A Holistic Mechanism Against File Pollution in Peer-to-Peer Networks

Zhuhua Cai, Ruichuan Chen, Jianqiao Feng, Cong Tang, Zhong Chen, Jianbin Hu*
Institute of Software, School of Electronics Engineering and Computer Science, Peking University, China
Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, China
{caizh, chenrc, tangcong, chen, hjbin}@infosec.pku.edu.cn

ABSTRACT

Content pollution is pervasive in the current peer-to-peer file sharing systems. Many previous reputation models have been proposed to address this problem, however, such models strongly rely on the participants' feedback.

In this paper, we bring forward a new holistic mechanism which integrates the reputation model, inherent file-source-based information and the statistical data reflecting the diffusion state to defend against pollution attack. First, we deploy a redundancy mechanism to assure that the file requester receives the correct indices that accord with the information published by the file provider. Second, we complement the reputation information with the diffusion data to help the file requester select the authentic file for downloading. Finally, we introduce a block-oriented probabilistic verification protocol to help the file requester discern the polluted files during the downloading with a low cost.

We perform a simulation which shows that our holistic mechanism can perform very well and converge to a high accuracy rapidly, even in a highly malicious environment.

Categories and Subject Descriptors

C.2.0 [**Computer-communication Networks**]: General—*Security and protection*; C.2.4 [**Computer-communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Design, Security

Keywords

File Pollution, File Source, Reputation, Probabilistic Verification

1. INTRODUCTION

So far, Peer-to-Peer (P2P) file sharing systems have achieved a significant success in practice, dominating a large fraction of the Internet's resources and the participating users [8]. Some measurement studies showed that P2P traffic constituted 60% of the Internet traffic in tier-1 ISP [1]; more than 8 million users are connected to three popular P2P file sharing systems including FastTrack/KaZaA, eDonkey and eMule [10]. Although popular, due to the decentralized and unauthenticated nature, P2P file sharing systems have been targeted by pollution attacks which render the systems rife with corrupted and mislabeled files, e.g., even over 80% of the copies of popular files in KaZaA are polluted [9].

Many pollution defense mechanisms have been proposed, and most of them are based on reputation models [5, 7, 18]. These reputation-based mechanisms focus on the deduced information derived by the user collaboration, however they ignore some other kinds of helpful information. In this paper, we propose a new holistic pollution defense mechanism based on the Distributed Hash Table (DHT) overlay, which can quickly identify and isolate *polluted* files and *poisoned* indices [10], to help participants verify the integrity and authenticity of the requested files effectively and efficiently. Our main motivation lies on the fact that besides the reputation information based on the feedback of the participants, some inherent file-associated information provided by the file provider and some statistical data about the diffusion state of the requested files are ignored.

In common P2P file sharing systems, the process of transactions between the file *providers* and the file *requesters* can be divided into four stages. First, the file provider publishes the file to the corresponding *index nodes*. We employ the *redundancy mechanism* here to make sure that the index nodes could not effectively fabricate the information of his maintained indices. Second, the file requester searches for his requested file from the associated index nodes. With the former redundancy mechanism, we have the capacity of filtering out poisoned indices. Third, the requester selects a specific version of the file. However, due to the lack of authenticity verification, the file requester generally cannot know, without downloading, if the requested version has been polluted. Here, we can resort to the general reputation-based information and some extra inherent file-related information (e.g., the copy number of the requested version) to help identifying and isolating the potential polluted versions. Finally, the requester downloads the selected version from the associated file providers. During this stage, we make the requester verify if the version being downloaded has been polluted, without totally downloading the whole requested

*Corresponding author

SAC' 09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

version. To reduce the verification overhead, we further present a block-oriented probabilistic verification protocol to partially verify the requested version without influencing the integrity assurance of the verification significantly. We perform the performance evaluation with various different system behavior configurations. The simulation results demonstrate that our proposed mechanism performs very well and only very few downloads will end up with downloading the polluted versions even in a highly malicious environment.

The rest of the paper is organized as follows. Section 2 presents related work on the anti-pollution mechanism. Section 3 discusses some related terminologies and assumptions. In section 4 we cover the design of our holistic anti-pollution mechanism in detail. In section 5 we will describe the evaluation methodology and the main results. Finally, some conclusions and future work are presented in Section 6.

