Defending against Sybil Nodes in BitTorrent

Jung So, Douglas Reeves

Defending Against Sybil Nodes in BitTorrent

Jung Ki So and Douglas S. Reeves

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
{jkso,reeves}@ncsu.edu

Abstract. BitTorrent and its derivatives contribute a major portion of Internet traffic due to their simple and scalable operation. However, the lack of security mechanisms makes them vulnerable to attacks such as file piece pollution, connection slot consumption, and bandwidth exhaustion. These effects are made worse by the ability of attackers to manufacture new identities, or Sybil nodes, at will. The net effect of Sybil nodes and weak security leads to inefficient BitTorrent operation, or collapse. In this paper, we present defenses against threats from Sybil attackers in BitTorrent. A simple, direct reputation scheme called *GOLF* fosters peer cooperation to exclude potential attackers. Locality filtering tentatively identifies Sybil nodes based on patterns in IP addresses. Under the proposed scheme, Sybil attackers may still continue malicious behaviors, but their effect sharply decreases. Comparison to existing reputation models shows GOLF effectively detects and blocks potential attackers, despite false accusation.

Keywords: BitTorrent; Sybil attacks; Reputation;

1 Introduction

Peer-to-Peer (P2P) systems account for a major portion of Internet traffic. The P2P paradigm enables a wide range of applications to operate as scalable network services; examples are file sharing, VoIP, and media streaming. The BitTorrent protocol [1], is one of the most popular approaches to P2P file-sharing. This protocol encourages maximum peer cooperation to distribute files. BitTorrent-like systems, such as Vuze (Azureus), uTorrent, BitComet, Tribler, and PPLive, contributed more than 50% of all P2P traffic, and roughly one third of all Internet traffic, in 2008/2009 [2].

P2P systems in general are quite robust to failures, and adapt readily to rapidly-changing conditions. Unfortunately, systems based on BitTorrent may be vulnerable to deliberate attacks by determined adversaries [3,4,5,6]. This is because BitTorrent incorporates few security mechanisms, or mechanisms that are only partly effective. For instance, although the BitTorrent protocol includes coarse-grained data integrity checking (i.e., a SHA-1 hash image per piece), it is highly vulnerable to contamination by fine-grained data pollution (uploading of fake blocks). Dhungel et al. [7] showed that even one polluter in a channel can degrade a streaming service severely in PPLive (i.e., a BitTorrent-like streaming application). As another example, attackers are able to hinder a compliant peer from exchanging data with potential neighbors by fake control messages [6]. In addition, attackers can exhaust legitimate peer's upload bandwidth [8].

Defending against attacks on P2P systems is made more difficult by the fact that one attacker can generate a great number of false identities at little cost; this is known as the Sybil attack [9]. The Sybil attack is a fundamental and pervasive problem in P2P systems. Attackers can use these identities to avoid detection, and to avoid repercussions for their malicious behavior. Since victims cannot differentiate Sybil attackers (Sybil nodes) from legitimate peers, it is difficult for a peer to avoid the above-mentioned attacks. Therefore, prevention or mitigation of Sybil attacks is key to making systems such as BitTorrent more robust.

Sybil nodes can aggressively attempt to compromise the swarm, disseminate polluted (corrupted) file pieces, and exhaust peer resources. To address these problems, we propose a light-weight reputation scheme, called good leecher friends (*GOLF*), combined with locality filtering. GOLF detects polluted file blocks through a light-weight, fine-grained integrity check. Peers using GOLF share information with each other about attackers. This information is weighted by their history of previous, mutually-successful exchanges. By this means, peers can learn about and avoid attackers. Locality filtering flags possible Sybil nodes, based on similarities in their IPv4 addresses. The BitTorrent tracker maintains a *locality filter* that classifies participants. This filter is updated when a peer joins or leaves the swarm, and is distributed to seeders by the tracker.

The primary aim of this paper is to mitigate the malicious impact from Sybil nodes through peer cooperation, in a way that is lightweight, and easily integrated with BitTorrent. As long as each peer cooperates with others, it can protect itself from attackers by use of GOLF with locality filtering. The proposed scheme has been implemented, and is shown to sharply reduce the impact of Sybil nodes. For example, the bandwidth cost is reduced more than 10 times in the presence of Sybil nodes. Comparison to other reputation schemes [10,11,12] shows GOLF effectively detects Sybil nodes, despite the dissemination of false information from neighbors. GOLF is a decentralized approach, and does not require a central authority for collection or dissemination of reputation information. Finally, GOLF improves the detection of attackers in BitTorrent [11] by a factor of 3 or greater.

2 Related Work

Douceur [9] introduced the Sybil attack in distributed systems. To exclude Sybil nodes, a central authority can be a solution. A trusted third party (TTP) can issue certificates for authorized participants, using public key or identity-based cryptography. This approach has the standard drawbacks of a centralized infrastructure (overhead, lack of scalability and reliability), as well as a requiring a sacrifice of anonymity. A system that charges for IDs can mitigate (but not prevent) the Sybil attack. The drawback is that barriers to entry discourage wide participation and cooperation.

