Making P2P Downloading Dependable by Exploiting Social Networks

Yingwu Zhu
Department of CSSE, Seattle University
Seattle, WA 98122, USA
Email: zhuy@seattleu.edu

***Abstract*—P2P file sharing networks are rife with unusable copies of files, polluted either incidentally or deliberately. Pollution has made P2P downloading undependable — users may end up receiving polluted copies of a target file. To combat pollution, we propose a recommender that maintains users' opinions on previously downloaded files over their social network and provides recommendations on demand to assist users in making file download decisions. The big challenge for the recommender is to produce right recommendations in face of various potential attacks. To this end, we propose novel techniques in managing user opinions and forming recommendations. Our experimental results show that our recommender greatly outperforms existing download selection strategies in identifying authentic files.**

Keywords: Pollution, peer-to-peer, recommender, social links.

I. Introduction

In past years, P2P file sharing networks have been applauded as a "happy land" for a variety of reasons, such as open environment without admission control, fast speed download, and free downloads without monetary charge. However, recent studies [1], [2] have warned users that P2P networks are not a "pure land" by revealing significant file pollution. File pollution comes from different sources. Inadvertent users often accidentally create damaged files and upload them into P2P network as authentic copies. Worse, attackers and professional companies like Overpeer [3] intentionally deposit a massive amount of *decoys* (e.g., files tampered with, spam files or virus) into P2P networks for different reasons, resulting in a significant level of pollution. In order to fool users to download decoys, each decoy is labeled as an authentic copy by associating it with the metadata of the authentic file (a wolf in sheepskin).

Pollution raises several issues. First, pollution makes P2P downloading undependable — a peer lacking measures to distinguish decoys from authentic files, may be fooled to download decoys, wasting network and client resources. Second, being unaware that what it downloaded is a decoy, a careless peer may help the decoy originator spread the decoy, further reducing availability of authentic files [4]. Lastly, pollution adversely impacts user experience.

Existing approaches to counteracting pollution fall into three broad categories. First, one can base download decisions on popularity of replicas in query responses [2] or random selection of query responses [5]. Second, one can use reputation systems to make informed download decisions [6]. Third, one can rely on a trusted database, which maintains content hashes of authentic objects, to make download decisions. Each of these approaches has certain advantages and disadvantages, which we discuss in Section II.

Our vision on P2P file sharing networks is that file pollution is inevitable due to open environment. In this paper, we have no ambition to make P2P networks a "pure land". Instead, we aim to design a recommender for users to make right download decisions. The recommender is built on top of the social network of participating users. Participating users advertise their opinions on previously downloaded files to the recommender over their social links. When a user issues a query for a file in the P2P network, the user also submits the query to the recommender via her social links. Upon receiving query responses from the P2P network, the user uses the recommendation (i.e., content hash of the authentic file) returned from the recommender to make informed download decisions. Via simulations, we show that the recommender is effective in defending against pollution in P2P networks and greatly outperforms existing heuristic anti-pollution strategies including random- and popularity-based selections.

The remainder of the paper is structured as follows. Section II provides overview of related work and background on random routes. We detail design of the recommender in Section III. Section IV describes experimental setup and provides experimental results. We conclude the paper in Section V.

II. Related Work

Pollution in P2P networks has been investigated by recent studies [1], [2], [4]. Existing anti-pollution proposals fall into three categories: simple heuristic selection strategies, reputation-based approach, and recommender-based approach. Simple heuristic selection strategies include popularity-based [2] and random-based [5]. The popularity-based strategy makes download decisions based on the popularity of replicas in query responses, and the random-based strategy randomly chooses replicas in query responses for download targets. While simple, both strategies are vulnerable to manipulation.

The reputation-based approach like Credence [6] gauges file authenticity based on voter peers' reputation to assist download decision making. While attractive, this approach has limitations in its reputation system, including bootstrapping of new peers, identity whitewashing, and Sybil attacks. Leveraging social networks, our solution can address these limitations.

The recommender-based approach maintains a trusted,

centralized database which contains fingerprints (or content hashes) of authentic objects [7]. Peers base their download decisions on the database queries. Our proposed recommender differs from this approach in that it exploits trust relationships embedded in social networks and allows participating users collectively to maintain the database over their social links.