2. RELATED WORK

Along with the extensive deployment of P2P file sharing systems, security problems gradually become the bottleneck of their further development. To address the problems, some previous work has proposed many reputation models, which can be categorized into three classes, i.e, peer-based models, object-based models and hybrid models.

Among peer-based models, EigenTrust [7], Scrubber [4] and PeerTrust [19] are representative of them, in which each peer is assigned a reputation value reflecting the peer's honesty. However, many of such peer-based reputation models have a similar drawback, i.e., a peer with high reputation may also insert polluted files to the system and the peer's reputation can not accurately reflect the quality of such inserted files, which contradicts with the data-centric property of the P2P file sharing systems.

Credence [18] denotes one of the object-based reputation models, in which the file requester computes a reputation for the file to download referring to its authenticity. This computation is based on a distributed vote gathering protocol for disseminating the object reputations in the network, and a correlation scheme which gives more weight to votes from like-minded peers.

XRep [5] and Hybrid [3] are typical hybrid models which complement peer-based and object-based mechanisms. Nevertheless, due to the fact that most of participating users in P2P content sharing systems are rational in seeking to maximize their individual utilities, the reputation models are penalized by the lack of reliable user cooperation.

In [6], Habib *et al.* incorporated block-oriented probabilistic verification protocol and tree-based forward digest protocol to verify the data integrity in the P2P media streaming. Their design provides high assurance of data integrity and incurs lower computation overhead. Moreover, they used multiple hashes or forward error correction (FEC) [13] codes to improve these two protocols, rendering them to work well with unreliable transport protocols in media streaming systems.

Focusing on the replication of computation instead of data replication and comparison, the Repeat and Compare [12] utilizes the attestation records and sampled repeated execution to ensure the integrity of both static and dynamic content in untrusted P2P content distribution networks. When the replicas respond to the client, they insert attestation records to their response. Clients then forward a fraction of these records to the randomly selected verifiers for verification and such verifiers repeat the response generation and compare the results with received attestation records.

3. TERMINOLOGIES AND ASSUMPTIONS

So far, a general P2P file sharing system is composed by nodes and files published by such nodes. In a DHT-based overlay, nodes can play one or several of following roles, a file *provider*, a file *requester* and an *index node*. A file *provider* can publish files to the system, and he will notify the index node to build up the corresponding indices. When a query aimed for some specific files are raised by the file requester, the index node will respond with a list of related indices for the file requester to select. However, due to the lack of central authorities in systems, some peers may perform some malicious behaviors, e.g., a file provider may insert into the system a lot of polluted files, and the index node may fabricate the maintained indices.

In order to illustrate our design, we introduce some terminologies. In our design, we define *title* as the specific content in a P2P file sharing system, such as a song, movie, document and so forth. A specific title can have many *versions*. For music and movie, these different versions can be created by a large number of rippers/encoders, each of which can create a different version of the same title. Furthermore, when the metadata of the file (such as the *file name*, the file size, ID3 tags) is modified, additional file versions are created. For a popular title, there may exist thousands of different versions. Each version is assigned an identifier by the hash function on the version, including the metadata and content of the version together. Finally, a version may have many *copies* in that the version may be downloaded from each other in the system.

To provide the mechanisms of availability and scalability, the underlying structure of our security framework is a DHT-based overlay. We utilize Chord [17] as the dedicated underlying DHT-based overlay. We can also conveniently utilize another DHT-based overlay (e.g., CAN [14], Pastry [16] or Kademlia [11]) as an alternative. Some assumptions about the network properties are listed as follows: first, we suppose that the routing protocol is reliable and peer/file discovery is perfect. Second, we assume that the identifiers can be pseudo-randomly produced, e.g., the hash of the IP address, and then the peer can not select his identifier at his will. Finally, to assure that the majority of peers in the system are honest, we assume that sybil attack does not take place in our system, which can be partially solved by client cryptographic puzzles [15] or SybilGuard [20].

