TasteBuddy-based Version Selection Strategy for BitTorrent Users against Content Pollution

Lingyun Ruan
Department of Automation
Tsinghua University
Beijing, China
Email: rlyswf@gmail.com

An'an Luo, Zhen Chen
Department of Computer Science and Technology
Tsinghua University
Beijing, China
Email: laa@mails.tsinghua.edu.cn, zhenchen@tsinghua.edu.cn

***Abstract*—Content pollution problem has attracted broad attention due to its impacts on P2P networks' efficiency and availability. Especially for BitTorrent users, unmanageable versions of BT torrents and chunk-based file sharing mode make it more difficult to avoid pollution dissemination. In our paper, we propose a smart version selection strategy based on taste buddies to help users select high-quality versions and keep away from polluted ones. Performance evaluation based on real data shows that our approach effectively lowers the probability of selecting polluted versions compared with other strategies.**

***Keywords*-BitTorrent; P2P; content pollution; version selection;**

I. Introduction

Content pollution in P2P networks has attracted broad attention since a mass of corrupted or inauthentic files are spreading through P2P nowadays. A considerable number of those files may even carry viruses or Trojan horses. When distributed and dynamic peers enjoy the highly freedom and convenience, the loose structure of P2P networks also leads to weak supervision and inspection on shared contents. Thus ill-intentioned peers are free to disseminate corrupted, inauthentic and malicious files into their network, which results in pollution dissemination that not only contributes to worm and spam's widely spreading, but also significantly impacts on efficiency and availability of P2P networks [1].

As one of the most popular P2P file sharing systems, BitTorrent (BT) contributes to the major P2P traffic [2]. In BitTorrent, decoy insertion is a general content polluting approach, where attackers purposely inject polluted versions/copies into its network [3]. Unfortunately, as far as data integrity is concerned, BitTorrent is virtually pollution free [4].

When a user looks for target file via BT, he usually queries a popular torrent website (e.g. PirateBay, Mininova) for available versions. But attackers may premeditatedly upload files with cheating description. Effective strategy is not available for users to avoid polluted versions, even references or comments are given. Moreover, once users choose polluted torrent file, chunk-based file sharing mode makes it difficult to detect pollution during downloading process, as we cannot distinguish corrupted chunks from uncorrupted ones, unless we get the whole chunks.

In our paper, we proposed TasteBuddy-based Version Selection Strategy (VSS) to help user choose high-quality versions and avoid polluted ones before joining swarm. Taste buddies are a group of peers who are congenial to the user's interests and share similarity on what they like to download from BT. Using information provided by taste buddies, we could reduce version selection misjudgment. Through real-data experiment, TasteBuddy-based VSS proves to be an effectual strategy - it considerably minimizes the probability of selecting polluted versions.

The rest of this paper is organized as follows. Section II presents related work and problem analysis. Section III introduces Tastebuddy-based version selection strategy. Section IV gives performance evaluation based on real data, and Section V shows conclusion.

II. Related Work and Problem Analysis

A. Related Work

Pollution attackers inject a mass of tampered or inauthentic contents into P2P networks in order to entice unsuspecting users to download the files and share those polluted copies with more users. In this manner, polluted copies with spam or virus spread rapidly and widely through the networks and hence attackers would easily achieve malicious goals [5][6]. Moreover, peer dynamics of P2P network allows attackers to cooperate with their accomplice to launch a large-scale pollution attack by building a botnet, which results in significant impacts on P2P traffic.

Researchers have proposed a number of P2P user models to investigate the general pollution dynamics in P2P systems [7][8]. However, recent experimental studies show that the pollution level in the existing P2P network is significantly larger than what these models predicted [5][6].

Some schemes against P2P pollution have been proposed. In the early stage, file matching and user filtering are two common approaches which enable user to detect pollution content after files are downloaded [6]. However, checking

the content of so many files manually after downloading is not really a viable option [9]. It is also impossible to maintain a trustable and complete matching database for all the resources in BitTorrent, because users can easily create and upload different versions of files. Hence user filtering is a limited-effective scheme which depends on users' incentive to filter out polluted files in time.

