Adversarial information retrieval in distributed systems

Jelle Licht, Johan Pouwelse*

August 24, 2017

Abstract

TODO: placeholder

1 Introduction

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

In this paper we demonstrate the various ways in which existing systems deal with spam, specifically by making use of votes cast by peers. Our major contributions are as follows:

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

The rest of this paper is organized as follows: Section 2 introduces the terminology used in the rest of the paper, as well describing the general challenges plaguing decentralized systems. Section 3 regards a case study

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

using Tribler, a real-world fully distributed content sharing and streaming system. Section 4 outlines the ongoing interplay between state-of-the-art and adversaries by describing the moving attack surface of spam in distributed systems, followed by a description of mitigation strategies in Section 5 which apply only in specific cases. Section 6 gives the concluding remarks of this paper.

2 Background

TODO: Terminology. Decentralized system architectures were originally used and actively researched as a means to make systems more robust against censorship. Properly designed decentralized system can exhibit attributes such as scalability, trustworthiness and reliability as well.

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. The adversary can influence the process of information retrieval depending on the specific system architecture. By gaining control of trusted part of the system, an adversary 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.

Modern examples of widely used decentralized systems are the BitCoin blockchain, p2p networks such as BitTorrent 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 businessTODO: REFS/footnotes!?.

3 Information retrieval in Tribler

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 TODO: CIT NEEDED. Users of Tribler can discover content from other peers via a gossip protocolTODO: [?]. Tribler search functionality focuses on three key requirements: fast results, correct results and spam protection [1].

3.1 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 [17]. 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[2]. A vote can either be positive (or "favorite"), or negative (or "spam"), as defined by the VoteCast protocol TODO: {Footnote or ref?}. 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.

3.2 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.

3.3 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 1, we see that a channel on average has 127 votes, with only 45

[1] `https://www.tribler.org/ContentSearch/`
[2] TODO: Not sure if this is still true, ask Martijn

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.

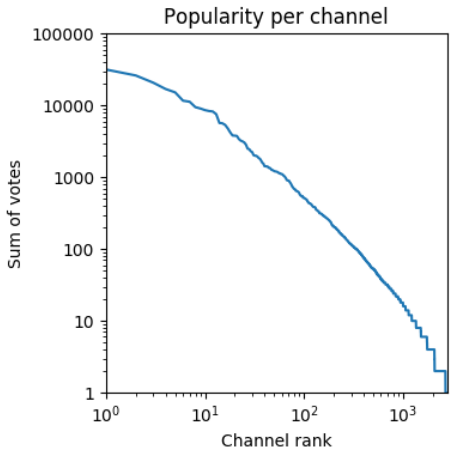

Figure 1: TODO: Re-make the nice xkcd plot

3.4 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 2 for the raw plot of this data, including the anomalous past votes.

Figure 3 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. [3]

4 Defining the area of attack

Denying users of a decentralized systems services can be as simple as disconnecting individual users from the system. If this is technically, legally or

[3] TODO: Coincide with release?

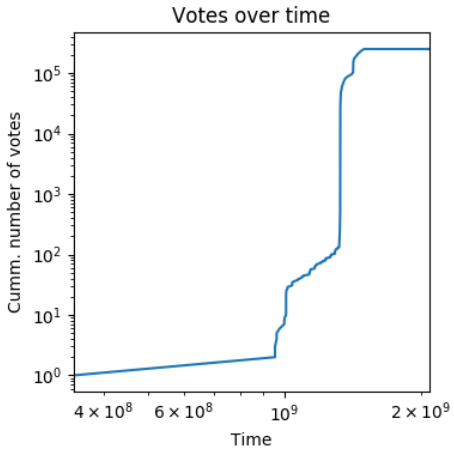

Figure 2: TODO: Add proper clustering

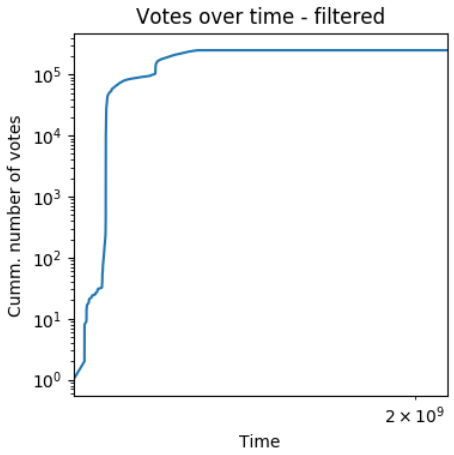

Figure 3: TODO: Add proper clustering

politically infeasible, an adversary might even try to shut down all services by launching a DDoS attack [10], conducted either in the open or anonymously.

