An Empirical Study of Collusion Behavior in the Maze P2P File-Sharing System

Qiao Lian[1], Zheng Zhang[1], Mao Yang[13], Ben Y. Zhao[2], Yafei Dai[3], Xiaoming Li[3]
Microsoft Research Asia, Beijing, China[1]
U. C. Santa Barbara, Santa Barbara, CA, USA[2]
Peking University, Beijing, China[3]

Abstract— **Peer-to-peer networks often use incentive policies to encourage cooperation between nodes. Such systems are generally susceptible to collusion by groups of users in order to gain unfair advantages over others. While techniques have been proposed to combat web spam collusion, there are few measurements of real collusion in deployed systems. In this paper, we report analysis and measurement results of user collusion in Maze, a large-scale peer-to-peer file sharing system with a non-net-zero point-based incentive policy. We search for colluding behavior by examining complete user logs, and incrementally refine a set of collusion detectors to identify common collusion patterns. We find collusion patterns similar to those found in web spamming. We evaluate how proposed reputation systems would perform on the Maze system. Our results can help guide the design of more robust incentive schemes.**

I. Introduction

File-sharing networks such as Kazaa and Gnutella have popularized the peer-to-peer (P2P) resource sharing model. In these networks, selfish users often "free-ride," or act in their self interests to exploit the system. Prior research has focused on the use of incentive systems [11] to encourage sharing among users. Despite their effectiveness, these incentives are generally vulnerable to user collusion along with the Sybil Attack [4], where users create and control multiple online identities.

Very little is known about user collusion in real systems. Measuring user collusion is extremely difficult, since it requires a global view of all transactions in a system. To date, most user studies log traffic at the edge nodes while performing queries or membership operations [13], [12]. While informative, these results reveal only a small subset of peer transactions, and do not shed light on active user collusion.

In this paper, we present measurements of user collusion activity in the Maze peer-to-peer file-sharing network [17]. Maze is a popular Napster-like P2P network designed, implemented and deployed by an academic research team at Peking University, Beijing China. As a measurement platform, Maze is unique in two ways. First, our control over the Maze software allows us to deploy and embed customized measurement code inside clients. Second, Maze's centralized architecture means all control and query traffic is logged and available to us. Maze uses a simple incentive system where user points increase with uploads and decrease with downloads. Our central server audits file transfers and adjusts user points accordingly.

In our study, we define collusion as collaborative activity of a subset of users that grants its members benefits they would not be able to gain as individuals. We note that we cannot determine a user's true intent, or definitively whether multiple online identities belong to the same user. Issues such as dynamic addressing via DHCP and shared addresses behind NATs prevent us from reliably detecting the use of multiple virtual identities. Instead of measuring user intent, our study focuses purely on observable action patterns that produce results similar to those produced by colluding users. Quantifying all forms of collusion is a topic to be addressed in ongoing work.

To the best of our knowledge, this is the first empirical study of collusion behavior in an incentive-based P2P system. Our results validate conclusions of previous work on incentive mechanisms [3], [8], [7], but also show that certain types of collusion are difficult to detect and deter. Our work makes several key contributions. First, we describe techniques to aggregate logs and construct collusion detectors that progressively reveal collusion behaviors and their patterns. We believe that these techniques have general applicability to other experimental settings when detecting collusions. Second, we find that while some behaviors are linked to our incentive system, their patterns are nearly identical to those found in Web spamming, albeit at smaller scales. Finally, we evaluate the ability of the EigenTrust reputation system [6], [7] to detect and mark colluders with lower reputations. Our results show that under several collusion patterns, EigenTrust produces non-ideal reputations for both well-behaved and colluding peers.

The rest of the paper is organized as follows. Section 2 gives a brief description of Maze and the measurement data used for this study. Next, Section 3 describes each of our collusion detectors in detail, along with results from the Maze dataset. Then in Section 4, we apply the EigenTrust algorithm to the Maze dataset and analyze the results. Finally, we discuss related work in Section 5 and conclude in Section 6.