Trust networks (or social networks) have recently been utilized to handle Sybil attacks in different applications [8], [9], [10], [11]. The basic insight is that any individual user cannot create an arbitrary number of trust links in the social network and the attack power is strained by the limited number of the trust links. Our work follows the same insight.

A. Background: Random Routes

Random routes are a special form of random walks. The difference between random walks and random routes is routing decisions: the routing decision for random walks is *random* while that for random routes is *deterministic*. In random routes, at each hop, the current node forwards a message to a predetermined outgoing link based on the incoming link of the message. Assume a node with d edges $x_1x_2\cdots x_d$ and a random permutation of the d edges $y_1y_2\cdots y_d$. Then, a random route entering the node via edge x_i will always exit via edge y_i. Random routes have a *convergence* property: Once two random routes traverse the same directed edge, they will merge and stay merged. Moreover, if two random routes converge on some directed edge, then one of them must start in the middle of the other (our object descriptor dissemination overriding policy exploits this to limit pollution of malicious descriptors in Section III-B). Random routes are also *back-traceable*: the outgoing edge uniquely determines the incoming edge. Each node X maintains a *registration table* for each of its edge. The ith entry in the registration table for edge e lists the public key of the node whose random route enters X via e at its ith hop. Since we use random routes of a length of L hops, the number of entries in the registration table for e is L. As will be shown later, the registration table allows a node to enforce the descriptor overriding policy in descriptor dissemination. Please refer to [8] for more details about random routes.

III. Design of the Recommender

A. Overview

When a user issues a query to a P2P network, the user also submits the same query to the recommender via her social links. After receiving query responses (without loss of generality, we assume that a query response includes (1) IP address of the node storing the queried file, and (2) content hash of the file) from the P2P network, the user matches content hashes in the query responses with the recommendation on content hash of the file returned from the recommender. If a match is found, the user downloads the file from the corresponding peer(s) associated with the matched content hash. Otherwise, the user resubmits a modified query that contains the recommended content hash to the P2P network. After receiving query responses, the user downloads the file from the corresponding peers.

Decoupled from a specific P2P network, the recommender is built on top of the social network of participating users. Nowadays, an Internet user likely has joined many online social networks (OSNs) such as Facebook, Twitter and Flickr. Note that not all OSNs are suitable for the recommender we aim to build. We have some assumptions: (1) The social links of the underlying social network reflect strong (trust) social connections, e.g., the human-established trust relationships among colleagues, friends or relatives. Attackers are limited in creating social links with honest nodes in the social network, which we call *attack edge* in the paper. (2) Each node in the social network has a pair of private and public keys. (3) Honest nodes in the social network are incentivized to follow protocols while attacker nodes collude with each other deviating from the protocols to prevent the recommender from returning right recommendations.

The recommender is a repository of participating users' opinions on their recently downloaded files. The user opinion is termed as *formatted object descriptor* (FOD) in the paper. A FOD contains *content hash*, and *descriptive metadata* that includes file name, type, bitrate, and other descriptive data fields (e.g., singer name of a song). Along a FOD stored are the public key of the publisher node of the FOD and the FOD hash encrypted by the publisher node's private key. The FOD only indicates whether the object dictated by the content hash matches the description. The descriptive metadata in a FOD is used to determine if the FOD matches a query. Without loss of generality, the descriptive metadata contains sufficient information to determine whether two pieces of descriptive metadata describe the same object. If the descriptive metadata in two FODs describes the same object, then the two FODs are in *descriptive congruence*, regardless of whether or not they contain the same content hash.

Two basic operations are provided by the recommender: (1) $put(d)$ which allows a user to publish a FOD d into the recommender via her social links; and (2) $h = get(q)$ which returns a recommended content hash h for a query q. The recommender also allows a user to submit feedbacks on recommendations she received, to refine the recommender, as discussed in Section III-D.