4. DESIGN RATIONALE

As described above, in four stages, three types of participants, including the file providers, the index nodes and the file requesters collaborate together to perform a transaction in a P2P file sharing system. As is shown in Figure 1, these four stages are publishing the version, searching for associated indices of a title, choosing a version of the title and downloading the version. After the download, the file requester may experience the version, and give feedback on the downloaded version to perfect our design. Four stages mentioned above will be discussed to illustrate our anti-pollution mechanism. We can show that our distributed and holistic anti-pollution mechanism can quickly identify and isolate

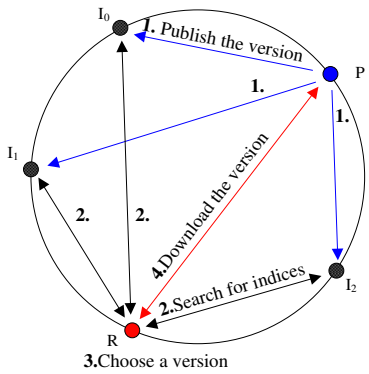


Figure 1: System illustration. Note that, each user practically plays one or several roles simultaneously

polluted files and *poisoned* indices, helping participants verify the integrity and authenticity of the requested files effectively and efficiently.

4.1 File Publication

The first basic procedure, which is illustrated in Figure 1, is the file publication initiated by the file provider P. Like in the original Chord [17], when the file provider P wants to publish a version of a file, he will choose a peer to maintain the index for the version. However, the peer selected may be malicious and refuse to insert the index for the version or just insert the fabricated index. To solve the problem, we employ a number of redundant nodes ($\{I_0, I_1, I_2\}$ in Figure 1) to make sure that any index node could not effectively poison the index. All these index nodes for the version can be chosen in the following way. Suppose the file name of the version is $filename$, the set of index nodes for the version is $I_v (I_v = \{I_0, I_1, I_2\}$ in Figure 1), then we apply the hash function on the suffixed file name as follows:

$$I_v = \{h_i | h_i = h(filename|i) \wedge i = 1, 2, \ldots, s\} \quad (1)$$

where h_i denotes the node that equals or follows the result of the hash function and s is the number of index nodes for the version. I_v keeps track of the version v, and any abnormal information stored in the set members can be discerned by the file requester. The index information is legitimate only if the majority of the index nodes in I_v agree on it. In this way, the index information received by the file requester R is really published by the file provider P with high probability, the analysis of which is left to the next subsection.

The index reord can be denoted in the following format.

$$\langle Id_v, Meta, Dig, VS, PS \rangle$$

where

- Id_v: the *id*entifier of the version.
- $Meta$:the *meta*data of the version, including the *file name*, the *file size* and the *file descriptors*.
- Dig: the *dig*est set of the version. Suppose a version can be divided into b blocks, then Dig has b elements, i.e., $\{d_i\}_{i=1}^{b}$, and Id_v can de deduced from Dig, e.g.,

$$Id_v = h(Meta|\{d_i\}_{i=1}^{b}) \quad (2)$$

With this equation, we can check if the digests of a version are real when the version is published for the first time.

- VS: the *voter set*, which is the set of peers who had given votes on the version.
- PS: the *provider set*, which is the set of the file providers of the version. Generally, PS may not be equal to VS for some peers giving votes on the version v may not provide the version and some file providers may not give votes on the version v.

Similarly, we can use the redundance mechanism to maintain the vote histories for a voter in the overlay. Each voter can broadcast his updated vote histories to a group of *vote maintainers*, who can be selected in the above way by hash function on the voter's identifier. In this way, the file requester R can get the vote history of the voter even when the voter is abruptly off line. Unlike the gossip protocol for Credence deployed in Gnutella, in our design any peer's vote history can be gotten at once based on the DHT overlay.

4.2 Index Search

When the file requester R searches for his requested file, he will get all the related indices from I_v. Any abnormal information from certain nodes in I_v will be discerned in the reason that the index information is considered to be right only if the majority of index nodes connected agree on it. On the other hand, if some attackers want to fabricate an index, they should compromise at least half of the members in I_v and all of these compromised index nodes should have the same information on the very index.