Decentralized approaches, such as resource testing [13,14], trusted networks [15,16], and reputation [17,18] are alternative defenses against the Sybil attack. Resource testing based on the fact a Sybil node has a limited resource may bring about false positives in a environment where nodes have heterogeneous capacities. Yu et al. [15] showed that a trusted network (i.e., a social network) can mitigate the effects of Sybil nodes. Use of a trusted network may however incur cold start problems (i.e. newcomer discrimina-

tion), increase reliance on a separate infrastructure, and limit scalability. Sybilproof [17] considers Sybil strategies, where a user is only concerned with increasing his own reputation, and the impact of "badmouthing" (i.e., false accusations).

Piatek et al. [12] attempted to achieve persistent incentives *across swarms* in BitTorrent systems. Their one-hop reputation scheme uses public/private key pairs for identity, which generates key management overhead and limits scalability and anonymity. Lian et al. [19] evaluated private experience and shared history to achieve a balance of reputation coverage and accuracy. Such schemes are vulnerable to whitewashing (a type of Sybil attack) and collusion.

Rowaihy et al. [14] reduced Sybil attacks with an admission control scheme that makes use of client puzzles and public key cryptography. Their scheme requires a trusted third party, creates artificial barriers to entry, and has the overhead of constructing a hierarchy. Sun et al. [20] investigated the effect of using Sybil nodes as a freeriding strategy. MIS scheme [5] detects a fake block (pollution) attack in P2P streaming applications through the use of hash functions at the block level.

The blacklisting approach [21] excludes IP address ranges of the attackers. SafePeer plugin, a blacklist approach, requires a delay of between 2 and 20 minutes to import a database of blacklisted IP addresses [22]. This drawback has limited usage of the SafePeer plugin. Also, it may mistakenly reject some benign peers in blacklisted IP address ranges.

The rest of this paper describes a fully distributed scheme for dealing effectively with content pollution and Sybil attacks. There is no penalty for newcomers (cold-start problem), and no sacrifice of anonymity. There is no reliance on a public key infrastructure, or on a trusted third party (other than the use of a tracker, which is a standard part of the BitTorrent protocol). There is no startup delay. The scheme uses direct reputation evidence based on bartering volume in a swarm, and the effects of badmouthing and collusion are considered. Careful attention is given to the use of space- and communication-efficient encoding of information.

3 Assumptions & Threat Models

3.1 Assumptions

We consider a basic BitTorrent system [1]. We assume the tracker and the torrent website provide correct information, and are available (methods of fail-over and redundancy are known and used). There is no central authority or trusted third party for peer authentication. Therefore, no peer can tell whether a peer identity has been faked, and all participants are initially assumed to be legitimate (non-malicious). A seeder can adapt different seeding algorithms to distribute file pieces to leechers. Each leecher follows the rate-based tit-for-tat (TFT) unchoking and LRF piece selection schemes [11].

[1] A BitTorrent system consists of a *tracker*, *seeders*, and *leechers*; this is collectively referred to as a *swarm*. The tracker is both a bootstrap server, and a coordinator informing leechers of potential neighbors. Each peer can be either a leecher or a seeder. A leecher has an incomplete file and a seeder has the complete file. Leechers obtain file pieces from other peers. Upon completion of file downloading, a leecher becomes a seeder. Readers are referred to [1] for more details.

We assume that malicious nodes can act individually, or together (in collusion with one another). An individual node has limited resources but is able to generate fake identities. A determined adversary can create a large number of Sybil nodes and effectively control them. We believe it is considerably easier to create effective Sybil nodes in limited address ranges. [21] showed that attackers are usually located in small network ranges, and our measurement study supports this conclusion as well (in section 5.2).

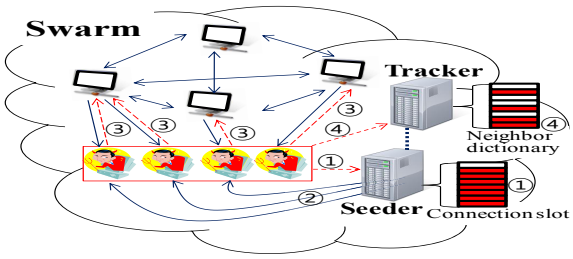


Fig. 1. Overview of malicious behaviors from Sybil nodes.

3.2 Threat models

Leechers may experience the effects of malicious behavior by Sybil nodes during piece exchange [4,7,6]. Malicious peers will cheat the seeder and the tracker [3]. Figure 1 shows Sybil nodes can annoy participants with the following attacks.

Connection slot attack (①): Sybil nodes can aggressively request TCP connections to consume limited connection slots. Once established, the Sybil node can send its neighbors (seeders and leechers) fake control messages to maintain their interest. Although the cost of the control messages sent to neighbors is trivial, the attack can make it difficult for non-malicious peers to connect with other benign neighbors. The result will be slow download times, and a decrease in cooperation.

Bandwidth attack (②): Sybil nodes may attempt to greedily consume the upload bandwidth of a seeder. In the event that Sybil nodes occupy most of the unchoke slots of the seeder, benign leechers may be starved (unable to download file pieces from the seeder). In addition, a Sybil node connecting with a benign peer may receive a considerable portion of the upload bandwidth of that peer.

Fake block attack (③): Sybil nodes may send fake blocks to neighbors, to waste their download bandwidth and verification (computation) time. A Sybil node may initially appear to be complying with the TFT protocol. Due to the coarse-grained file piece integrity mechanism (i.e., using hash values of file pieces), verification of fake blocks consumes a non-trivial amount of download bandwidth, reassembly effort, and buffer space, and the victim has to re-download the genuine pieces from other neighbors.