As an anti-pollution component, the recommender will draw various attacks as described in Section III-E. Hence, the recommender needs to address two important issues: (1) How to maintain FODs in the underlying social network facilitating search while containing pollution of malicious FODs? (2) How to aggregate relevant FODs for user queries and form right recommendations in the presence of various attacks? To produce right recommendations with high probability in the presence of those attacks, the recommender: (1) utilizes the concept of random routes and a novel dissemination overriding policy to spread FODs across the underlying social network (Section III-B); (2) uses multiple random routes to aggregate relevant FODs from the underlying social network and employs a simple ranking algorithm to form the final recommendation for user queries (Section III-C); and (3) allows users to submit feedback on received recommendations

to refine itself (Section III-D).

B. FOD Dissemination

We use random routes of L hops to disseminate FODs. Consider a user who wants to publish a FOD d to the recommender via her social links. Let X denote the associated node in the social network. Node X forwards d over all its outgoing links. Upon receiving d, each node Y along a random route stores d at a descriptor database (DDB) corresponding to the incoming link [1] over which d is received, if d does not exist in this DDB. If Y is not a tail node of the random route, Y forwards d to the pre-determined outgoing link corresponding to the incoming link. This dissemination process allows a node X with m social links to store d on a set of up to $L{\cdot}m$ nodes. Dissemination redundancy over multiple social links increases visibility of FODs.

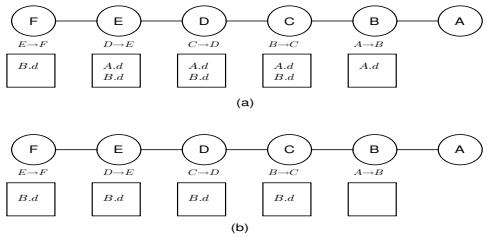


Fig. 1. FOD dissemination. (a) w/o the overriding policy. (b) w/ the overriding policy.

If an attacker node with m attack edges launches Sybil attacks by creating multiple identities behind each attack edge, each of these identities can advertise their malicious FODs only to a subset of the $L{\cdot}m$ nodes due to the convergence property of random routes. Simply put, attacker nodes collectively having m attack edges pollute at most $L{\cdot}m$ nodes even with Sybil attacks; contamination of malicious FODs is thus controlled by random routes — the pollution power is throttled by the number of attack edges instead of the number of identities.

To further limit contamination of malicious FODs, we propose a novel policy *descriptor dissemination overriding*. The basic idea is that a downstream publisher node's FOD overrides an upstream publisher node's FOD along the random route, if the two FODs are in *descriptive congruence* (defined in Section III-A). The overriding policy works as follows: upon receiving a dissemination message for a FOD d, each node Y checks whether its local database has any FOD in descriptive congruence with d (in other words, node Y checks whether it has published a FOD in descriptive congruence with d). If so, node Y discards the dissemination message; otherwise, Y checks if the corresponding DDB associated with the incoming link of the dissemination message contains any FOD in descriptive congruence with d. If no such a FOD is identified, Y stores d in this DDB and forwards the dissemination message via the pre-determined outgoing link (if Y is not a tail node). Otherwise, Y needs to determine whether d overrides the existing FOD: if the publisher node of d is a downstream node of the publisher node of the existing FOD on the random route that includes this incoming link [2], then d overrides the existing one by replacing it in the DDB and Y forwards the dissemination message to the pre-determined outgoing link (if Y is not a tail node of the random route); otherwise, Y simply discards the dissemination message.

Figure 1 illustrates FOD dissemination w/ and w/o descriptor overriding. A is an attacker node while the other five nodes B to F are honest nodes. $A{\rightarrow}B$ is an attack edge. $A{\rightarrow}B{\rightarrow}C{\rightarrow}D{\rightarrow}E{\rightarrow}F$ is a random route. A and B each advertises their FODs $A.d$ and $B.d$ for the same object, respectively. That is, malicious FOD $A.d$ and authentic FOD $B.d$ are in descriptive congruence. Without descriptor overriding, A and B each stores their FODs $A.d$ and $B.d$ on the nodes along a random route of 4 hops, respectively. Nodes B, C, D and E are polluted by $A.d$. As shown in Figure 1 (b), descriptor overriding removes $A.d$ from the corresponding DDBs at B, C, D and E, reducing contamination scope of $A.d$, since B is a downstream node of A on this random route.