Suppose the size of aforementioned collusion group is m and the size of peers in system is n, then the probability of a pseudo-randomly chosen index node h_i belongs to the collusion group is $\frac{m}{n}$. As the size of the I_v is s, the probability that t out of s members in I_v belongs to the same collusion group is

$$P(s,t) = \binom{s}{t} (\frac{m}{n})^t (1 - \frac{m}{n})^{s-t} \quad (3)$$

then the probability that the file requester can get the right index from I_v is $\sum_{t=0}^{\lfloor \frac{s-1}{2} \rfloor} P(s,t)$. Suppose the ratio of total malicious peers in the system is β, then we can get that $\frac{m}{n} \leq \beta$. With the tradeoff between the communication overhead and degree of assurance, our design can provides adjustable levels of security. For example, if β is 0.2, s is 5, then the probability that the file requester can get the right index from I_v is 94.2%, which is sufficient for common systems. In the other words, with the redundancy mechanism, the requester R can get the right indices for his requested file with high probability.

Similarly, R can also get a voter's vote history directly from the voter or the vote maintainers. For the avoidance of fabrication, R can aggregate the information from such maintainers or directly from the voter.

4.3 Version Evaluation

Based on the above redundancy mechanism, the file requester R can get the real indices of all the versions with the same file. Due to the lack of authenticity verification, the file requester R generally can not know, before downloading,

if the listed versions have been polluted by the file provider. Here, we resort to the general reputation-based information and the copy number of the version to help identifying and isolating the potential polluted versions.

Unlike the traditional index record whose main function is to help locating the file, in our design, the index record $\langle Id_v, Meta, Dig, VS, PS\rangle$ of each *version* can provide us two key components, namely, the *voter set* VS and *file provider set* PS.

With VS, the file requester can know the voters and get their vote histories. Based on such vote histories, the file requester can compute the corresponding relation coefficient(denoted as θ) of each voter as the way in Credence [18]. Aggregating each voter's opinion, the file requester can compute a version v's reputation $Rep_{i(v)}$ as follows:

$$Rep_{i(v)} = \frac{\sum_{j\in VS_v} V_{j(v)}\theta_{(i,j)}}{\sum_{j\in VS_v} |\theta_{(i,j)}|} \tag{4}$$

where VS_v, $V_{j(v)}$ and $\theta_{(i,j)}$ denotes the voter set of version v, the vote value that peer j puts on version v, and the relation coefficient between peer i, and peer j. Also we must point out that $Rep_{i(v)} \in [-1, 1]$, and we only consider peers j when $\mid \theta_{(i,j)} \mid \geq 0.5$.

According to the file provider set PS, we can get the copy number of a version, which reflects the popularity of the version. Since a popular version with a large number of copies can be downloaded more easily and faster in the swarming downloads, many current P2P file sharing systems utilize the property and let a popular version more possible to be downloaded. Our design also embodies this idea, rendering the version with more providers a higher probability to be selected by the file requester.

Now we model our mechanism of the version evaluation as follows. For file request i, the probability of selecting a particular version $v \in V(t)$ is:

$$p_v(t) = \frac{Rep'_{i(v)} n_v(t)}{\sum_{u\in V(t)} Rep'_{i(u)} n_u(t)} \tag{5}$$

where $n_v(t)$ is the copy number of the version v with the file t. For $Rep_{i(v)} \in [-1, 1]$, in order to let $p_v(t) \geq 0$, we make a linear mapping and let $Rep'_{i(v)} = Rep_{i(v)} + 1$.

In practice, a popular file always has a large number of polluted versions, and this problem can not be solved by Credence [18] because insufficient votes have been put on such polluted versions. However, our design can work a good effect on such condition. Even if the polluters change their attack model and increase the copies of certain polluted version, our design also can filter out the polluted version for more number of honest users will experience the polluted version and give the negative votes on the version, leading the $Rep'_{i(v)}$ to be very low. In addition, since the number of malicious peers is limited and a peer generally can have one copy of certain version, without the unintentional help of the honest peers, it is difficult for malicious peers to produce a version with a large number of copies.

4.4 Downloading with Probabilistic Verification

After the three stages above, the requester begins to download the selected version from the associated file providers in the form of swarming downloads. However, the version may be polluted due to *identifier corruption* [2]. In order to reduce the cost of the file requester, we make the requester verify the authenticity of the version, without totally downloading the requested file. Note that, the index node provides the file requester the digests of each block, and with such digests of blocks, the file requester can verify each block when the block downloading is finished. When the file requester finds that a downloaded block is polluted, he just drops it and connect another file provider.