Afterwards, methods of detecting pollution before downloading are proposed, most of which are based on establishing trust or reputation system either for peers or for objects. Kevin Walsh proposed Credence which addresses content pollution by providing reliable estimates of object authenticity [10]. Scrubber imposes severe and quick punishment to content polluters by taking individual experience and peer testimonial into account [9]. PeerTrust [11] built a reputation-based trust system for peer-to-peer electronic communities. However, current reputation systems may suffer from low robustness against collusion and are complex to implement [4]. Meanwhile, cheating behaviors, sibling attacks and white-washing add the difficulty to detect polluted content and to identify malicious peers who are responsible for injecting polluted files into the swarm [12]. Besides these, the system with moderators, who are trustable contents providers, seems very effective in removing fake and corrupted files. However, such mechanisms rely on a small number of moderators acting as submitters that inject numerous daily contents into network, which depends on global components and is extremely difficult to distribute [4].

Though there are a number of studies about building decentralized trust or reputation system (i.e. Credence [10], Scrubber [9], PeerTrust [11]) to help users discover and select good resources, most of them are designed for Kazaa and Gnutella, not for BitTorrent. In BitTorrent, it is difficult for users to make an exact choice, since any single selection strategy cannot guarantee that users pick out exactly what they want. Moreover, chunk-based file sharing mode further makes it difficult to detect file pollution while downloading, which brings slow response and limits effect for current schemes. So it is critical to build an early warning mechanism in pollution detection for such peers inside the swarm.

Some works related to P2P but not directly correlative to content pollution problem are also enlightening. J.A. Pouwelse built reputation mechanisms based on taste buddies for Personal Video Recorder [13] and P2P sharing system (Tribler) [14] to accelerate data sharing speed and resource discovery, but not for anti-pollution.

B. Problem Analysis

In most of today's P2P file sharing systems, how to choose high quality version is an open question. Each user may have its own Version Selection Strategy (VSS). The existing VSSs can be reduced to three categories and any single one is limited in resisting pollution attack.

1) FileAttribute-based VSS: In most cases, users regard versions as authentic just according to explicit file attributes, such as file name, file size and file description. In this manner, the probability of choosing a version is only related to users' subjective judgment. Without loss of generality, after excluding unsuited files subjectively, the probability of choosing a polluted version is directly proportional to the number of the polluted versions remaining after exclusion, and is inversely proportional to the total number of remaining versions. Let V_T denote the set of select versions related to the topic T. $Polluted\_V_T$ denotes the set of polluted versions with topic T. So the probability for user choosing a polluted version on the topic T is:

$$Prob_T(FA) = \frac{|Polluted\_V_T|}{|V_T|} \tag{1}$$

Under this strategy, pollution attackers can easily raise the probability $Prob_T(FA)$ by randomly injecting more decoy or polluted versions with fraudulent description.

2) SeedNumber-based VSS: In this case, users concern more about downloading speed - they would like to choose the available version with the largest number of seeds, which means more source nodes in the swarm. Some sites provide to users seed number as reference. Dismissing extreme cases, we assume that users randomly select a version from the set V_{S_T} whose elements have more seeds than a threshold S^*. Let $S(v)$ denotes the seed number of certain version v. Thus the probability for a user to choose polluted version is:

$$V_{S_T} = \{v | S(v) \geq S^*, v \in V_T\}$$

$$Polluted\_V_{S_T} = \{v | v \in V_{S_T} \cap Polluted\_V_T\}$$

$$Prob_T(SN) = \frac{|Polluted\_V_{S_T}|}{|V_{S_T}|} \tag{2}$$

With this strategy, pollution attackers can easily raise $Prob_T(SN)$ by injecting a large number of polluted seeds about a single version in order to attract innocent users.

3) Reputation-based VSS: In P2P systems, many users select version through referring to the assessments provided by other users. In reputation-based scheme, higher reputation indicates better quality and higher authenticity. There are both centralized reputation system (i.e. Amazon, eBay) and decentralized reputation system (i.e. Credence) allowing users make their own assessments to resources. For quantitative analysis, we assume that users will randomly select a version among the set of V_{R_T} whose element has higher reputation value than a threshold R^*. Let $R(v)$ denotes average reputation value of a certain version v. So we have:

$$V_{R_T} = \{v | R(v) \geq R^*, v \in V_T\}$$

$$Polluted\_V_{R_T} = \{v | v \in V_{R_T} \cap Polluted\_V_T\}$$

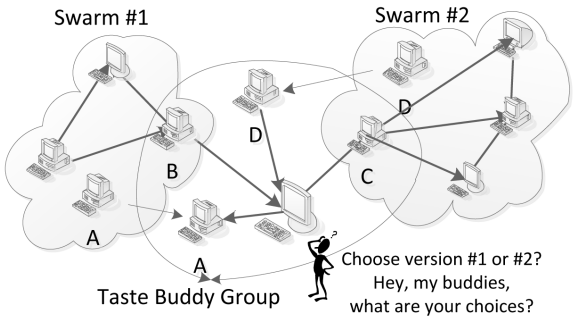

Figure 1. Main idea of TasteBuddy-based Version Selection Strategy

$$Prob_T(REP) = \frac{|Polluted\_V_{R_T}|}{|V_{R_T}|} \quad (3)$$

Generally, polluted versions bear lower reputation after a number of users' examination. However, unfortunately, reputation assessments are not effective to deal with pollution attackers, since they can raise reputation of the polluted versions and smear authentic versions by giving opposite assessments, just like bad mouth attacks.