The overriding policy does not affect much visibility of authentic FODs. For example, if A is an honest node and B is an attacker node. B is determined to drop A's dissemination messages anyway, making A's FODs unable to reach nodes further downstream. If A and B both are honest nodes which are incentivized to publish only authentic FODs, the overriding policy makes each node on the random route keep only one copy of the authentic FOD (this does not reduce visibility of authentic FODs but only the copies of authentic descriptors; as will be discussed in the subsequent section, keeping a single copy of the authentic FOD for each content hash on the node is sufficient, which is shown in Line 2 of Algorithm 1).

C. Recommendation Generation

When a user issues a query in the P2P network, the user also submits the same query to the recommender. Suppose the associated node in the underlying social network for this user is node X with m social links. Then, node X duplicates the query over all its m social links to aggregate relevant FODs. Each query takes a random route of length L hops. Upon receiving the query, each node Y along the random route checks its local DDB and DDBs associated with its social links and sees if there is any FOD matching the query. If one such FOD is identified, the pair $(content\ hash, public\ key)$ is extracted from the FOD. The query response is routed back to the query initiator along the random route in a reverse direction: each node Y along the route inserts its identified pairs (if any) to the response message (only one pair is kept for duplicate pairs).

Upon receiving m query response messages, X needs to rank query responses in order to produce a right commendation

[1] A node with m social links will have m DDBs associated with these incoming links and a local DDB that stores its own FODs.

[2] Node Y determines this by simply comparing the two publisher nodes' entry numbers in the registration table associated with the incoming link: the publisher node with a lower entry number is the downstream node.

with high probability. Algorithm 1 illustrates the ranking algorithm which returns a recommended content hash based on the query responses. The returned content hash is the fingerprint of the file that the user is looking for and is used to guide download decision making.

Algorithm 1 X.ranking()

Require: R: a list of $(content\ hash, votes)$ pairs, initially empty
1: **for** each query response message **do**
2: list $l \longleftarrow$ extract unique content hashes from the response message
3: **for** each content hash $h \in l$ **do**
4: **if** $h \notin R$ **then**
5: insert $(h, 1)$ to R
6: **else**
7: extract (h, c) from R
8: replace (h, c) with $(h, c+1)$ in R
9: **end if**
10: **end for**
11: **end for**
12: sort R in descending order according to $votes\ c$
13: return the content hash in the top-ranking pair

As described above, FOD dissemination and aggregation both use L-hop random routes. If a random route of the query initiator intersects with a random route of a publisher which advertised a FOD matching the query, the content hash advertised by this publisher is very likely in the query response (the object overriding policy may make the publisher's content hash not included in the query response; this policy could shorten a publisher's FOD publishing scope). Because of the limited number of attack edges, the random route of the query initiator is more likely to intersect with that of an honest publisher than that of a malicious publisher. An exception is a query initiator with a nearby attack edge. Hence, (dissemination and query) message redundancy is used to address such nodes. Due to the presence of malicious content hashes in the query responses, the query initiator needs to distinguish authentic content hashes from malicious ones. We use a simple ranking algorithm: the content hash receiving the most votes is recommended as the fingerprint of the file the user is going to download.

D. Feedback on Recommendations

Users are incentivized to submit feedback on recommendations in order to progressively improve the recommender. If a recommended content hash is authentic, the user will publish authentic FODs for this file as described in Section III-B. If the recommended content hash is inauthentic, the user needs to perform two tasks. First, the user blacklists the nodes [3] (uniquely identified by their public keys) whose responses include this recommended content hash and removes the FODs advertised by these nodes from the DDBs on her associated node in the social network. Second, via the associated node, the user publishes a "thumb-down" FOD, which contains one additional field indicating objection to the contained content hash. Table I shows the FOD dissemination overriding policy in the presence of both usual and thumb-down FODs. The downstream usual FOD always overrides the upstream FOD (whether usual or thumb-down) as long as the two FODs are in descriptive congruence. "Conditional" means that the downstream thumb-down FOD overrides the upstream FOD only if the FODs are in descriptive congruence and contain the same content hash. Note that the thumb-down object FOD is not included in query response messages, and it solely aims to throttle pollution of malicious FODs by the overriding policy.