However, when the size of digests set Dig is very big, the overhead of communication and computation for the verification is high. To reduce the verification overhead, we further present a block-oriented probabilistic verification protocol to partly verify the requested file without influencing the integrity assurance of the verification significantly. Suppose a version has N blocks $\{b_1, b_2, \cdots, b_N\}$ and digests set Dig is $\{d_1, d_2, \cdots, d_N\}$, our probabilistic verification protocol runs as follows:

- *Step* 1: The file requester gets the real digest set Dig from the index nodes and checks it based on the equation (2).
- *Step* 2: The file requester connects each file provider and asks for some certain blocks, supposing the number of such blocks is k.
- *Step* 3: The corresponding file provider responds with the k blocks.
- *Step* 4: The file requester randomly chooses r out of the above k blocks and computes the digests of such blocks. If these computed digests match the digests in Dig, then he should accept such k blocks; otherwise, he rejects them.

The protocol can provide adjustable levels of security and reduce the computation overhead. When a malicious node tampers t out of k blocks and the file requester randomly chooses r blocks to verify, the probability for the attacker to succeed is

$$P(t, r, k) = \frac{\binom{k-t}{r}}{\binom{k}{r}} = \frac{(k-t)!(k-r)!}{k!(k-t-r)!} \tag{6}$$

The probability for the attacker to pollute the version drops obviously when the attacker wants to pollute more blocks. Generally, when 25% or more blocks are verified and 30% or more blocks are polluted, the probability to detect the pollution is very high.

After the download of the version, the file requester should give feedback to the system. If the downloaded version is polluted, the requester should take a vote on the version and delete the version to avoid the diffusion, otherwise he should give the positive vote on the version and publish the version. As one of four interdependent procedures, the collaboration among honest users plays an indispensable role to defend against pollution.

5. EVALUATION

This section describes our evaluation through simulation of our holistic mechanism (abbr., *Holistic*) and *Credence* [18], comparing them with a baseline system (abbr., *Baseline*) where each peer selects a version according to the popularity of the version. Our analysis is based on two main metrics, i.e., *accuracy* and *convergence*. The former describes

Table 1: Fixed System Configuration

Parameter	Value	
# total peers	1000	
# download sources	10	
# titles F	100	
# versions V	100	
downloads rate	4 files/day	
Shared Files(Start up)	Honest Peers	Polluters
# decoy insertion	20	400
# identifier corruption	20	50

the system's ability to reduce pollution and is measured by the fraction of daily unpolluted downloads. The latter relates to the time it takes to reach a steady state that the fraction of polluted downloads does not change significantly. In section 5.1, we will describe our simulation model in detail. Section 5.2 will present our main results.

5.1 Simulation Methodology

To evaluate the performance of Holistic in defending against pollution attack, we build an event-driven simulator of P2P file sharing system based on Chord [17]. The simulator has the following components:

Network Model: We follow the model of Chord file sharing network described in [17]. Moreover, swarming downloads and two redundancy mechanisms referring to the maintenance of the indices and the vote histories are deployed as described in section 4. Finally, our simulation has some assumptions, i.e., the query routing and peer/file discovery are reliable with the transfer time being ignored.

Object Model: There are F unique titles and each *title* has V unique versions with each version having a varying number of copies over time. At simulation startup, files shared by the peers are first selected by *title*, then its *version*. Both selections follow Zipf distributions with parameter $\alpha = 0.8$. Throughout our simulation, files to be downloaded are selected by first choosing the titles randomly and then choosing the version according to the mechanisms of Holistic, Credence and Baseline respectively.

Peer Model: There are two types of peers in our simulation, i.e., honest peers (fraction:70%) and malicious polluters (fraction:30%). At simulation startup, honest peers only hold unpolluted files with malicious peers only polluted ones. Throughout our simulation, the number of downloads per peer follows the uniform distribution. Both types of peers may download polluted files and unpolluted files, however, malicious peers may give a positive vote on a polluted file and a negative vote on an unpolluted file. For Holistic and Credence, we update the correlation θ between the file requester and all other peers immediately after each download to optimize our system. As our focus is on the pollution dissemination, we do not model the effects of intermittent network connectivity and churn.

