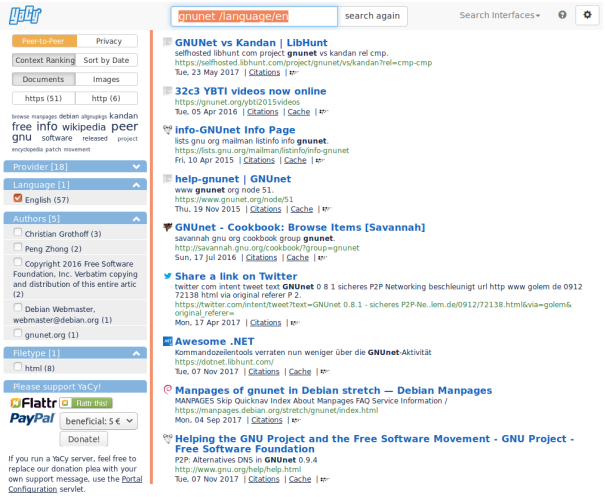


Fig. 2. The YaCy search interface, showing all options

D. Information retrieval in Tribler

Tribler search functionality focuses on three key requirements: fast results, correct results and spam protection [10]. Tribler started out as fork of Yet Another BitTorrent Client, aiming to use social networks to enhance the user experience. Active research has extended Tribler to make it a tool for researchers of decentralized systems to run experiments in the real world. Tribler is compatible with other BitTorrent clients. Users of Tribler can discover content from other peers via a gossip protocol.

1) Communities: Tribler is extendable by introducing communities. A Tribler community is a network overlay via which peers can exchange predefined messages. The SearchCommunity is a Tribler community used to share and receive partial torrent files, allowing users to actually search for content on the Tribler network in a decentralized manner. Each user can have any number of content-channels associated with them. A content channel is essentially a collection of torrent files, as well as an assorted list of subscribers [31]. The AllChannel community stores users' vote preference for content channels, and shares known votes among peers, allowing for filtering based via a distributed moderation system. A vote can either be positive (or "favorite"), or negative (or "spam"), as defined by the VoteCast protocol [11]. Tribler also allows peers to change their mind at a later time and revoke their vote. Changes in voting preferences are propagated over the network via a gossip protocol.

2) Voting data: Tribler stores the current beliefs about of the vote counts per channel in a local database, allowing for offline analysis of voting behaviour within the network. We need an understanding of user voting behavior to evaluate how resilient Tribler is to vote-based spam.

The storage scheme as of this writing allows us to create a coarse overview of how popular each content channel is by calculating an effective vote count per channel. We subtract the number of 'spam' votes from the number of 'favorite' votes for a specific channel, reaching a number of effective votes.

3) Analysis: The VoteCast data crawled from the AllChannel over several hours allows us to see how popularity is distributed over the channels in Tribler. Looking at Figure 3, we see that a channel on average has 127 votes, with only 45 channels having more than 1000 votes. The gathered data leads us to the conclusion that there are a handful of reasonably popular channels, and a myriad of less-popular channels.

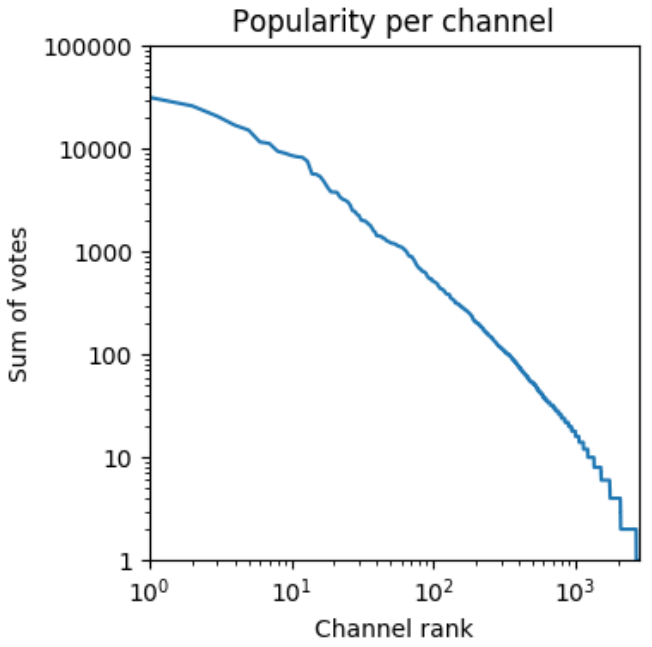


Fig. 3. Content channel popularity, demonstrating that all but the 40 most popular channels only have a handful of subscribers.