[3]A more effective approach would be to blacklist the link from which the inauthentic FOD originates. We leave this to our future work.

TABLE I
OBJECT DESCRIPTOR OVERRIDING POLICY.

	upstream usual FOD	upstream thumb-down FOD
downstream usual FOD	Yes	Yes
downstream thumb-down FOD	Conditional	Conditional

To summarize, feedback on recommendations, on one hand, aggressively spreads authentic FODs, and on the other hand, reduces contamination of malicious FODs by blacklisting and descriptor overriding, thereby progressively improving the recommender. Blacklisting is not only an incentive to keep honest users following the protocols but also a deterrent to attackers. For a node which intentionally or inadvertently deviates from the protocols, the node may find it unable to publish and search FODs via its social links (nodes one the other side of the social links or near to the social links will drop its dissemination and search messages if the node is in their blacklists). The node's social links become useless and the node needs to whitewash itself by creating a new set of social links with a new pair of public and private keys, in order to regain access to the recommender. However, construction of a new set of such social links is prohibitive.

E. Attacks

The recommender could become target of various attacks. One might think of legal attacks such as lawsuits against the recommender. This might not be a concern due to the fact that the recommender is fully decentralized and maintains only content hash.

Leveraging attack edges, attacker nodes can collude with each other to launch technical attacks against the recommender by deviating from the protocols. Below, we present a list of possible attacks. We do not claim that it is a complete list but we believe it covers most of the potential attacks. We have evaluated the performance of the recommender upon these attacks.

Publishing malicious FODs. To pollute the recommender for a specific file, the attacker nodes each can publish malicious FODs either with the same content hash or with different content hashes, over all attack edges. However, with the descriptor overriding policy in dissemination and content hash ranking by votes in forming recommendations, it is in their best interests to publish malicious FODs with the same content hash for the specific file (to maximize the number of votes for this content hash).

Attacks in routing authentic FOD dissemination mes-

sages. Upon receiving an dissemination message from an honest node, the attacker node simply discards and stops forwarding the message, thereby reducing the visibility of authentic FODs.

Attacks in routing query messages. Upon receiving a query message, the attacker node stops forwarding the message along the random route, and loads all the $(content\ hash, public\ key)$ pairs from the relevant malicious FODs advertised by all the attacker nodes, to the query response message.

Abusing feedback on recommendations. The attacker node exhaustively explores authentic FODs by repeatedly issuing queries and then publishes thumb-down FODs for each such content hash contained in the authentic FODs.

Whitewashing. If the attacker node becomes notorious as being blacklisted by many honest nodes, the attacker node may choose to whitewash, i.e., to leave and rejoin the social network with a new identity (new public and private keys). However, whitewashing is prohibitive because the attacker node needs to rebuild attack edges.

IV. Evaluation

A. Experimental Setup

In this section we aim to answer a question: How well does the recommender perform by providing right recommendations, in the face of those attacks as outlined in Section III-E?

Few social network datasets are publicly available to us. Thus, we use the widely accepted Kleinberg's synthetic social network model [12] in our evaluation, which generalizes from theWatts-Strogatz model [13]. We use the model to instantiate different $10,000$-node graphs with different random seeds. The average node degree is 20. Attacker nodes exploit attack edges to launch attacks. We select attacker nodes as follows: we start from a randomly-chosen "seed" node and perform a breadth-first search from the seed. Nodes encountered are marked as attacker nodes that collude with each other until the total number of attack edges reaches a certain value. The number of attack edges used in our simulations is 108, which means that the attackers have to convince 108 real human beings to be their friends. We believe that it is reasonable to assume the hardness of creating 108 such social links by the attacker. We also consider the attacks discussed in Section III-E except whitewashing in our experiments.

B. Results

In this section we present our results averaging over 10 simulations with different random seeds. We first studied impact of random route lengths, number of honest nodes as FOD publishers and feedback submission, on performance of the recommender.

