

Green: Towards a Pollution-Free Peer-to-Peer Content Sharing Service

Ruichuan Chen
Peking University
chenrc@infosec.pku.edu.cn

Jon Crowcroft
University of Cambridge
jon.crowcroft@cl.cam.ac.uk

Eng Keong Lua
Carnegie Mellon University
englua@cmu.edu

Ennan Zhai
Peking University
enzhai@pku.edu.cn

Zhuhua Cai
Peking University
caizh@infosec.pku.edu.cn

Zhong Chen
Peking University
chen@infosec.pku.edu.cn

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems—*Distributed applications*

General Terms

Design, Security

Keywords

Peer-to-Peer Networks, Content Pollution, Social Reputation Model

1. OVERVIEW

Peer-to-Peer (P2P) content sharing systems have experienced an explosive growth, and now dominate large fractions of both the Internet users and traffic volume. However, due to the decentralized and unauthenticated nature, these systems are vulnerable to the content pollution attack [3], where attackers aggressively inject a large quantity of polluted content into the systems. Such polluted content could largely reduce the availability of the original authentic content, thus enormously shattering genuine users' confidence in the P2P content sharing systems.

In this paper, we propose *Green*, a pollution-free P2P content sharing system, which can effectively defend against content pollution by exploiting not only the traditional reputation model but also the inherent content-related information and the social networking technique.

2. DESIGN RATIONALE

Generally, in a decentralized P2P content sharing system, a participant plays one or several of the following roles: *file provider*, *file requester* and *index maintainer*. In *Green*, the transactions among these roles can be divided into four stages, i.e., *file publication*, *file query*, *file selection* and *feedback after downloading*.

In the stage of file publication, a file provider publishes the indices of her shared content to a group of index maintainers self-organized in a security overlay, so that a file requester can obtain the indices of her requested file from the associated index maintainers in the stage of file query. Here, we refer to a specific content (e.g., document, video or song) existing in the system as a *file*, and each file generally has a number of different *versions*. Afterwards, the file requester utilizes a *social reputation model* to filter out the polluted

versions, and then selects an authentic version for downloading. Finally, after the downloading, the file requester may give some explicit and/or implicit feedback on the downloaded version, and publish such feedback information to the security overlay.

File Publication. For providing the guarantees of security and availability, we create a structured P2P overlay to maintain the published indices of shared files. Without loss of generality, we utilize Chord [4] as the dedicated underlying P2P overlay. The index of a version V can be recorded in the following format:

$$\langle ID, Dig, PL, VL \rangle$$

Here, ID and Dig denote the identifier and the block digests of the version V , respectively; PL and VL denote the associated provider list and voter list, both of which may dynamically vary with the system running. Practically, considering the dynamic nature of P2P networks and the possibility of an index maintainer being malicious (or compromised), having only one index maintainer for each file should be unreliable. Therefore, we map a specific file onto m redundant index maintainers selected by hashing the filename via m different hash functions. Similarly, each participant can publish her voting history to m redundant *vote maintainers* organized in the same overlay.

File Query. In *Green*, when a file requester R searches for her requested file F , she can contact the redundant index maintainers according to the m specific hash functions, and then collect all the indices of the requested file. The index information is considered to be correct only if the majority of the associated index maintainers agree on it. By doing so, the abnormal indices fabricated by malicious index maintainers could be filtered out. Since each file's associated index maintainers are distributed in the security overlay uniformly at random, the probability that more than half of these associated maintainers are malicious is low as long as the network environment is Sybil-free. Even if there exist Sybil attacks, we could simply utilize the social networks existing in P2P content sharing systems to create a SybilLimit-style [6] "random routes" structure to mitigate Sybil attacks.

File Selection. From the above security overlay, the file requester R can obtain the indices of different versions of her requested file F . By leveraging the tuple VL in the index $\langle ID, Dig, PL, VL \rangle$, the file requester obtains the voting history of each voter by directly contacting the voter or the associated vote maintainers in the security overlay. Based

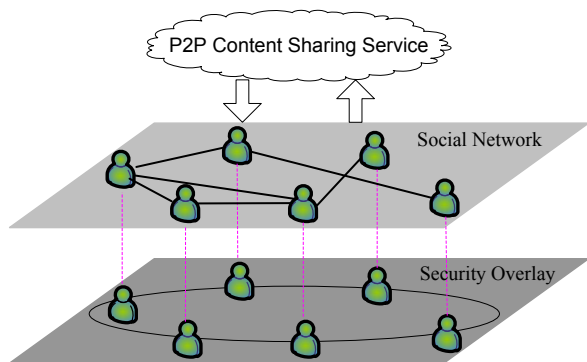


Figure 1: Green Architecture

on such voting histories, the requester R first aggregates all these voters' corresponding votes on a specific version V of her requested file, and then computes a Credence-style [5] correlation coefficient for each vote, and finally executes the weighted averaging to compute a reputation score $Rep_{R(V)}$ for the version V .