4) Votes over time: Giving a closer look to the gathered VoteCast data reveals that there exist two clusters of votes which predate the existence of Tribler, and by extension VoteCast, by respectively 28 and 6 years. It can be assumed that these anomalies are either the result of either a bug or a curious user testing the limit of the VoteCast validation logic. These findings have no bearing on further elaboration of vote-spam behaviour in Tribler. See Figure 4 for the raw plot of this data, including the anomalous past votes.

[9] https://yacy.net/en/index.html
[10] https://www.tribler.org/ContentSearch/
[11] https://www.tribler.org/Votecast/

Figure 5 shows how votes are distributed when properly filtered. We see that the first few votes were steadily cast, after which the bigger group of users started using the VoteCast system. [12]

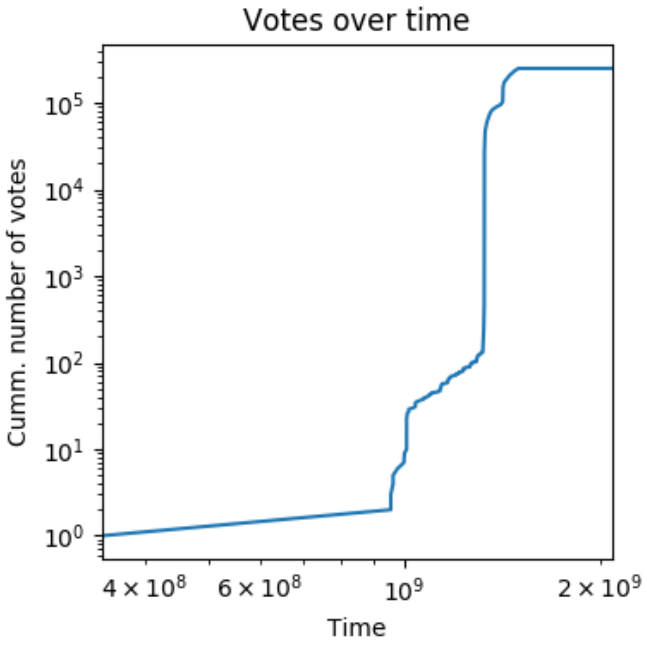


Fig. 4. Voting behaviour over time, unfiltered to include phantom votes.

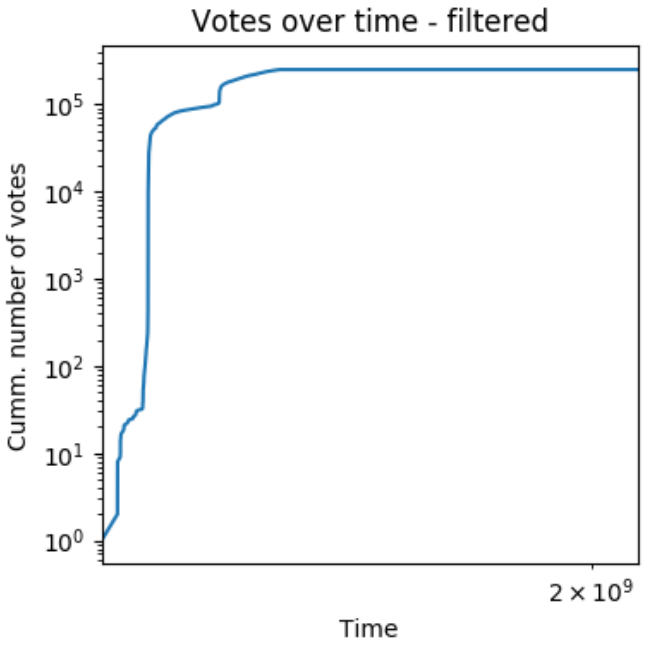


Fig. 5. Voting behaviour over time, filtered to show the distribution of legitimitely cast votes.

V. Conclusions

Clearly delineating the set of honest from dishonest peers in a fully decentralized system is a difficult and for the general case unsolved problem. The cold-start problem has newly joined peers having trouble judging the authenticity of claims made by peers in the network, while the lack of an accepted ground truth necessitates heuristics and compromises to establish an initial bearing on trustworthiness. Adding out of band information to assist systems with making these decisions alleviates the cold start problem, but this is not a general solution for systems without persistent identities or systems with high churn.