Swarm poisoning (④): Malicious nodes create fake (Sybil) IDs and attempt to join a swarm. While the tracker may be trustworthy, it cannot discriminate whether a joining peer is malicious without attack evidence. The tracker may therefore suggest Sybil nodes as potential neighbors whenever it is requested to provide neighbor lists.

4 GOLF scheme & Locality filtering

In this section, we present a simple reputation scheme, **GO**od **L**eecher **F**riends (GOLF), with locality filtering. The ultimate aim is to mitigate malicious attacks from Sybil nodes. Leechers cooperate with direct neighbors to combat Sybil nodes by GOLF. The tracker and seeders reduce the impact of Sybil nodes through locality filtering. The GOLF scheme enables a leecher to detect potential attackers by sharing its experiences with direct neighbors. Locality filtering helps the tracker and seeders to discriminate against Sybil nodes, using an efficient data structure for the purpose.

4.1 GOLF scheme

The goal of GOLF is diminishing the effect of attackers. GOLF relies upon cooperation among leechers. To identify the possible Sybil nodes, each leecher uses a filter-based detection mechanism. GOLF expands the local view of attackers to immediate neighbors by exchanging information about past behavior. The local trust value is based on previous TFT volume, and the detection of corrupted blocks.

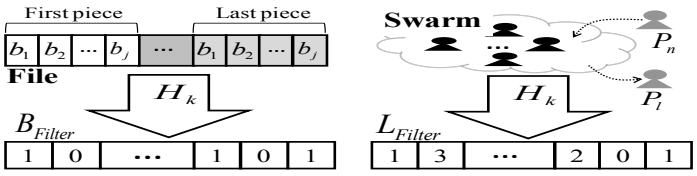
GOLF protocol : GOLF is based on good interactions, or exchanges of legitimate (non-corrupted) blocks between neighbors. If a neighbor interacts successfully and properly, the leecher regards the neighbor as a "friend". Otherwise, the leecher records the neighbor's ID and misbehavior in its attack history. The leecher will refuse connection requests from previously-misbehaving peers. The leecher propagates information about attackers to its direct neighbors, who can use that information in making their own decisions. Consequently, the gossip between friends can exclude potential attackers from connecting.

Block filter against fake block attack : Sybil attackers can directly impact leechers by uploading corrupted blocks[2]. Checking data integrity using the SHA-1 signature of a piece prevents leechers from accepting corrupted pieces, but at significant cost. For instance, Sybil attackers may upload corrupted blocks of a piece, in return for being unchoked (TFT). Other blocks may be uploaded from other peers. When the piece signature fails verification, the leecher will not know which peer(s) provided false blocks.

To tackle this problem, a *block filter* (B_{Filter}) based on Bloom filtering [23] is used. The block filter is a summary of all blocks in the shared file. Figure 2(a) shows the creation steps for B_{Filter}. The original seeder hashes each block in the file with k hash functions, and marks the corresponding k bits in the filter. After processing all blocks, the seeder adds this B_{Filter} to the *torrent* metadata[3]. After obtaining the *torrent* file, leechers do not need to download it again when they rejoin the swarm. Although the size of B_{Filter} in the torrent metadata depends on the number of blocks and the expected rate of false positives, it is very small relative to the size of most files being shared; detailed overhead costs are analyzed in 5.4. Additionally, unlike MIS scheme through HMAC and server's intervention [5], filter-based detection enables each leecher to directly identify a real attacker (polluter).

[2] In the BitTorrent protocol, each file piece (e.g., 256KB) is further divided into blocks (e.g., 16KB per block) for exchange purposes.

[3] The metadata contains information about a file name, its length, SHA-1, and tracker location.



(a) Block filter creation (b) Locality filter update

Fig. 2. The original seeder creates B_{Filter} with all blocks and the tracker updates L_{Filter} with swarm participants. In (a), each piece is divided into even-size blocks (b_1, b_2, . . ., b_j). In (b), peer P_n indicates a newcomer and peer P_l indicates a leaver.

Attacker detection using Block filter : GOLF uses B_{Filter} to counter the fake block attack. Upon obtaining B_{Filter}, leechers can check block integrity. Verification of a block involves repeating the hash functions and checking that the expected k bits in the filter are set. Integrity checking can then be done on individual blocks, rather than solely at the file piece level. Failure to be verified by the block filter indicates the block is corrupted, while successful verification means that the entire piece must still be downloaded and verified (via the SHA-1 hash). A leecher receiving a fake block, or a corrupted file piece, can set a flag indicating this neighbor is unreliable (assumed malicious). Each leecher independently maintains a history of attacks or misbehavior, based on its own direct interactions with other peers. Naturally, each leecher will prefer to cooperate with good leecher friends.

Countering false accusations : Sybil nodes may provide false information to their neighbors concerning their experiences. This has to be considered in the choice of information to use in assessing potential attackers. A Sybil node may falsely accuse a benign peer of malicious behavior. In order to reduce the effect of false accusations, trust is first based on individual (private) experience.