III. TASTEBUDDY-BASED VERSION SELECTION STRATEGY

According to the aforesaid analysis, among three current version selection strategies, any single one has limited effects in resisting pollution attacks for BitTorrent users. Hence it is necessary to provide a more practical and effective VSS. An intuitive idea is to simply integrate three current strategies to select authentic version. However, it is still not sufficient.

As shown in Fig. 1, based on taste buddies and three current strategies, we propose our hybrid strategy. In order to reduce probability of bad selection, we add a role like trial jury, which is called taste buddies, to help user make the best choice. Taste buddies are a group of peers who are congenial to the user's interests about what they like to download from BitTorrent network. Each peer in BT network maintains a preference list, which contains a certain amount of high quality torrent file versions that they are strongly willing to recommend to other users. Notice that preference list is not merely a recent download list, as a torrent file can be added into it only after its authenticity and quality is approved and a high evaluation is given. Each peer updates its taste buddy list according to the preference lists' similarity with its buddies' preference lists.

A. Version Selection Algorithm

Our TasteBuddy-based version selection algorithm contains two steps: CandidateList Generation and Version Selection.

First, *candidateList* is a list of strongly recommended versions generated by intersection of version top-L lists sorted by different strategies (i.e. *FileAttribute* , *SeedNumber*, and *Reputation*). As BitTorrent users may have version selection strategy of their own favor, we bring in weight vector $W =< \omega_1, \omega_2, \omega_3... >, \omega_i \in [0, 1]$ to reflect the influence degree of each strategy. Then we obtain *candidateList* by intercepting a certain number of leading elements from an overall top-L ranking list, which is sorted by weighted average rank with corresponding W, as shown in Algorithm 1. W offers local user flexibility for personal customization. L denotes the length of ranking list, which must balance between information abundance and maintenance cost, and it is related to the total number of versions user received.

Algorithm 1 $CandidateListGen(versions)$

```
// weight vector of each strategy, set by BT users
W =< w1, w2, w3 >;
// versions sorting by torrent website
list1 = top L versions sorted by file attribute;
list2 = top L versions sorted by seed number;
list3 = top L versions sorted by reputation;
for each version in list1 or list2 or list3 do
   // rank number is the same as index(0 L − 1)
   // if the version is not in the list_n return L
   r1 = rank number of version in list1;
   r2 = rank number of version in list2;
   r3 = rank number of version in list3;
   weight = ((L−r1)*w1+(L−r2)*w2+(L−r3)*w3) / (w1+w2+w3);
end for
candidateList= top L version sorted by each version's weight;
return candidateList
```

Second, as shown in Algorithm 2, hash value of each torrent in *candidateList* will be sent to the peers in *tasteBuddyList* to ask for recommendation. Each taste buddy will check if a candidate version exists in its preference list with the same hash value, and then return a version index, after receiving which the corresponding candidate version's $popularity$ will increase by one. Finally the user will select a version with highest $popularity$ as the target torrent. If *selectedVersion* is null, the user will randomly select one version from the *candidateList*.

Algorithm 3 shows how to create and update *tasteBuddyList*. Each peer maintains a *preferenceList*, which records a certain amount of torrents this user prefers and recommends recently. And a *tasteBuddyList* is created and sorted according to the similarity degree of each buddy peers' *preferenceList*. Function *CalSimilarity()* calculates similarity degree of two *preferenceList* by comparing torrents' hash value. In BitTorrent, once joining in a swarm, each peer keeps TCP connections with certain amount of other peers called connectable peers [5] , so the maximum number of *tasteBuddyList* (M) is the same as maximum TCP con-