Honest peers can be misled and contribute to the activities of dishonest peers in various ways. The effects of such poisoned peers in the network should be mitigated, while allowing poisoned peers redemption in the long term if their behavior is corrected. Incentivizing peers to participate in the human-driven quality assurance process that lies at the basis of most technical solutions to identify dishonest peers is also still an open problem.

The advent of tamper-proof distributed ledger technologies brings with it a renewed interest in creating a Sybil-resistant scheme for identifying dishonest peers, by making peers accountable for their behavior. As of this writing, no scalable system has been deployed with magnitudes comparable to BitTorrent.

References

[1] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. “DOS-resistant authentication with client puzzles”. In: *International workshop on security protocols*. Springer. 2000, pp. 170–177.

[2] Eric Chang. “Defending against Spam in Tagging Systems via Reputations”. In: (2016). URL: http://dlc.dlib.indiana.edu/dlc/handle/10535/10221 (visited on 07/17/2017).

[3] Nitin Chiluka et al. “Leveraging trust and distrust for sybil-tolerant voting in online social media”. In: *Proceedings of the 1st Workshop on Privacy and Security in Online Social Media*. ACM. 2012, p. 1.

[4] Edith Cohen, Amos Fiat, and Haim Kaplan. “Associative search in peer to peer networks: Harnessing latent semantics”. In: *Computer Networks* 51.8 (2007), pp. 1861–1881.

[5] Fabrizio Cornelli et al. “Choosing reputable servents in a P2P network”. In: *Proceedings of the 11th international conference on World Wide Web*. ACM. 2002, pp. 376–386.

[6] C. Costa and J. Almeida. “Reputation Systems for Fighting Pollution in Peer-to-Peer File Sharing Systems”. In: *Seventh IEEE International Conference on Peer-to-Peer Computing (P2P 2007)*. Sept. 2007, pp. 53–60. DOI: 10.1109/P2P.2007.15.

[12] http://demo.polr.me/7

TABLE I
OVERVIEW OF COMPARED METHODS FOR IDENTIFYING SPAM IN DECENTRALIZED NETWORKS.

System	Decentralization	Out-of-band information	Identify Sybil region	Manual interaction	Real-life demo
PageTrust [14]	Unstructured	None?	Yes	Yes	?
Sorcery		Pre-defined trust edges?	Yes	Yes?	Yes
EigenTrust					
[3]					
[25]					
[7]					

[7] Arturo Crespo and Hector Garcia-Molina. "Routing indices for peer-to-peer systems". In: *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*. IEEE. 2002, pp. 23–32.

[8] George Danezis and Prateek Mittal. "SybilInfer: Detecting Sybil Nodes using Social Networks." In: *NDSS*. San Diego, CA. 2009.

[9] John R Douceur. "The sybil attack". In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 251–260.

[10] Rossano Gaeta, Marco Grangetto, and Lorenzo Bovio. "DIP: Distributed Identification of Polluters in P2P Live Streaming". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 10.3 (Apr. 2014), 24:1–24:20. ISSN: 1551-6857. DOI: 10.1145/2568223. URL: http://doi.acm.org/10.1145/2568223 (visited on 07/17/2017).

[11] Christian Grothoff et al. "The gnet whitepaper". In: *Purdue University* (2002).

[12] Hatem Ismail, Daniel Germanus, and Neeraj Suri. "P2P routing table poisoning: A quorum-based sanitizing approach". In: *Computers & Security* 65 (2017), pp. 283–299. URL: http://www.sciencedirect.com/science/article/pii/S016740481630178X (visited on 07/17/2017).

[13] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. "The eigentrust algorithm for reputation management in p2p networks". In: *Proceedings of the 12th international conference on World Wide Web*. ACM. 2003, pp. 640–651.

[14] Cristobald de Kerchove and Paul Van Dooren. "The pagetrust algorithm: How to rank web pages when negative links are allowed?" In: *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM. 2008, pp. 346–352.

[15] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine generals problem". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3 (1982), pp. 382–401.

[16] Uichin Lee et al. "Understanding Pollution Dynamics in P2P File Sharing." In: *IPTPS*. Vol. 6. 2006, pp. 1–6. URL: http://netlab.cs.ucla.edu/internal/wiki-internal/files/uclee2006iptps.pdf (visited on 07/16/2017).