Let D_i^t denote the total downloaded volume of genuine blocks from peer i through rechoke period t[4] and U_i^t denote the total uploaded volume to peer i. A peer computes the *contribution value* C_i^t of each of its directly-connected neighbors i as $\frac{D_i^t}{U_i^t+D_i^t}$, where $0 \leq C_i^t \leq 1$, at every rechoke interval. Note that symmetric exchange between neighboring peers will result in contribution values of .5.

The bartering fraction of a neighbor i of a peer having N neighbors is simply $\frac{D_i^t}{\sum_{j=1}^{N} D_j^t}$. A peer computes the *interaction value* I_i^t of each of its neighbors i as the product of its bartering fraction and contribution value, i.e., $I_i^t = \frac{D_i^t}{\sum_{j=1}^{N} D_j^t} * C_i^t$. The interaction value can range from 0 (minimum interaction) to 1 (maximum interaction), and represents the importance of a neighbor. A neighbor uploading only a small amount of the total of received blocks, or downloading much more than uploading will have a small interaction value.

[4] In a normal TFT unchoking scheme, every rechoke period is 10 seconds.

The *trust value* of a neighbor i, denoted as T_i^t, is computed as $T_i^t = \frac{I_i^t}{\sum_{j=1}^{N} I_j^t}$. The trust values represent the opinion of a peer about the neighbors with which it has directly bartered file pieces.

A peer will compute a *suspicion value* S_k^t for other peers k based on the history of its direct interactions, and information reported by other peers. This value ranges from 0 (not suspected of being malicious) to 1 (known to be malicious). If a peer has directly experienced an attack by neighbor i at rechoke period t, $S_i^{t'}$ will be set to 1 for all $t' \geq t$.

Peers exchange their suspicion values with each other, and use this reputation information to update their own suspicion values. A suspicion value reported by peer i about peer j at rechoke period t is denoted as $\Delta_{i \to j}^t$. Upon receiving this reported suspicion value, a peer updates its own suspicion value S_j^t as

$$S_j^t = [\frac{S_j^{t-1} \times (t-1) + T_i^t \times \Delta_{i \to j}^t}{t}] - T_j^t \quad (1)$$

The term inside the square brackets in equation 1 represents the average degree of suspicion for peer j, while T_j^t reduces this according to the trust directly earned by j. The suspicion value is calculated for neighbors and for peers for which Δ values are received.

A peer makes an independent judgement about other peers, based on the received suspicion values, and stored trust values earned by successful interactions with its neighbors. Since the number of neighbors decides possible bartering ranges for the swarm, the threshold for the suspicion value is set as a fraction of the number of connection slots. A high trust value based on direct experience diminishes the effect of other peers' prejudices against a neighbor. Each peer suspends the decision about whether to suspect a neighbor (to reduce a hasty judgement) until the provider of suspicion information has correctly bartered at least some minimum number of pieces. A malicious attacker will attempt to influence the suspicion value of a benign peer. False accusations correspond to inaccurate high suspicion values. In the following section, the impact of strategic Sybil nodes that attempt to compromise reputation information is considered.

4.2 Locality filtering

Locality filtering reduces network resource exhaustion and swarm poisoning through IP address *binning*. In this approach, a bin represents peers who share the same IPv4 /24 IP address prefix (e.g., 10.9.8.6 and 10.9.8.7 share the same /24 prefix, while 10.9.8.6 and 10.9.5.6 do not). The tracker groups participants with the same IP /24 prefix using a *locality filter* (L_{Filter}). Locality filtering helps a peer avoid Sybil nodes, thereby preserving network resources for benign leechers.

Locality tracking by the tracker : The tracker is charged with monitoring membership / participation in the swarm. The L_{Filter} is an implementation of a counting Bloom filter [24]. As shown in Figure 2(b), the tracker maintains a L_{Filter} that reflects a snapshot of current participants. The set of participants can be (and usually is) frequently changing; the tracker updates the L_{Filter} whenever a peer joins or leaves. Each peer reports its state to the tracker at regular intervals in the normal BitTorrent protocol. For example, when a newcomer joins, the tracker hashes its IP /24 prefix using

k hash functions, and adds 1 to each resulting index value (counter). Conversely, if a known peer leaves the swarm, the tracker decreases the corresponding k index values. The tracker shares L_{Filter} with seeders at regular intervals.

Locality tracking uses L_{Filter} to select neighbors in different IP /24 ranges. The tracker provides the requestor with suggestions for neighbors until it has sent a sufficient number. The tracker randomly selects candidate neighbors. The tracker checks the /24 prefix of each candidate using the L_{Filter}. If the number of peers in the swarm having the same /24 address prefix exceeds a threshold parameter, and one peer in this address range has already been suggested as a neighbor, the tracker will reject additional neighbors in this same address range before sending suggestions to the requestor.

Locality seeding by a seeder : In order to alleviate network resource exhaustion from Sybil nodes, a seeder uses L_{Filter} for effective unchoke allocation. If Sybil nodes take a majority of unchoke slots, benign leechers will potentially suffer data starvation. Locality seeding is helpful in reducing the abnormal selection of Sybil nodes. Such seeding operates similarly to locality tracking. Requesting peers are sorted by some metric, such as download rate, random selection, or service priority. In this order, the seeder checks the requesting peer's IP /24 prefix against the L_{Filter}. If the count for this address prefix is less than a threshold value, the seeder will assign the next unchoke slot to the requesting peer. Otherwise, the seeder will move on (in order) to the next candidate.