Properly designed voting systems can alleviate some of the more serious weaknesses seen in most file-sharing systems. As seen in the Tribler case study, even voting systems that employ proven cryptographic methods are sometimes abused or circumvented. Another 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 this section we aim to give an overview of the measures adversaries have taken to prevent users from finding relevant content by spamming some part of the service. Machinations to prevent attack vectors from being successful can be broadly categorized in one of two types. The first one is to prevent or disincentivize an adversary from exhibiting the malicious behavior. The second type mitigates and/or isolates the effects of the malicious behavior such that benign users can make use of the service unhindered.

4.1 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 [16], 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. the routing table poi. 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 [11].

A way to deal with index- and routing table poisoning is to routinely determine 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. The authors of [5] propose a quorom-based approach, where TODO: {how to summarise quorom? Rewrite wrt standardised notation}.

4.2 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 [9]. 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 [9] 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. TODO: explain heuristic for identifying polluters

- TODO: Firewall
- TODO: NAT
- TODO: Prefix + prefix-merging

Credence [14] introduces the concept of object-based reputation as a way to estimate content authenticity. A simple voting protocol is used where a vote consists of $(\langle i, v \rangle_k, K$, where;

- i is the hashed identifier for a content item
- v is the value of the cast vote, with $v \in -1, +1$
- (K, k) represents a public-private key-pair of a specific user

In principle, an adversary can employ a Sybil attack [3] and generate multiple key-pairs (K, k) to quickly generate lots of misleading votes. A gatekeeper mechanism that disincentivizes joining the network multiple times in rapid succession would alleviate this issue, such as a cryptographic challenge.

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. 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, then a coefficient between peers A and B defined by:

$$\theta = (p - ab)/\sqrt{a(1-a)b(1-b)} \tag{1}$$

, where:

- a is the fraction of votes where A voted positively
- b is the fraction of votes where B voted positively
- p is the fraction of votes where both A and B voted positively
- θ is a correlation coefficient, with $\theta \in \big[-1, 1\big]$

When A and B tend to agree, that is, $\theta \geq 0.5$, a vote has a weight of θ. In all other cases, a vote gets a weight of 0.[4]

An alternative approach to Credence is Scrubber. Scrubber [2] allows for swift punishment of malicious users, while still allowing redemption. It operates on the assumption that at least 25% of users react to punishment by removing the polluted content from the network. TODO: why 25?.

4.3 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.

SpamResist [22] follows a scheme similar to Credence and Sorcery [19], albeit applied to tag clouds. By dividing peers in two groups, the *unfamiliar peers* and the *interacted peers*, two different heuristics with widely different

[4] TODO: {worst-case? (all +, or all -, no overlap???)

characteristics can be applied. SpamClean [21] is another way to combat tag spam [5]. SpamLimit [1]...[6]

4.4 Collusion/sybil 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 [15], [8]. TODO: {Describe WHAT collusion is, and HOW to detect it.}

SumUp [13] 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. TODO: How is this different from Pim's work?

TorrentTrust [12] 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 [14] clearly states that any different gate-keeping scheme can be used. TODO: {After finalizing Credence notation, extend this part with the changes to Credence to make TorrentTrust work}.

4.5 Insensitive users

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.

The authors of [7] challenge both of these assumptions in an empirical study. 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.

As a corollary, this also means that cooperative users often help polluters spread misinformation and spam in the network.

- [7] TODO: 20/80 rule
- TODO: Awareness

[5] TODO: What is 'the cosine technique'?
[6] TODO: Compare SpamLimit vs SpamResist vs SpamClean

This lack of active and informed user participation motivated the design choices in [18]. TODO: summary -> text.

5 Novel mitigation strategies

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. Having defined the plethora of ways in which an decentralized file-sharing system can be abuses and secured in Section 4.

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

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 [19] and [20], 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 ϑ connected peer in the network. [7]

6 Conclusions

TODO: Everything does not work. It totally should. TODO: Categorize:

- [6]
- [4]

References

[1] 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).

[7] TODO: describe and rewrite alg.

[2] 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.

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

[4] 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).

[5] 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).

[6] 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.

[7] 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).

[8] 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).

[9] 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).

[10] 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).

[11] 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.

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

[13] 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).

[14] 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).

[15] 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).

[16] 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).

[17] 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.

[18] 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).

[19] 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).

[20] 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.

[21] 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).

[22] 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).