[17] Qiao Lian et al. "An empirical study of collusion behavior in the Maze P2P file-sharing system". In: *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*. IEEE, 2007, pp. 56–56. URL: http://ieeexplore.ieee.org/abstract/document/4268209/ (visited on 07/16/2017).

[18] Jian Liang, Naoum Naoumov, and Keith W. Ross. "Efficient blacklisting and pollution-level estimation in p2p file-sharing systems". In: *AINTEC* 3837 (2005), pp. 1–21. URL: http://link.springer.com/content/pdf/10.1007/11599593.pdf#page=10 (visited on 07/17/2017).

[19] Haiman Lin et al. "Conducting routing table poisoning attack in DHT networks". In: *Communications, Circuits and Systems (ICCCAS), 2010 International Conference on*. IEEE, 2010, pp. 254–258. URL: http://ieeexplore.ieee.org/abstract/document/5582015/ (visited on 07/16/2017).

[20] Qin Lv et al. "Search and replication in unstructured peer-to-peer networks". In: *Proceedings of the 16th international conference on Supercomputing*. ACM. 2002, pp. 84–95.

[21] Naoum Naoumov and Keith Ross. "Exploiting p2p systems for ddos attacks". In: *Proceedings of the 1st international conference on Scalable information systems*. ACM. 2006, p. 47.

[22] Johan Pouwelse and Martijn de Vos. "Laws for Creating Trust in the Blockchain Age". unpublished paper. 2017.

[23] Sean C Rhea and John Kubiatowicz. "Probabilistic location and routing". In: *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 3. IEEE. 2002, pp. 1248–1257.

[24] Ian Sibner et al. "TorrentTrust: A Trust-Based, Decentralized Object Reputation Network". In: (2016).

[25] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. "Enabling efficient content location and retrieval in peer-to-peer systems by exploiting locality in interests". In: *ACM SIGCOMM Computer Communication Review* 32.1 (2002), pp. 80–80.

[26] Dinh Nguyen Tran et al. "Sybil-Resilient Online Content Voting." In: *NSDI*. Vol. 9. 2009, pp. 15–28. URL: https://www.usenix.org/legacy/events/nsdi09/tech/full_papers/tran/tran_html/ (visited on 07/16/2017).

[27] Kevin Walsh and Emin Gun Sirer. *Thwarting p2p pollution using object reputation*. Tech. rep. Cornell University, 2005. URL: https://ecommons.cornell.edu/handle/1813/5680 (visited on 07/17/2017).

[28] Yongang Wang et al. "Dspam: Defending against spam in tagging systems via users' reliability". In: *Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on*. IEEE. 2010, pp. 139–146.

[29] Baoning Wu and Brian D. Davison. "Identifying link farm spam pages". In: *Special interest tracks and posters of the 14th international conference on World*

Wide Web. ACM, 2005, pp. 820–829. URL: http://dl.acm.org/citation.cfm?id=1062762 (visited on 07/16/2017).

[30] Quan Yuan et al. "AStudy OF INDEX POISONING IN PEER-TO-PEER FILE SHARING SYSTEMS". In: (). URL: https://pdfs.semanticscholar.org/ef1e/10d6047a1c2f2a07d0214d29092f1270e810.pdf (visited on 07/17/2017).

[31] N. Zeilemaker et al. "Tribler: Search and stream". In: *2011 IEEE International Conference on Peer-to-Peer Computing*. Aug. 2011, pp. 164–165. DOI: 10.1109/P2P.2011.6038729.

[32] Ennan Zhai et al. "Resisting tag spam by leveraging implicit user behaviors". In: *Proceedings of the VLDB Endowment* 10.3 (2016), pp. 241–252. URL: http://dl.acm.org/citation.cfm?id=3021939 (visited on 07/17/2017).

[33] Ennan Zhai et al. "Sorcery: Could we make P2P content sharing systems robust to deceivers?" In: *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE, 2009, pp. 11–20. URL: http://ieeexplore.ieee.org/abstract/document/5284532/ (visited on 07/16/2017).

[34] Ennan Zhai et al. "Sorcery: Overcoming deceptive votes in P2P content sharing systems". en. In: *Peer-to-Peer Netw. Appl.* 4.2 (June 2011), pp. 178–191. ISSN: 1936-6442, 1936-6450. DOI: 10.1007/s12083-010-0074-2. URL: https://link.springer.com/article/10.1007/s12083-010-0074-2.