5 Evaluation & Discussion

This section presents a trace measurement and the results of applying GOLF with locality filtering. The goal is to understand the performance of the proposed scheme against malicious behavior by Sybil nodes. The experimental setup is first described, followed by the results, and discussion. In order not to impact a real BitTorrent swarm, we report the the results of simulation, rather than mount attacks in actual networks.

5.1 Experimental setup

We developed a BitTorrent simulator that is a faithful implementation of the BitTorrent protocol, with the ability to enable or disable GOLF and locality filtering. The simulator is event-driven, and includes events such as joins and leaves, bartering pieces, unchoking (including optimistic unchoking), and exchange of piece messages. The normal BitTorrent TFT and LRF policies were implemented. Sybil actions such as sending fake blocks, discarding received data from leechers, consuming seeders' bandwidth, and making false accusations were also implemented.

In the simulator, some fraction of the nodes were assumed to be Sybil nodes; the exact fraction is described for each experiment. Peer addresses, except for Sybil nodes, were for the most part located in different /24 address ranges. This assumption is consistent with measurements described in 5.2. A random delay caused by the impact of network topology was added when sending a piece to all peers [8]. According to [25], the volume of the control messages in BitTorrent is negligible compared to piece messages. Thus, we do not reflect delays due to control messages. To reduce simulation

complexity, the network was assumed to have no bottlenecks or transmission errors [8]. Each peer had an asymmetrical bandwidth capacity that reflects the ADSL standard models [26]. Every peer had between 500Kbps and 1.3Mbps for an upload rate. The original seeder had 5Mbps as its upload rate.

Locality tracking was implemented in the tracker module. The simulator included three different seeding algorithms (i.e., bandwidth-first, random, and round-robin seeding) for leecher selection. Results were similar for each, and only the evaluation results for round-robin (RR) seeding are described in this section. The RR seeding algorithm sorts leechers based on their service priority (i.e., leechers having received the least are given the highest priority). This seeding algorithm combined with locality filtering is denoted as *CRR* in the following.

The number of peers was limited to 1,000 nodes, based on a previous measurement study [27]. Each simulation started with one seeder and one tracker. They served all participants in the swarm throughout the simulation. Peers joined the swarm based on an arrival process derived from a real BitTorrent trace log [28]. Once downloading the entire file, a leecher became a seeder until it left the swarm. To explore malicious attacks, the fraction of Sybil nodes was varied from 5% to 50%. File sizes were set between 5 MB to 500 MB; results are shown only for smaller sizes (larger file sizes yielded similar results). A simulation run finished when all benign peers completed the file download. Each simulation was run 30 times to compute 95% confidence intervals.

5.2 Measurement study with RedHat9

We analyzed the distribution of IPv4 addresses of peers in a RedHat9 (1.77GB) trace [28]. The trace reflects downloads over a period of 5 months, and has all events from the tracker of the torrent. The log contains report time, IP address, port number, peer ID, upload and download size, and events. Results are presented for the distribution of IPv4 addresses during the first 5 days of flash crowd events, which are particularly challenging for file sharing systems.

Table 1. Number of peers per IP/24

# of peers in IP/24	Day 1		Day 2		Day 3		Day 4		Day 5	
1	13,306	96.2%	5,049	96.0%	3,451	97.0%	2,624	97.1%	2,230	97.4%
2	439	3.2%	184	3.4%	81	2.3%	60	2.2%	38	1.7%
3	46	0.3%	18	0.3%	11	0.3%	8	0.3%	1	0.0%
4	16	0.1%	3	0.1%	0	0.0%	3	0.1%	1	0.0%
≥ 5	31	0.2%	8	0.2%	13	0.4%	6	0.2%	19	0.8%

Table 1 shows the number of peers per /24 prefix. At least 96% of leechers were in a /24 address range with no other leechers present. Address ranges with 4 or fewer leechers present accounted for 99.2% of all leechers. Accordingly, in the following a threshold parameter value of 5 was used to identify potential Sybil node address ranges.

5.3 Experimental results

We present the results of simulating the proposed scheme against Sybil nodes, for both peer and performance impacts.

Seeder impact : In the first experiment, the seeder was required to distribute 5MB of content to all benign users. The total bandwidth required in order to achieve this included bandwidth that was wasted on malicious (Sybil) nodes. Figure 3(a) shows the total amount of data sent by the seeder for the Round Robin seeding policy. Performance was measured with and without locality filtering. Locality filtering greatly reduces the impact of Sybil nodes. The bandwidth consumed by Sybil nodes is decreased by a factor of 10 or greater if the Sybil node percentage exceeds 10%. This is because the filter helps the seeder allocate most unchoke slots to benign leechers, not Sybil nodes.