In each simulation, the attacker nodes publish their malicious FODs for a specific file via the attack edges. We also randomly choose a set of honest nodes to publish authentic FODs for the specific file. Then, the rest of honest nodes each issues a query for this file to request a recommendation. We define *success rate* as fraction of queries that get a right recommendation, which is used to quantify performance of the recommender under attacks.

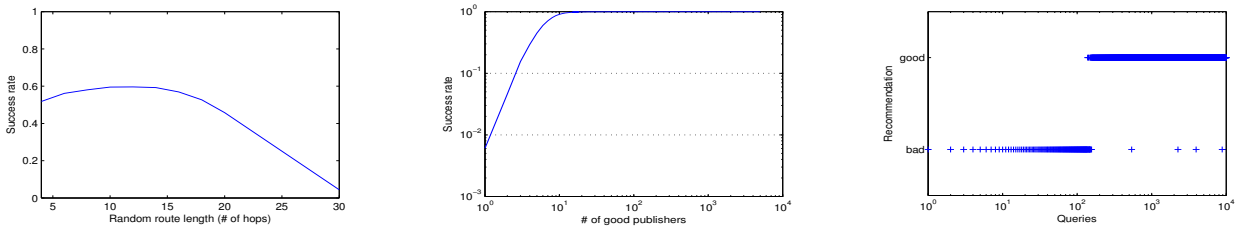
Figure 2(a) depicts the success rate with respect to random route lengths. It shows that a random route length of 12 hops yields the best performance, because a shorter random route decreases the visibility of authentic FODs, and a longer random route, on the other hand, increases pollution scope of malicious FODs. We thus use random routes of 12 throughout the rest of the experiments unless otherwise specified.

Figure 2(b) plots the success rate as the number of honest nodes as FOD publishers varies. The success rate quickly ramps up as the number of honest publisher nodes increases from 1 to 10. When the number of honest publisher nodes reaches 20, the success rate is over 0.99. Thus, we believe the recommender can provide good recommendations to defend against pollution, if a small number of honest nodes collectively publish their authentic FODs.

Figure 2(c) shows impact of feedback on recommendations for a particular simulation. In each simulation, the attacker nodes publish their malicious FODs for a specific file via their attack edges. Initially, one honest node is randomly chosen to publish the authentic FOD for the specific file. Then, for the list of the remaining honest nodes, each time we randomly choose one and remove it from the list. The selected honest node issues a query for the file and then submits feedback on the received recommendation. We repeat it until the list is empty. The success rate is calculated based on the list. We ran the simulation 10 times with different random seeds. The success rate averaging over the 10 simulations is 0.98. This shows that, despite one single honest publisher node in the beginning, feedback on recommendations, is very effective in disseminating authentic FODs and reducing pollution scope of attackers' FODs.

In Figure 2(c), the x-axis denotes the order of the queries issued by the honest nodes and the y-axis represents received recommendations by the queries. Note that almost the first 150 queries received bad recommendation due to rampant malicious FODs. However, the feedback of the first 150 queries, mostly thumb-down FODs, significantly reduces contamination scope of the malicious FODs due to the descriptor overriding policy and blacklisting, thereby making voice of the authentic FODs heard. The feedback on good recommendations by the previous queries makes the voice of the authentic FODs louder and louder. Nearly all the subsequent queries get good recommendations except four queries. We found that the four query initiator nodes actually were surrounded by many attack edges and they were in the area heavily polluted by the malicious FODs. For all the queries receiving a bad recommendation, we resubmit them to the recommender and all the queries get a good recommendation. This means that the average number of query submissions to get a good recommendation is 1.02.

The last experiment compares performance of the recommender with that of heuristic anti-pollution strategies (random-based and popularity-based selection strategies described in Section II). We assume a 10,000-node P2P network where



(a) Success rate vs. random route length w/o feedback

(b) Success rate vs. number of honest nodes as FOD publishers w/o feedback

(c) Recommendations vs. feedback

Fig. 2. Performance of the recommender.