Pollution Attack Model: With various system configurations and peers' malicious behavior patterns in a highly malicious environment discussed above, we simulate content pollution by following concrete attacks.

- *Decoy insertion*: a pollution mechanism made by inserting into the P2P system a polluted version of the file with the same metadata of the original file but a different identifier [9].

- *Identifier corruption*: a way of pollution made by generating a polluted file with the same identifier as the original unpolluted file, exploiting the weakness of current method of identifier generation or maliciously modifying the client software [2].

System Configurations: Some fixed parameters are listed in the Table 1, where we set the number of polluted versions per title smaller and the copies of corresponding polluted versions larger than usual to create a severer environment. The other parameters are described as follows:

- Pv: the probability that a peer gives a vote on the version after a download.

- Pd: the probability that a user deletes a polluted version immediately after a download.

- Pr: the probability that a user judges the quality of a version correctly.

- Pt: the probability that a malicious polluter pretends to be a honest peer and takes a correct feedback on the downloaded file.

5.2 Results

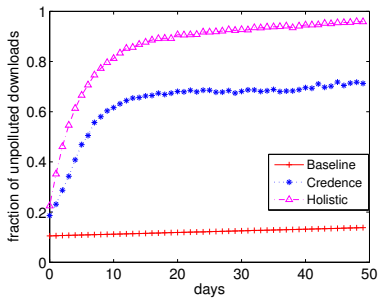
We perform experiments of P2P file sharing system with many configurations and obtain similar results. Without especially noted, we set $Pv = 1$, $Pd = 1$, $Pr = 0.9$ and $Pt = 0$. In order to eliminate the error brought by experiment deviation, each simulation result is an average of 5 runs.

Comparison. We present the comparison results by the daily fraction of unpolluted downloads in Figure 2. For decoy insertion, by severely isolating polluted versions, Holistic has a much better maximum accuracy than Credence and Baseline, though Holistic converges as fast as Credence. Even by the 30^{th} day, Holistic achieves a high ratio of success, maintaining roughly 90% correct classification; whereas Credence can only have an accuracy of 66%.

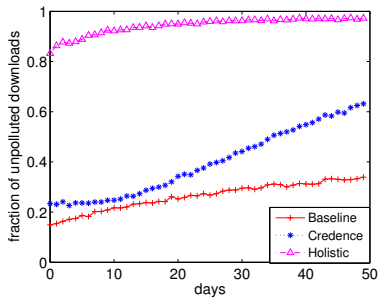
For identifier corruption [2], since Holistic can use our proposed probabilistic verification protocol to verify the authenticity of the version during and after the download, it guarantees that, from the 8^{th} day on, at least 90% of daily downloads are unpolluted. However, Credence and Baseline do not perform well since they cannot filter the polluted copies out of the selected version, and they take much longer to converge to the same accuracy as Holistic does.

Adaptability under Decoy Insertion. As shown in Figure 3, by changing the values of Pv, Pr, Pt. Holistic performs much better than Credence against decoy insertion. Figure 3a shows that, less cooperation among users would penalize the convergence speed of both systems. However, when $Pv = 0.2$, without distinctive loss of accuracy, Holistic achieves a high performance with a ratio of 80% of daily downloads are unpolluted since the 15^{th} day; whereas on the same condition, Credence only has an accuracy with 42%.

In Figure 3b, we find that Pr has a strong influence on the accuracy of Credence and Holistic that reducing Pr can result to the decline of both systems' accuracy. However, Credence is more sensitive to the change of Pr. When the value of Pr is changed from 0.9 to 0.8, the maximum accuracy for Credence is approximately reduced by 33%; whereas, the accuracy of our mechanism is only approximately reduced by 6%.