II. The Maze Peer-to-Peer system

We begin by providing necessary background information on the Maze file-sharing system. Maze was originally deployed to address issues of data location and load-balancing on FTP servers as part of the T-Net Web search project [14]. T-Net's increasing popularity led to significantly degraded download performance across its limited number of FTP servers. Maze

27th International Conference on Distributed Computing Systems (ICDCS'07)
0-7695-2837-3/07 $20.00 © 2007 IEEE

IEEE
COMPUTER SOCIETY

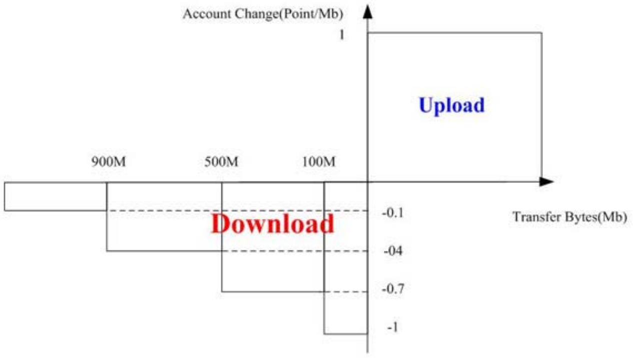


Fig. 1. The Maze point system

provided a way to distribute content while minimizing infrastructure costs.

At login, each Maze peer[1] authenticates to a central server, and uploads an index of its locally available shared files. The central server maintains heartbeats with all online peers, and its central index supports full-text queries across all shared files in the system. In addition to searching for files via the central index, users can browse three peer lists for files: a friend-list, a neighborhood-list, and an altruistic list.

The friend-list is a user-controlled list of friendly peers, initially bootstrapped as a random set of peers by the central server. Over time, these friend-lists across users form a continuously adapting social network. A peer's neighborhood list is a set of peers with the same B-class IP address provided by the server. These are peers likely to share high-bandwidth, low latency links with the local host. Finally, the altruistic list is a collection of peers with the highest "points" as determined by the server. They are hosts who have contributed the most to the system as determined by the Maze incentive system. Their status as "celebrities" in the user population provides additional social incentive for sharing [17].

A peer can browse and download the storage contents of any host on these lists. As of November 2004, more than two-thirds of all downloads are initiated through these peer lists. As will become clear later, these social lists have unexpected impacts on the design of a good incentive system. After each transaction, peers involved send a report to the central server, which adjusts their points accordingly.

A. The Maze incentive system

Maze currently operates using a point system, where peers consume points by downloading files and earn points by uploading files. Download requests are queued according to their points: $requestTime - 3 \log P$, where P is the requester's point total. Frequent uploads provide peers with higher points and faster downloads. The Maze user community voted for an asymmetric point system where uploading receives more points than downloading the same amount of data. While this encourages uploads, it also allows two peers to create a net gain in points after mutual interaction. To allow new users to participate, Maze initializes new users with enough points to download $>$ 1GB of data before its downloads are throttled. The details of the points system are as follows:

1) New users are initialized with 4096 points.
2) Uploads: +1.5 points per MB uploaded.
3) Points used per file downloaded:
    - -1.0/MB downloaded within first 100MB.
    - -0.7/MB per additional MB between 100MB and 400MB.
    - -0.4/MB between 400MB and 800MB.
    - -0.1/MB per additional MB over 800MB.
4) Service differentiation:
    - Requests are ordered by $T = requestTime - 3 \log P$, where P is the requester's point total.
    - Users with $P < 512$ are limited to 200Kb/s.

B. Data collection

We perform our analysis on a log segment gathered during the span of a one-month period from 2/19/05 to 3/24/05. During this period, more than 161,000 users participated in more than 32 million file transfers totaling more than 437 Terabytes. Data gathered for this study includes user point values and the detailed traffic log. Each log entry contains the following: uploading peer-id, downloading peer-id, upload time, transfer start and end times (source), bytes transferred, file size, downloader IP, file MD5 hash, and full file path.

We tried to associate online identities with the physical machine the peer uses to detect when a user was controlling multiple identities. We first used the hash of the hard drive serial number, but later discovered that the serial number is not guaranteed to be unique. Thus to uniquely identify the machine that a peer uses, we concatenate the peer's IP address with the hash of the hard drive serial number. As ongoing work, we are investigating the use of network MAC addresses as an alternative identifier.

All logs are anonymized for user privacy. In this paper, we refer to distinct users using common names (*e.g.* Alice and Bob), and random alphabetic letters to represent 8-bit blocks of an IP address (*e.g.* C.H.97.140).