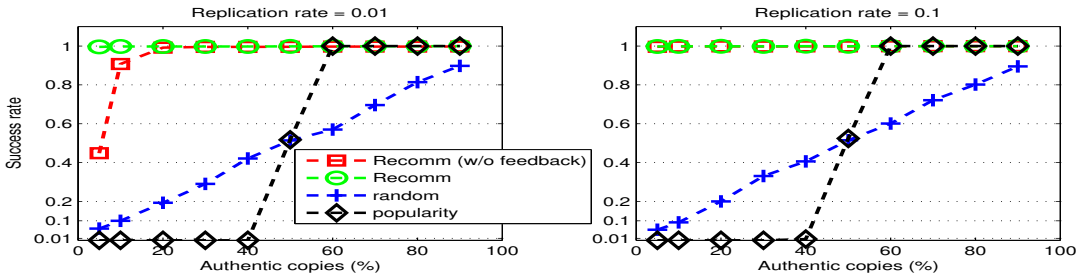


Fig. 3. Performance comparison. "Recomm" represents the recommender.

each honest peer/user has a corresponding node in the social network while attackers have 108 attack edges in the social network as specified in Section IV-A. In the P2P network, for a specific file we assume various replication rates (i.e., the fraction of nodes that claim possession of the file) and various authentic copy rates (i.e., the percentage of copies is authentic, ranging from $5-90\%$). Note that attackers can recruit as many nodes as possible for pollution in the P2P network but are limited in the number of attack edges in the social network underlying the recommender.

We assume that the P2P network returns up to 100 responses for a query for a specific file. When making download decisions, we compare random-based selection, popularity-based selection and the recommender. For a specific file, $1,000$ queries from different, randomly-chosen honest peers are considered. Figure 3 plots the results for replication rates of 0.01 and 0.1 respectively. Two important observations can be made: (1) The recommender significantly outperforms random-based and popularity-based strategies for various replication rates (the results for other replication rates are omitted due to space limitations); (2) Feedback on recommendations is especially effective when the number of authentic file copies is small.

V. Conclusions

In this paper we present a recommender to defend against pollution in the P2P file sharing network. What makes the recommender an attractive anti-pollution solution, is that it leverages users' trust links in their social network to manage their opinions on previously downloaded files and to provide recommendation on file downloads. The primary goal of the recommender is to generate right recommendations in face of various attacks. We present novel FOD dissemination, recommendation generation and feedback on recommendations to achieve this goal. Via simulations, we show that the techniques, combined together, effectively counteract pollution in the P2P network. Not all social networks are suitable for the recommender. One assumption we made is that social links in the underlying social network must reflect strong human-established connections which requires nontrivial efforts to acquire. We plan to crawl real social network graphs to validate our results in our next step.

References

[1] J. Liang, R. Kumar, Y. Xi, and K. Ross, "Pollution in p2p file sharing systems," in *Proceedings of IEEE INFOCOM*, vol. 2, pp. 1174–1185, March 2005.

[2] 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*, pp. 68–77, 2005.

[3] "Overpeer." www.overpeer.com, Sept. 2008.

[4] U. Lee, M. Choi, J. Cho, M. Y. Sanadidi, and M. Gerla, "Understanding pollution dynamics in p2p file sharing," in *Proceedings of IPTPS*, Feb. 2006.

[5] 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 ACM SIGMETRICS*, pp. 38–49, June 2005.

[6] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing," in *Proceedings of NSDI*, pp. 1–14, May 2006.

[7] "Sig2dat tool for fasttrack network." www.geocities.com/vlaibb/tools.html, Sept. 2008.

[8] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," in *Proceedings of ACM SIGCOMM*, pp. 267–278, Sept. 2006.

[9] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2008.

[10] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi, "Ostra: Leveraging trust to thwart unwanted communication," in *Proceedings of USENIX NSDI*, April 2008.

[11] N. Tran, B. Min, J. Li, and L. Submaranian, "Sybil-resilient online content voting," in *Proceedings of the 6th USENIX NSDI*, 2009.

[12] J. Kleinberg, "The small-world phenomenon: an algorithm perspective," in *Proceedings of ACM STOC*, pp. 163–170, 2000.

[13] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, June 1998.