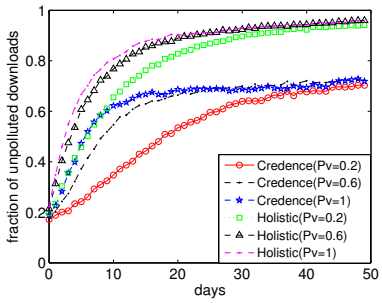


(a) Comparison under Decoy Insertion

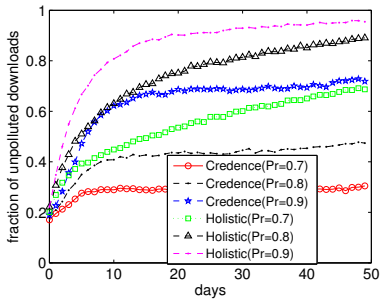


(b) Comparison under Identifier Corruption

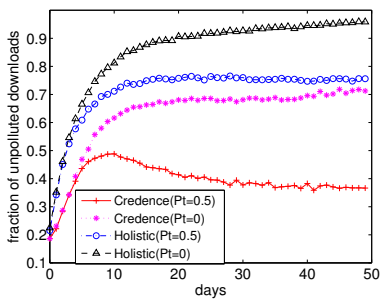
Figure 2: Effectiveness of Holistic



(a) Impact of voting(Pv)



(b) Impact of vote correctness(Pr)



(c) Impact of tricky attack(Pt)

Figure 3: Adaptability under Decoy Insertion

In Figure 3c, we evaluate the performance of Holistic and Credence against the tricky attack, in which polluters may pretend to be honest peers by giving correct votes on files, with the order to defraud the honest peers of credence. Then the malicious polluters may attack some specific versions by giving opposite votes. Figure 3c shows that, when confronted with decoy insertion, compared with condition without tricky attack, Credence works a lower accuracy, which is approximately reduced by 50%. In addition, Credence under this type of attack reaches its maximum accuracy in the 9^{th} days and then *declines*, and the reason for this is that honest peers can discern like-minded peers at the startup but gradually be deceived by the malicious peers launching tricky attack. Our mechanism is also influenced by this type of attack and its maximum accuracy is reduced by about 15%; however, the above decline does not occur in Holistic since that the statistical popularity information is utilized to counteract the tricky attack's influence.

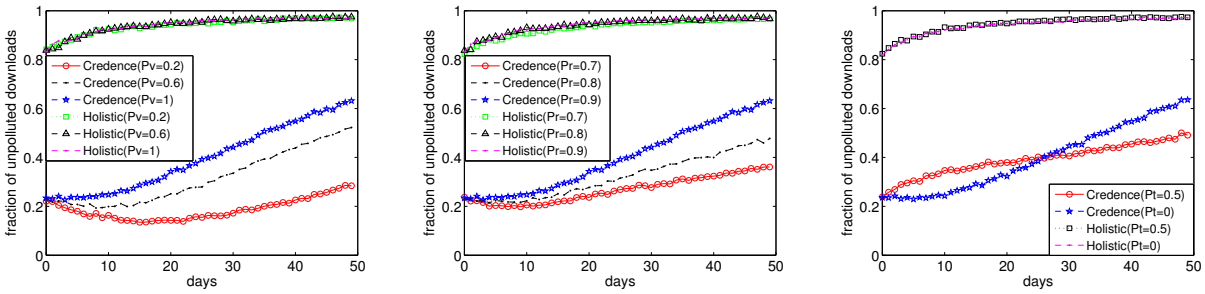
Adaptability under Identifier Corruption. By probabilistically verifying the authenticity of the blocks within a file, Holistic keeps a high accuracy and convergence speed when changing Pv, Pr and Pt as above.

Nevertheless, Credence is more sensitive to such changing parameters, though it can also reduce pollution dissemination in the long run. Figure 4a shows that reducing Pv can decelerate Credence's convergence a lot since the file requester cannot make a correct selection without sufficient feedback on the potential versions from the other peers. Especially at the simulation startup, when $Pv = 0.2$, the performance of Credence declines and the honest peers' discernment is tampered by the opposite votes taken by malicious peers. Without up to 20 days to effect the stabilization, Credence can not improve its performance.

Figure 4b shows that Pr has a strong impact on the convergence of Credence against identifier corruption. Reducing Pr can significantly slow down Credence's convergence speed when facing the identifier corruption. When Pr decreases from 0.9 to 0.7, the fraction of daily unpolluted downloads is reduced by approximately 48% in the 50^{th} day.