[35] Ennan Zhai et al. "Spamclean: Towards spam-free tagging systems". In: *Computational Science and Engineering, 2009. CSE'09. International Conference on*. Vol. 4. IEEE, 2009, pp. 429–435. URL: http://ieeexplore.ieee.org/abstract/document/5284153/ (visited on 07/16/2017).

[36] Ennan Zhai et al. "SpamResist: making peer-to-peer tagging systems robust to spam". In: *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6. URL: http://ieeexplore.ieee.org/abstract/document/5425801/ (visited on 07/16/2017).

VI. **TODO** PUT IN PROPER CONTEXT

- [13] Assign each peer a global trust value (via Power Iterations??). Using transitive trust, leads to 'a system where global trust values correspond to the left principal eigenvector of a matrix of normalized local trust value'. Dealing with 'malicious collectives' is done by adding a pre-set possibility of crawling to a predetermined trustworthy peer instead of only the self-reinforcing links within the malicious collective. Pre-trusted peers are required for this to work (ie converge and be aperiodic) in any way. M score managers are assigned to each peer via DHT who democratically present that peers trust score on request. (NB, score managers need to properly pass on their state and computations to their DHT neighbors when leaving the network).
- [22] wrt interaction rules in trust systems
- [28] wrt tag spam
- [20] Centralized vs decentralized + structured vs decentralized + unstructured. This paper applies random graph walks on decentralized + unstructured systems like Gnutella, and shows an improvement of resource usage at the scale of two orders of magnitude compared to the old Gnutella flooding scheme. Expanding ringe (increase TTL each time while flooding, expecting 'hot' items to be replicated more often (and thus, closely to peer issuing the query). k random walkers after T steps should/could have reached the same nodes as 1 walker after kT steps. And indeed simulations confirm this. TODO: WHICH. TTL walkers continue until TTL is 0 or the object has been located, while 'checking' walkers check in with requester to see if the object has already been located by a different walker. Magical number of 16-64 walkers seem to work, with checking after every 4 node. State keeping per random walk set makes sure the same routes are not repeated for a certain node, but this only seems to work in simulations and not in existing networks. Squareroot replication is needed to limit overhead, which can happen with owner replication (searcher stores object as well), or path replication (each node along the search path stores object). Random replication looks at the final hop count for the query, and selects 'hop count' nodes from all the visited nodes to replicate the object. In experiments, this seems to have the best results compared to other replication strategies. This is more difficult to implement correctly however. Any solution should "adaptive termination, minimizing message duplication and small granularity of coverage" when making queries.
- [4]
- [23] Attenuated loom filter = lossy distributed index. Route queries from a client to the closest replica adhering to certain properties (such as shortest network path). These consist of layer of bloom filter per edge, with each layer representing the bloom filter for the {1,2,3,...} hop document content, although this is an inherently non-deterministic process (WHY?)
- [7] Search without index, search with index nodes (centralized search) and index-per-node (distributed search). This document describes storing routes instead of indices, (which seems a bit like DHT?), allowing each 'step' to get closer to the content, while being robust against dynamic network topology. Index size => O(num of neighors) instead of O(num of documents)? Compound RI vs Exponential RI (finger table?) vs Hop count RI (DHT again?)
- [25] Early approach to "neighbourhoods" or taste buddies: Allow the system to keep track of a group of peers with matching performance characteristics and relevant content availability, while still falling back to Chord (TODO: Is this DHT?) and eventually flooding for looking up content. Neighborhoods are defined as being "peers who have the content we are looking for"???
- [5] "The use of a P2P network for information exchange involves two phases, the first phase is the search of

the servent where the requested information resides. The second phase, which occurs when a servent has identified another servent exporting a resource of interest, requires to establish a direct connection to transfer the resource from the exporting servent to the searching servent"

- [3] "Reducing the effect per attack edge of sybil region, by combining Social Network-based Sybil Defenses (SNSD) and Signed Network Analysis (trust and distrust relationships). They need ~10% vigilant nodes to make some things work, which is actually quite high according to earlier research about people not voting properly.
- [14] Extending PageRank to incorporate negative links, which would normally invalidate the non-negativity assumption from the original PageRank theory (TODO: Ref pageRank?. By working around this problem by defining a distrust matrix which "works both ways?", negative edges can be incorporated in PageRank, re-interpreting where a random walker should go (and thus ends up in the steady state) => resulting in PageTrust with potentially better results.