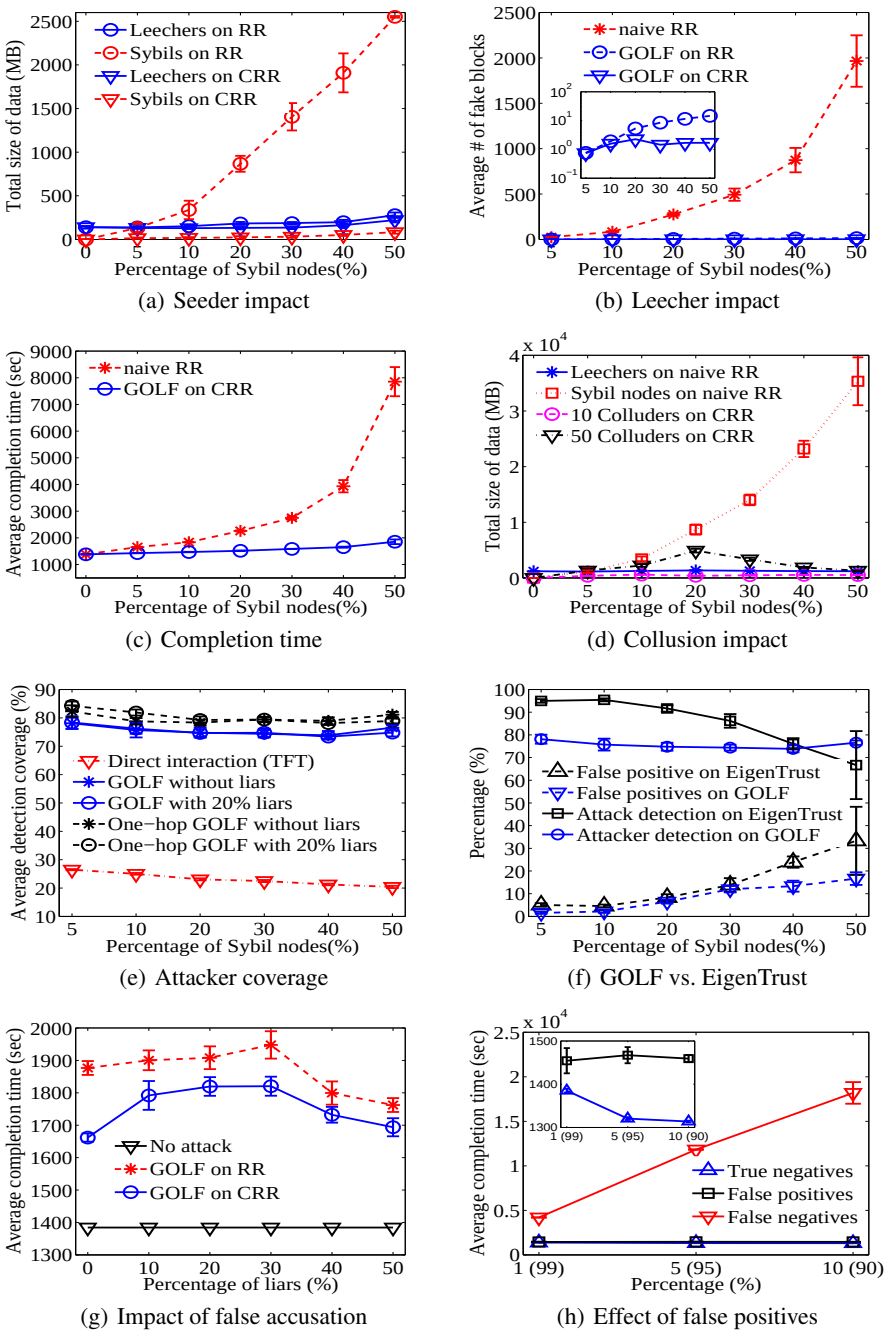
Benign user impact : The second experiment evaluated the average number of downloaded fake blocks per leecher, in a swarm sharing a file of size 100MB. Figure 3(b) shows the results. In RR seeding (without locality filtering), each leecher experienced an exponential increase for the average download rate of fake blocks, as the Sybil node fraction increased. However, the proposed scheme (locality filtering + GOLF) decreased the downloading of fake blocks to almost zero. This is because each leecher discriminates against direct and reported attackers using GOLF.

Completion time : The next experiment investigated the average completion time for benign leechers to download the entire file, for a file of size 100 MB. The results are shown in Figure 3(c). BitTorrent without locality filtering showed exponential increases as the percent of Sybil nodes increased. This is because Sybil nodes occupy unchoke slots of benign peers, reducing the opportunities for benign peers to exchange file pieces with one another. In contrast, the use of locality filtering resulted in near constant download completion times, regardless of the fraction of Sybil nodes.

Collusion effect : Another experiment investigated the impact of collusion among attackers. In this scenario, Sybil nodes were distributed among multiple IP /24 prefixes. The number of distinct prefixes is referred to here as the number of colluders, and was varied. The percentage of Sybil nodes was also varied.

The occurrence of collusion had little impact on the download completion time of benign users, and as such, is not shown. The seeder, however, was affected by the number of colluders. Figure 3(d) shows these results. The upload bandwidth ("total size of data" in the figure) of the seeder increased exponentially as a function of the percent of Sybil nodes without the use of locality filtering. Collusion also affected BitTorrent with locality filtering, until the number of Sybil nodes per /24 address range exceeded the threshold parameter. Thereafter, locality filtering greatly reduced the waste of seeder bandwidth (by a factor of 30 or greater for 50% Sybil nodes). An attacker who is able to spread their Sybil nodes throughout the network will obviously have more impact, but at a significantly higher cost of implementation and deployment.