Algorithm 2 $VersionSelection(versions, tasteBuddyList)$

$candidateList = CandidateListGen(versions)$;
for each $peer$ in $tasteBUddyList$ **do**
  if $peer$ is connectable **then**
    send to $peer$ each candidate torrent's hash value;
    $index$ = torrent index from $peer$'s feedback;
    if $index \geq 0$ **then**
      $candidateList[index].popularity++$;
    end if
  end if
end for
$selectVersion$ = version with the highest popularity in the $candidateList$;
if $selectedVersion == null$ **then**
  $selectedVersion$ = randomly select form $candidateList$;
end if
return $selectedVersion$

Algorithm 3 $BuddyListUpdate(connectablePeers, preferenceList)$

for each $peer$ in $tasteBuddyList$ **do**
  if $peer$ is not connectable for a period of time **then**
    delete $peer$ from $tasteBuddyList$;
  end if
  if $peer$ made a wrong recommendation **then**
    delete $peer$ from $tasteBuddyList$;
  end if
end for
randomly select M peers from $connectablePeers$;
for each $peer$ **do**
  if $peer$ is not in $tasteBuddyList$ **then**
    send hash value of each torrent in $preferenceList$ to $peer$;
    $tempList$ = torrent hash value from $peer$'s $preferenceList$;
    $peer.similarity = CalSimilarity(preferenceList, tempList)$;
    if $tasteBuddyList.length \geq M$ **then**
      $lastPeer = tasteBuddyList$'s last peer;
      if $peer.similarity \geq lastPeer.similarity$ **then**
        delete $lastPeer$ from $tasteBuddyList$;
        insert $peer$ into $tasteBuddyList$ by ranking of similarity;
      else
        insert $peer$ into $tasteBUddyList$ by ranking of similarity;
      end if
    end if
  end if
end for
return $tasteBuddyList$

nections (typically 40 peers). The *tasteBuddyList* will be updated by periodically exchanging *preferenceList* between peers, sorting *tasteBuddyList* with refreshed similarity degree and displacing old buddy with new one, the time period is set as peer connection timeout in BT (typically 300 sec). Buddy peers will also be removed if they give wrong recommendation or have been not connectable for a long period of time.

B. Algorithm Discussion

According to Eq. 1- 3, for our hybrid strategy, the probability of choosing polluted version is $Prob_T(FA, SN, REP)$. Assume

$$Prob_T(FA) = p_1, Prob_T(SN) = p_2, Prob_T(REP) = p_3$$

It is obvious that

$$Prob_T(FA, SN, REP) \leq min\{p_1, p_2, p_3\}$$

Especially, when $Prob_T(FA)$, $Prob_T(SN)$, $Prob_T(REP)$ are irrelative, we will have

$$Prob_T(FA, SN, REP) = p_1 * p_2 * p_3$$

The equation above means the pollution-selection probability of hybrid strategy is surely much lower than adopting any single one of the three strategies. However, a smart attacker could control a botnet and create a torrent with high swarm population and high reputation, which looks rather attractive, especially on some hot topics. Therefore, we bring in taste buddies to give valuable recommendations, which can further protect user from making wrong choices and joining polluted swarm.

Someone may argue about the effectiveness of taste buddies, because we can't guarantee the existence of the same torrents in taste buddies' *preferenceList* since the sources' topics are so diverse. According to Algorithm 3, we define max length of *preferenceList* as P, and P_i is the number of same torrents appearing in *preferenceList* of both local peer and i^{th} taste buddy. The similar degree, which reflects the similarity of users' recent hobbies and interests, is hence defined as P_i/P. So the probability of a user's target torrent also appearing in any taste buddy's *preferenceList* is: $1 - \sum_{i=1}^{M}(1 - P_i/P)$. Though P_i/P might be very small, since M is much larger, it is very likely to find someone who has downloaded the version you are interested in and give you their suggestion.

Another question is on the correctness of recommendations from taste buddies concerning there are malicious peers in *tasteBuddyList*. For BitTorrent, as connectable peers are randomly selected by trackers or DHT, it is hard for attackers to be peer of a certain target. Even so, actually pollution attacker is selected to be a taste buddy only under condition of its high similarity degree with good peers (P_i/P), which means attackers have to share at least part of a victim's interests. It limits the opportunity of malicious

peers providing misleading recommendations. And once a taste buddy is detected in making a wrong recommendation, no matter it is malicious peers or good peers, it will be deleted from *tasteBuddyList*.