III. Identifying Collusion Topologies

We now discuss our efforts to detect collusion attempts in the Maze system. Based on our experiences and analysis of the traffic logs, we design a number of collusion detectors aimed at locating different types of collusion patterns. We describe these in detail in this section.

A. Repetition-based Collusion Detection

Our first attempt starts by looking at how users use uploads to generate points. Given the point system generates a net gain from a symmetric operation, colluders can benefit from using only a small "working set" of files to generate points. We use this assumption to generate our first collusion detector.

[1] We use the terms "user" and "peer" interchangeably in this paper. We also use "clients" of peer X to refer to the peers that download from X.

27th International Conference on Distributed Computing Systems (ICDCS'07)
0-7695-2837-3/07 $20.00 © 2007 IEEE

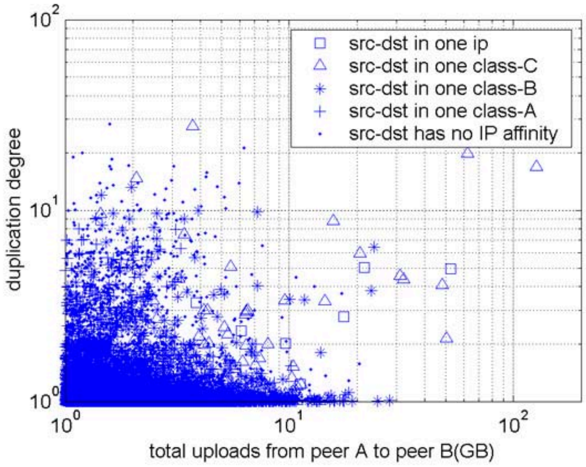
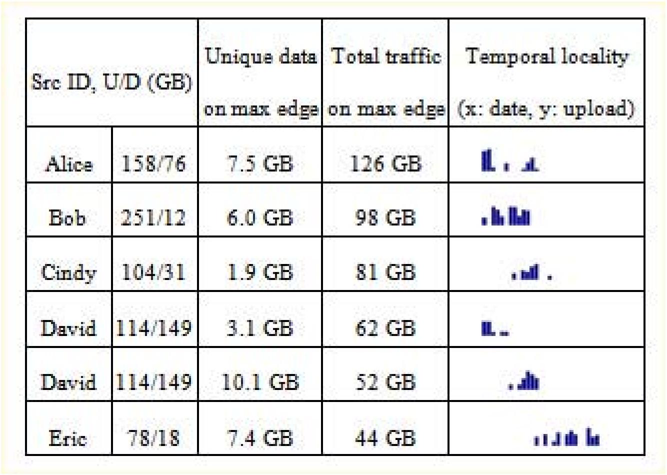


Fig. 2. Duplication degree in uploading from peer A to peer B

TABLE I

TOP 6 EDGES WITH THE MOST REDUNDANT TRAFFIC



Src ID, U/D (GB)		Unique data on max edge	Total traffic on max edge	Temporal locality (x: date, y: upload)
Alice	158/76	7.5 GB	126 GB	
Bob	251/12	6.0 GB	98 GB	
Cindy	104/31	1.9 GB	81 GB	
David	114/149	3.1 GB	62 GB	
David	114/149	10.1 GB	52 GB	
Eric	78/18	7.4 GB	44 GB	

Detector 1: (Repetition detector) Colluders generate large amounts of upload traffic with repeated content.

We examine our transaction log and construct a large graph, where vertices represent individual users and edges represent aggregated file transfers between users. This results in a directed graph with roughly 4.5 million individual edges. Out of all edges, 221,000 (roughly 4.9%) contain duplicate files in the transfer traffic. We define duplication degree as the ratio of total upload traffic (bytes) to amount of non-duplicated data (bytes). A high duplication degree means a small proportion of traffic across the link is unique.

We plot the duplication degree of all edges against their total upload traffic as a scatter plot in Figure 2. For thresholds set at duplication degrees of 5, 10 and 20, there are 890, 148, 27 edges with duplication degree greater than each threshold. Since this data is generated from activities performed over the span of a month, it is highly likely that many of these peers are actively colluding. We also note that colluders are likely to use nearby machines to perform the transfers. Such network locality maximizes throughput and gain from collusion. We show this in Figure 2: Duplication degree in uploading from peer A to peer B by classifying edges by the IP affinity between the two peers. IP affinity confirms that edges with high amounts of duplicate traffic are likely to be across peers with similar IP addresses.