Figure 4c shows that for identifier corruption, with the deployment of probabilistic verification protocol, Holistic performs perfectly, and is not sensitive to malicious peers' tricky attack. On the contrary, Credence is susceptible to this type of attack. In the simulation startup, due to the help of the malicious peers' intentional hypocrisy, which partially impels the honest peers' discernment; however, in the long run, such intentional hypocrisy will impact the system's convergence speed, and the phenomenon can be seen from the 30^{th} days.

We also measure the impact of peers' deleting polluted files under both types of attacks. For Credence, whether honest peers delete the polluted versions does not influence the effects because peers in Credence only choose the version based on reputation but not employ the other helpful information(e.g., the diffused state of the version). As a result, whether honest peers delete the polluted files has no impact on the accuracy of the system. For Holistic, increasing Pd favors the system's convergence and accuracy to a



(a) Impact of voting(Pv) (b) Impact of vote correctness(Pr) (c) Impact of tricky attack(Pt)

Figure 4: Adaptability under identifier corruption

little extent. The more honest peers deleting the polluted files, the better effect Holistic works, though the impact is not very evident. Due to the space limit, we omit the two corresponding figures.

6. CONCLUSION AND FUTURE WORK

Existing P2P file sharing systems suffer from pollution attack. To address the problem, this paper presents a distributed and holistic mechanism which complements the deductive reputation-based information with file-associated information to filter out the pollution. We deploy our holistic mechanism in four stages involved in a transaction to optimize our system and help users discern the quality of files. In addition, simulation results demonstrate that our proposed mechanism performs well and only very few downloads will end up with polluted versions even in a highly malicious environment.

For future work, we plan to introduce some other existing advanced approaches to further improve the system performance and extend our holistic mechanism to many other overlays, including Gnutella, KaZaa, eDonkey and so forth.

7. REFERENCES

[1] Cachelogic: The picture of p2p file sharing, http://www.cachelogic.com/research.

[2] N. Christin, A. S. Weigend, and J. Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *ACM Conference on Electronic Commerce*, pages 68–77, 2005.

[3] C. P. Costa and J. M. Almeida. Reputation systems for fighting pollution in peer-to-peer file sharing systems. In *Peer-to-Peer Computing*, 2007.

[4] C. P. Costa, V. Soares, J. M. Almeida, and V. Almeida. Fighting pollution dissemination in peer-to-peer networks. In *SAC*, pages 1586–1590, 2007.

[5] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *ACM Conference on Computer and Communications Security*, 2002.

[6] A. Habib, D. Xu, M. Atallah, B. Bhargava, and J. Chuang. Verifying data integrity in peer-to-peer media streaming. In *MMCN*, pages 1–12, 2005.

[7] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, pages 640–651, 2003.

[8] R. Kumar, D. D. Yao, A. Bagchi, K. W. Ross, and D. Rubenstein. Fluid modeling of pollution proliferation in p2p networks. In *SIGMETRICS/Performance*, pages 335–346, 2006.

[9] J. Liang, R. Kumar, Y. Xi, and K. W. Ross. Pollution in p2p file sharing systems. In *INFOCOM*, 2005.

[10] J. Liang, N. Naoumov, and K. W. Ross. The index poisoning attack in p2p file sharing systems. In *INFOCOM*, 2006.

[11] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS*, pages 53–65, 2002.

[12] N. Michalakis, R. Soulé, and R. Grimm. Ensuring content integrity for untrusted peer-to-peer content distribution networks. In *NSDI*, 2007.

[13] J. M. Park, E. K. P. Chong, and H. J. Siegel. Efficient multicast packet authentication using signature amortization. In *IEEE Symposium on Security and Privacy*, pages 227–240, 2002.

[14] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[15] H. Rowaihy, W. Enck, P. McDaniel, and T. L. Porta. Limiting sybil attacks in structured p2p networks. In *INFOCOM*, pages 2596–2600, 2007.

[16] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[17] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1), 2003.

[18] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing (awarded best paper). In *NSDI*, 2006.

[19] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng.*, 16(7):843–857, 2004.

[20] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM*, pages 267–278, 2006.