The time complexity of our VSS algorithm is very low, because the major cost of list sorting, which evaluates a version through file attribute, seed number and reputation into consideration, can be done on BT server side uniformly and then returned to peers. The communication complexity of Algorithm 1&2 is $O(LM)$, in which L and M is the length of *candidateList* and *tasteBuddyList* respectively, while Algorithm 3 is $O(MP)$, among which P denotes *preferenceList's* length. Since L and M are both small, the traffic overhead of peer communication is pretty low. For $TasteBuddyUpdate$ algorithm, the size of $connectablePeers$ and $preferenceList$ are fixed number (M and P), which means each peer will send $M \times P \times sizeof$(hash value of torrent) traffic every hundreds of seconds.

IV. Performance Evaluation

In order to verify the performance of our hybrid TasteBuddy-based VSS, we collect real data from famous BT websites to make a contrast between our algorithm and other strategies. We search hot topics (Movie, Music, and Software) on each websites and collect all torrents' information as a dataset for strategy verification. All the fake, inauthentic, irrelevant, incongruous torrents are pre-identified and counted based on torrent information. And we define Pollution Rate as the proportion of polluted torrents in the whole torrents collected, which equals the probability of a polluted version in a completely random select.

Torrent dataset with various keywords are presented in Table I is collected from three hot BT websites (www.mininova.org, www.torrentreactor.net, and torrentfinder.com). Take the movie *Transformers 2, 2009* for example; we acquired 385 different torrents, among which 263 versions are polluted. The reason of such a high pollution rate is that *Transformers 2* is newly released. Then we apply five different version selection strategies (completely randomly select, file-attribute strategy, seed-number strategy, reputation strategy and TasteBuddy-based strategy) to avoid polluted versions. We compare their effectiveness by calculating the probability of selecting polluted versions is calculated for each VSS. To simulate our TasteBuddy-based strategy on the dataset, we assume torrent files are equally distributed among the peers, each peer keeps 10-torrent preference list and 40 taste buddies, and that weight vector W is $[1/3, 1/3, 1/3]$ which means equal weight of three existing strategies.

Finally, by analyzing simulation result of five VSSs based on our dataset, as shown in Fig. 2, several conclusions can be obtained.

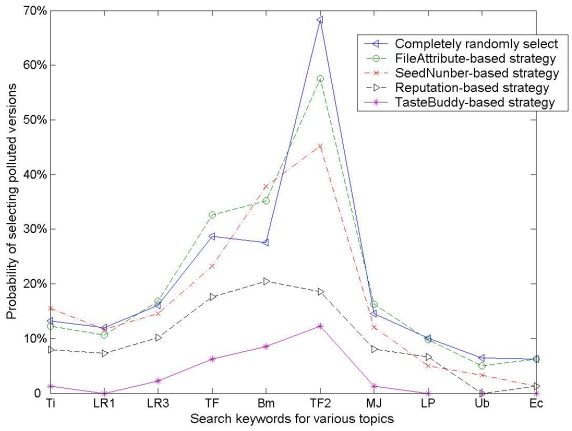

Figure 2. Comparison on probability of selecting polluted versions under various strategies.

First, pollution attackers prefer to attack videos, especially hot movies, rather than music and software. Pollution rate of the newly released movie is much higher, as its vast demand offers a great opportunity for pollution dissemination.

Second, FileAttribute-based strategy seems not effectual in reducing probability of selecting polluted versions, sometimes even worst than random selection, because pollution attackers are good at forging torrents and making them look so attractive. And SeedNumber-based strategy is also not good, especially for hot resources. Polluted versions of the latest movies can gather a great number of users in a very short time, which increases its selected probability even under the SeedNumber-based strategy. Reputation-based strategy works better than other two strategies, since the more peers join in, the more correct evaluation is given to user. However, this kind of central reputation system is maintained by web servers, which is not scalable and robust. The simulation results of the three common used strategies can also be used as reference when setting weight vector W.

Third, compared with other strategies, our hybrid TasteBuddy-based VSS is the most effective way to avoid polluted torrents. In some cases, the pollution selection probability can be limited to 0. Furthermore, it is found that SeedNumber-based VSS and TasteBuddy-based VSS are closely-related, which is reasonable in that pollution proportion of seeds has influence on the effectiveness of taste buddy group to a certain extent.