To better understand this behavior, we take a closer look at the temporal distribution of duplicate traffic by individual users. Table I lists the top-6 edges with the most duplicate traffic. The table shows each user's total uploads, and uploads on the edge with the most repeat traffic (max edge). Each table entry also includes a temporal locality graph. Each bar stands for one day, and the height of the bar is proportional to that day's upload traffic. These results show that there is strong temporal locality present. If the same file is uploaded multiple times close in time, then it is more likely to be used as a colluding tool than legitimate sharing.

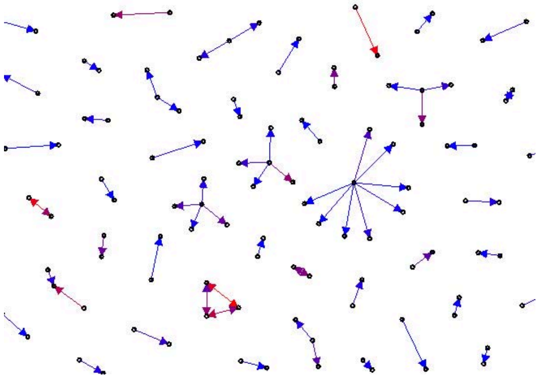


Fig. 3. Collusion link topology of 100 links with the highest ratio of duplicate transfers

The temporal locality provides strong evidence that all 5 of these peers are colluding aggressively. The maximum duplication degree is close to 43 by Cindy. David colludes with two different peers, with non-overlapping temporal behavior. Our data shows that the data transferred during these colluding sessions are generally large files or directories. For example, Alice uploaded the a 3GB DVD image 29 times.

To better understand collusion topologies, we built a visualization tool that draws edges with the highest ratios of duplicate traffic. Figure 3 gives a snapshot of the top-100 duplicate traffic links. This figure shows collusion patterns graphically. In addition to pair-wise collusions that exploit the net-gain in points from transfers, we also observe more complex 3-party and star-shaped topologies. We examine these complex collusion behaviors in detail in the next two sections.

B. Group-based Collusion Detection

We now turn our attention to mutually colluding peers. In Maze, group collusion occurs when peers exchange large amount of data among themselves to earn points. This is a consequence of the asymmetric point assignment in Maze. If

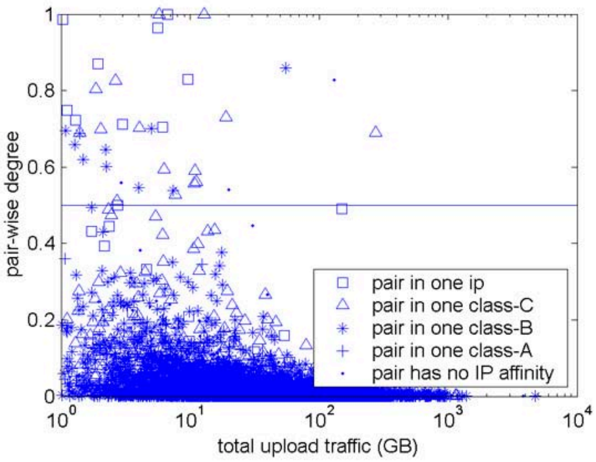


Fig. 4. Pair-wise collusion detector by the ratio of mutual upload traffic over total traffic

two peers upload 10GB data to each other, each of them will acquire at least 5000 points. The asymmetric point system was a result of extensive discussions and voting on the Maze user forum, where users wanted to encourage uploading.

Our data shows that most group collusion are pairs, and groups of three or more are rare. Intuitively, the traffic pattern for a pair-wise colluding group is where the traffic between two peers outweigh their traffic with outside peers. Note that the total traffic for a peer measured through our log segment is a rough granularity approximation of a peer's download/upload rate. We define the metric *pair-wise degree* as the ratio of total traffic between two peers to the sum of all traffic uploaded by both peers. We use this as our first group collusion detector:

Detector 2: (Pair-wise detector) high rate of mutual upload traffic compared to total uploads.