Attacker detection coverage of GOLF scheme : BitTorrent with TFT is limited in its view. GOLF is intended to disseminate knowledge of attackers slightly more widely, but with limited overhead (no non-neighbor communication or global coordination required). In the next experiment, the effectiveness of GOLF in identifying Sybil nodes was measured. The results are shown in Figure 3(e) as the probability of (correctly) detecting attackers. Three cases are considered: (1) attackers are detected only by direct



(a) Seeder impact (b) Leecher impact

(c) Completion time (d) Collusion impact

(e) Attacker coverage (f) GOLF vs. EigenTrust

(g) Impact of false accusation (h) Effect of false positives

Fig. 3. Evaluation results. I-shaped lines indicate 95% confidence. From (a) to (f), x-axis indicates Sybil nodes' percentage. In (g), x-axis indicates liars' percentage, where 20% Sybil nodes. In (h), the fraction of nodes who were benign, and not suspected of being Sybil nodes (true negatives), is varied from 99% to 95% to 90%. The fraction of nodes who were benign, but (incorrectly) suspected of being Sybil nodes (false positives) and the fraction of nodes who were Sybil nodes, but not suspected (false negatives) are varied from 1% to 5% to 10%. In (b) and (h), the inner graph magnifies the result.

experience (i.e. TFT) [11], (2) attackers are detected based by direct experience or by information provided by immediate neighbors, or (3) attackers are detected based on direct experience, information from immediate neighbors, and information from their neighbors (i.e., one-hop neighbors) [12].

The use of information from immediate neighbors, weighted by their suspiciousness, results in a three-fold increase in the likelihood of detecting Sybil nodes, from about 25% to over 75%. The use of information from one-hop neighbors provided additional benefits in this experiment. On the contrary, one-hop GOLF incurs uncertainty and complexity about one-hop neighbors' trust. Liars (i.e., Sybil nodes falsely accuse leechers) compromised peers' attacker history. Some peers mistakenly rejected connection requests from benign peers. In this experiment, a maximum of 6.76% peers never completed downloading the file because of false information.

Comparison to EigenTrust with false accusations : Generally, reputation systems are vulnerable to false information. Trust in EigenTrust [10] reflects global and local updates. The global vector is liable to be compromised by badmouthing from malicious attackers. Although a local trust value is high, a peer might mistakenly block a connection request from a benign peer. Similarly, liars (Sybil nodes) can make false accusations about other peers in the GOLF scheme.

The last experiment compared GOLF to EigenTrust with respect to detection rates and false positive rates when there are false accusations. Figure 3(f) shows the probability of detecting Sybil nodes and falsely accusing benign peers, for a file of size 100MB. With false accusations, the false positive rate of GOLF is lower than EigenTrust. The percentage of falsely rejected peers out of the total peers ranged from 1.5% (for 5% Sybil nodes) to 16% (for 50% Sybil nodes). For attacker detection rate, EigenTrust is better. To accomplish this, however, EigenTrust requires pre-trusted peers and incurs much higher communication as well as computation overhead. By comparison, GOLF uses a simple computation based on empirical piece interactions, in a distributed manner.

5.4 Discussion

We discuss adversary reactions to GOLF, and the issue of false positives. After that, we analyze the overhead for deploying GOLF with locality filtering.

Adversary Strategies against GOLF : Generally, reputation systems are vulnerable to counter strategies. For example, Sybil nodes may be liars (make false accusations about other peers), may be traitors (engage in productive exchanges before providing false information about other peers), or may be whitewashers (in case of accusation, leave and rejoin with a new identity). The effect of false accusations is mitigated by the weighting by trust (inversely, suspiciousness).

Figure 3(g) shows the average completion time of benign users as a function of the number of neighbors that lie, for a 100MB file. In this experiment, 20% of the nodes are Sybil nodes. The completion time increases about 500 seconds compared to the no attack case. This is because the reports of liars are reflected to benign neighbors and are propagated to their friends. Adjusting the computation of trust to further reduce the effects of liars and traitors remains as future work.

False positives by locality seeding : False positives may occur because of the innocent existence of benign peers in the same /24 address range as Sybil nodes. For instance, a number of benign peers behind NATs may be falsely identified as a Sybil node. They may experience very slow download speeds because of the discrimination of locality seeding. In spite of the delay of getting initial currency (i.e., uploading 4 file pieces), locality tracking can help the peers overcome seeder's discrimination.

Figure 3(h) shows the effect of false positives by locality filtering. It compares to the average completion times with the CRR model for a 100MB file, based on setting detection categories of each peer by locality seeding. A benign node's completion time is not affected much, regardless of whether or not it is suspected. The false positives (i.e., benign peers behind NATs) are delayed around two minutes on average.

Deployment overhead : To deploy the proposed scheme, additional costs are incurred. The size of B_{Filter} depends on the file size, and the size of L_{Filter} depends on the number of participants. We propose the use of Bloom filters, which are well-known space-efficient data structures that are easily updated. The computation overhead requires multiple hash operations to compare the values. B_{Filter} for a 2GB file requires 1MB of storage, and L_{Filter} for 1,000 peers in the swarm requires 8KB of storage. Note that the information sent to the seeder does not have to be the entire counting filter. To reduce size overhead, the tracker can inform the seeder of locality violations using a much smaller Bloom filter.