V. Conclusion

Pollution dissemination seriously impacts the availability of P2P applications. Most of previous work focused on building a centralized database for reputation system. But these approaches are limited to detect pollution in time and to restrict pollution level effectively in BT network, because BT's user-decided version selection mode and chunk-based

Table I
TORRENT SEARCH RESULTS OF CERTAIN KEYWORDS

Keywords	Category	Total Torrents	Polluted Torrents	Pollution Rate
Titanic, 1997 (Ti)	Movie	219	29	13.24%
The Lord of the Ring 1, 2001 (LR1)	Movie	158	19	12.03%
The Load of the Ring 3, 2003 (LR3)	Movie	144	23	15.97%
Transformers, 2007 (TF)	Movie	216	62	28.70%
Batman: The Dark Knight, 2008 (Bm)	Movie	203	56	27.59%
Transformers 2, 2009 (TF2)	Movie	385	263	68.31%
Dangerous, Michael Jackson (MJ)	Music	62	9	14.52%
Minutes To Midnight, Linkin Park (LP)	Music	178	18	10.11%
Ubuntu Linux 9.04 desktop (Ub)	Software	62	4	6.45%
Eclipse SDK 3.3 (Ec)	Software	44	3	6.82%

file sharing mode inhibits powerful official inspection and instant feedback from users.

In our paper, we proposed a smart version selection strategy based on taste buddies to help users choose high-quality versions and avoid polluted ones. Performance evaluation based on real data shows that our approach effectively decreases the probability of selecting polluted versions compared with other strategies.

Our future work is to design an effective scheme under which pollution warnings can be spread rapidly to all the peers in a swarm-to-swarm manner. We will also collect more trace data of pollution swarms to build a more precise model for BitTorrent contents pollution.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China (NSFC) No. 90718040 and No.60773138, the National Grand Fundamental Research Program of China (973) under Grant No. 2006CB303000 and 2010CB328105, the National High-Tech Research and Development Plan of China (863) under Grant No. 2007AA01Z468 and No.2008AA01Z212.

REFERENCES

[1] G. Suryanarayana, J. R. Erenkrantz, and R. N. Taylor, "An architectural approach for decentralized trust management," *IEEE Internet Computing*, vol. 9, pp. 16–23, 2005.

[2] T. Mennecke, "Bittorrent remains powerhouse network," *Slyck News*, January 31 2005.

[3] K. Walsh and E. G. Sirer, "Fighting peer-to-peer spam and decoys with object reputation," in *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, ser. P2PECON '05. New York, NY, USA: ACM, 2005, pp. 138–143.

[4] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Peer-to-Peer Systems IV*, ser. Lecture Notes in Computer Science, M. Castro and R. van Renesse, Eds. Springer Berlin / Heidelberg, 2005, vol. 3640, pp. 205–216.

[5] U. Lee, M. Choi, J. Cho, M. Y. Sanadidi, and M. Gerla, "Understanding pollution dynamics in p2p file sharing," in *In Proceedings of the 5th International Workshop on Peer-toPeer Systems (IPTPS'06)*, 2006.

[6] J. Liang, R. Kumar, Y. Xi, and K. Ross, "Pollution in p2p file sharing systems," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, 2005, pp. 1174 – 1185 vol. 2.

[7] N. Christin, A. S. Weigend, and J. Chuang, "Content availability, pollution and poisoning in file sharing peer-to-peer networks," in *Proceedings of the 6th ACM conference on Electronic commerce*, ser. EC '05. New York, NY, USA: ACM, 2005, pp. 68–77.

[8] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel, "Denial-of-service resilience in peer-to-peer file sharing systems," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '05. New York, NY, USA: ACM, 2005, pp. 38–49.

[9] C. Costa, V. Soares, J. Almeida, and V. Almeida, "Fighting pollution dissemination in peer-to-peer networks," in *Proceedings of the 2007 ACM symposium on Applied computing*, ser. SAC '07. New York, NY, USA: ACM, 2007, pp. 1586–1590.

[10] K. Walsh and E. G. Sirer, "Fighting peer-to-peer spam and decoys with object reputation," in *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, ser. P2PECON '05. New York, NY, USA: ACM, 2005, pp. 138–143.

[11] L. Xiong and L. Liu, "Peertrust: supporting reputation-based trust for peer-to-peer electronic communities," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 7, pp. 843 – 857, 2004.

[12] H. Chen and G. Chen, "A resource-based reputation rating mechanism for peer-to-peer networks," in *Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on*, 2007, pp. 535 –541.

[13] J. Pouwelse, M. V. Slobbe, J. Wang, M. J. T. Reinders, and H. Sips, "P2p-based pvr recommendation using friends, taste buddies and superpeers," in *in Beyond 2005: A Workshop on the Next Stage of Recommender Systems Research*, 2005.

[14] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. V. Steen, and H. J. Sips, "Tribler: A social-based peer-to-peer system," in *In The 5th International Workshop on Peer-to-Peer Systems (IPTPS06)*, 2006.