So far, the file requester R has obtained the provider list PL and the reputation score $Rep_{R(V)}$ for each version V of her requested file. In practice, many polluted versions of the requested file may not be effectively filtered out merely via the reputation model due to the lack of sufficient votes on those polluted versions; therefore, considering both the popularity and the reputation score of each version can better help identify an authentic version. Specifically, in Green, the probability of selecting a version V for downloading is proportional to the product of $(|PL| \times Rep'_{R(V)})$, where $|PL|$ denotes the length of the version V 's provider list PL , and $Rep'_{R(V)} \in [0, 1]$ is the normalized $Rep_{R(V)}$.

Social Enhancement. Currently, many P2P content sharing systems have already merged the friend information from social networks. A user and her friends generally share similar interests and give similar votes on a specific version; moreover, the friends are usually more trustworthy than other common users. To exploit the social information, Green provides two kinds of social enhancement for reputation model: *efficiency improvement* and *voting history extension*. Figure 1 illustrates the social network in our Green architecture.

• **Efficiency Improvement.** Following the above scenario, if the voter list VL includes many friends of the file requester R , then R does not need to resort to the normal reputation computation; as an alternative, she can utilize these friends' votes to efficiently evaluate the reputation of the target version V . Assume f' of R 's friends have given votes $\{v_1, \dots, v_{f'}\}$ on a version V , then R can directly compute the average of these friends' votes as the reputation score of V . Practically, some exceptions should be considered: firstly, if f' is small or there are significant differences among these f' friends' votes, then the computed reputation score may be unreliable, so we should return back to use the normal reputation model. Secondly, some friends may be malicious, so we should filter out such friends before executing the efficiency improvement; here, whether a friend is malicious can be evaluated by computing the correlation coefficient towards this friend in a Credence-like way [5].

• **Voting History Extension.** Current reputation models are generally vulnerable to the cold start and inactive

participant problems, where a file requester without or with little voting history cannot accurately compute a target version's reputation score. By considering the voting histories of social friends, the file requester can extend her local voting history, and then perform a more accurate reputation computation with the extended voting history. Assume that the requester R with local voting history VH_R has f friends, denoted by $\{F_i\}_{i=1}^f$; moreover, each friend F_i has the voting history VH_{F_i} . In our extension design, R computes the extended voting history VH'_R by aggregating her own voting history and her friends' voting histories weighted by an attenuation coefficient γ as follows:

$$VH'_R = avg(VH_R, VH_{F_1} \times \gamma, \dots, VH_{F_f} \times \gamma)$$

Here, each voting history is treated as a vector, and the *avg* is defined to be a function of computing the averages of *nonempty* values at each position in these vectors. When a user has not voted on a version, the corresponding vote is *empty*. Unfortunately, if R has only a few or even no friends, our social reputation model falls back to the basic form. In some sense, the requester R should "pay the price" for such an exception. Actually, R is also able to cope with this exception by removing malicious friends when their correlation coefficients are small, or by making more online friends via social interactions.

Feedback after Downloading. During the downloading, the file requester R downloads the data blocks of the selected version V from multiple providers in PL in parallel, and verifies the integrity of these blocks via a block-oriented probabilistic verification protocol as elaborated in [1, 2]. After the downloading, Green renders the requester R to give feedback to the system. If R *explicitly* casts a vote on the version V , Green publishes the vote to the security overlay by updating R 's local voting history and then inserting R into V 's associated voter list VL ; if R stores (or removes) V in her shared folder, Green *implicitly* inserts (or deletes) R in V 's associated provider list PL . As one of the four interdependent stages, user feedback plays an indispensable role in defending against the content pollution.

3. REFERENCES

- [1] Z. Cai, R. Chen, J. Feng, C. Tang, Z. Chen, and J. Hu. A holistic mechanism against file pollution in peer-to-peer networks. In *SAC*, pages 28–34, 2009.
- [2] R. Chen, E. K. Lua, J. Crowcroft, W. Guo, L. Tang, and Z. Chen. Securing peer-to-peer content sharing service from poisoning attacks. In *Peer-to-Peer Computing*, pages 22–29, 2008.
- [3] J. Liang, R. Kumar, Y. Xi, and K. W. Ross. Pollution in p2p file sharing systems. In *INFOCOM*, pages 1174–1185, 2005.
- [4] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.
- [5] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI*, 2006.
- [6] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, pages 3–17, 2008.