Communication overhead is due to the need to share B_{Filter}, L_{Filter}, and attacker information. B_{Filter} shared among peers can be included in the torrent file that is already downloaded at the first join time. L_{Filter} can be updated, whenever each seeder queries the tracker to harvest new neighbors. Attacker reports can also be combined with existing control messages.

6 Conclusion

This paper proposes the GOLF scheme with locality filtering to mitigate Sybil attacks in a BitTorrent system. GOLF uses cooperation between directly-connected peers to spread information about suspected attackers. Each leecher learns of such suspicions from neighbors with whom it exchanges file pieces. The input from neighbors is weighted by their past beneficial behavior. Locality filtering helps a seeder evade traffic exhaustion by Sybil nodes, and helps the tracker guide leechers to good neighbors in the swarm. The overhead of locality filtering is mitigated by the use of Bloom filters. Whereas Sybil nodes devastate the performance of the standard BitTorrent, the proposed scheme effectively defends against the malicious behavior of Sybil nodes. By virtue of GOLF with locality filtering, the expected download completion time for non-malicious nodes is affected very little by the Sybil attack. The data that must be uploaded by a seeder when Sybil nodes are present is reduced by a factor of 10 or greater.

Acknowledgments. We thank the anonymous reviewers for their fruitful feedbacks. This work was partly supported by the Secure Open Systems Initiative (SOSI) at North Carolina State University.

References

1. "The bittorrent protocol specification." http://wiki.theory.org/BitTorrentSpecification
2. H. Schulze and K. Mochalski, "Ipoque. internet study 2008/2009." http://www.ipoque.com/study/ipoque-Internet-Study-08-09.pdf
3. M. A. Konrath, M. P. Barcellos, and R. B. Mansilha, "Attacking a swarm with a band of liars: evaluating the impact of attacks on bittorrent," in *P2P Computing*, 2007, pp. 37–44.
4. P. Dhungel, D. Wu, and K. W. Ross, "Measurement and mitigation of bittorrent leecher attacks," *Computer Communication*, vol. 32, no. 17, pp. 1852–1861, 2009.
5. Q. Wang, L. Vu, K. Nahrstedt, and H. Khurana, "Mis: malicious nodes identification scheme in network-coding-based peer-to-peer streaming," in *INFOCOM'10*, 2010, pp.296–300.
6. D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee, "Bittorrent is an auction: analyzing and improving bittorrent's incentives," *in SIGCOMM '08*, vol. 38, no. 4, pp. 243–254, 2008.
7. P. Dhungel, X. Hei, K. W. Ross, and N. Saxena, "The pollution attack in p2p live video streaming: measurement results and defenses," in *P2P-TV '07*. NY, USA: ACM, 2007.
8. K. Shin, D. S. Reeves, and I. Rhee, "Treat-before-trick: Free-riding prevention for bittorrent-like peer-to-peer networks," in *IPDPS '09*. pp.1–12, 2009.
9. J. R. Douceur, "The sybil attack," in *IPTPS '01*, 2002, pp.251–260.
10. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *WWW '03*. 2003, pp. 640–651.
11. B. Cohen, "Incentives build robustness in bittorrent," in *P2PECON '03*, Berkeley, May 2003.
12. M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson, "One hop reputations for peer to peer file sharing workloads," in *NSDI'08*. 2008, pp. 1–14.
13. J. Newsome, E. Shi, D. X. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *IPSN*, 2004, pp. 259–268.
14. H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta, "Limiting sybil attacks in structured p2p networks," in *INFOCOM '07*, 2007, pp. 2596–2600.
15. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *SIGCOMM*, 2006, pp. 267–278.
16. N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting," in *NSDI'09*. Berkeley, CA, USA: USENIX Association, 2009, pp. 15–28.
17. A. Cheng and E. Friedman, "Sybilproof reputation mechanisms," in *P2PECON '05*.
18. H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao, "Dsybil: Optimal sybil-resistance for recommendation systems," in *IEEE Symposium on Security and Privacy*, 2009, 283–298.
19. Q. Lian, Y. Peng, M. Yang, Z. Zhang, Y. Dai, and X. Li, "Robust incentives via multi-level tit-for-tat: Research articles," *Concurr. Comput. : Pract. Exper.*, pp. 167–178, 2008.
20. J. Sun, A. Banerjee, and M. Faloutsos, "Multiple identities in bittorrent networks," in *NETWORKING'07*, 2007, pp. 582–593.
21. J. Liang, N. Naoumov, and K. W. Ross, "Efficient blacklisting and pollution-level estimation in p2p file-sharing systems," in *AINTEC*, 2005, pp. 1–21.
22. "Safepeer." http://wiki.vuze.com/w/SafePeer
23. A. Z. Broder and M. Mitzenmacher, "Survey: Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, 2003.
24. F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "An improved construction for counting bloom filters," in *ESA'06*. 2006, pp. 684–695.
25. A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *IMC '06*. New York, NY, USA: ACM, 2006, pp. 203–216.
26. "Adsl." http://en.wikipedia.org/wiki/Asymmetric_digital_subscriber_line
27. A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in bittorrent systems," in *SIGMETRICS '07*. New York, NY, USA: ACM, 2007, pp. 301–312.
28. "Redhat 9 torrent tracker trace." http://mikel.tlm.unavarra.es/~mikel/bt_pam2004/