Figure 4 shows the results of applying this detector to our dataset. There are 28,000 pairs of peers with mutual uploads, each plotted as a single point with total uploads on the x-axis and pair-wise degree on the y-axis. A horizontal line is plotted for pair-wise degree equal to 0.5. Above that line are 73 pairs of peers whose pair-wise upload exceeds uploads to external peers. While it is possible for two friends to share large amounts of mutually interesting content, the highly concentrated nature of these uploads appear to indicate collusion. Regardless of whether the users intend to collude, this type of behavior results in artificially inflated point values for peers who are not contributing to the community at large.

One impediment to effective collusion of any kind is connectivity. It is tedious to transfer large amounts of data through a narrow pipe just for collusion. While pair-wise collusion across the wide-area is possible, signing on as a new user (whitewashing [18]) is an easier alternative to replenish a peer's points. Thus, we assert that pair-wise collusion requires good connectivity between peers. To verify this, we show the IP affinity of peer pairs in Figure 4 with different symbols. We see that most colluding peers have similar IP addresses, implying that the peers are likely physically close in the network and likely to be connected using a high bandwidth connection.

TABLE II
TOP-3 MUTUAL UPLOAD PAIRS. PEER 1/EXTERNAL SHOWS ALL TRAFFIC FROM PEER 1 NOT GOING TO 2.

Peer 1 external	Peer 1	Mutual upload2	Peer 2	Peer 2 external
1.7GB	Fred	24GB$\longleftrightarrow$ 23GB	Gary	5GB
23GB	Cindy	81GB$\longleftrightarrow$ 27GB	Harry	0GB
52GB	David	62GB$\longleftrightarrow$ 126GB	Alice	32GB

We take a closer look at specific examples of possible colluding peer-pairs. Table II lists the top 3 pairs ranked by pair-wise degree. Given the asymmetric point system, even the most unbalanced peer (Harry) will end up with a net point gain (after uploading 27GB and then downloading 81GB).

C. Spam Account Collusion

In our repetition-based topology in Figure 3 that showed 100 links with the highest ratio of duplicate transfers, we observed an unexpected colluding topology: the star-shaped colluding group. The center of the star gains points by uploading to many other peers without downloading anything in return. This seems contrary to behavior observed in our previous colluding patterns. The willingness of these "leaf peers" to download duplicate content from the central peer without personal gain is puzzling.

In reality, all peers in this topology are controlled by the same user, and all peers collaborate to increase the point total of the center peer. We call these leaf-peers *spam accounts*, since they are created and then discarded when they expend their initial point allocations[2]. This strategy is similar to the link spam [16] problem in search engines using page rank to sort results. One might ask why a user would prefer this strategy to performing whitewashing (restarting with new identities). One explanation is that users need to maintain one persistent primary account either for social status from higher point values or to maintain a persistent friends-list. This strategy is also efficient because it earns points much faster than pair-wise collusion for the same amount of traffic. For this to work, however, the colluder must use multiple machines.

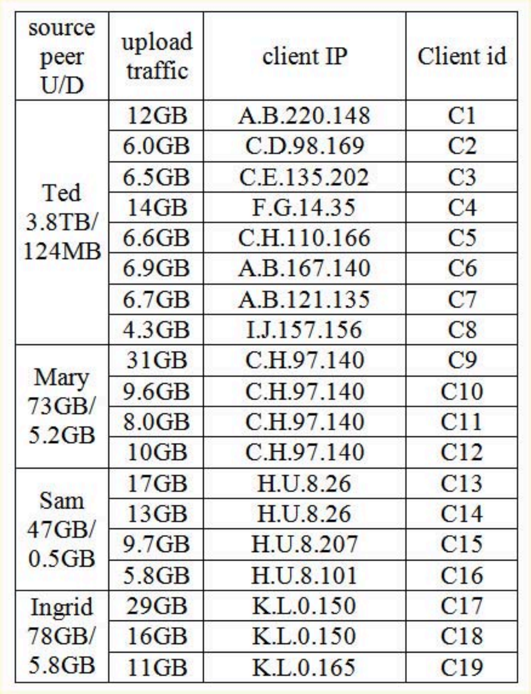
There are 4 star-shaped topologies caught by the repetition detector in Figure 3. Ted has fan-out of 8, Mary and Sam have fan-out of 4, and Ingrid has fan-out of 3. We take a closer look at them in Table III. Except for Ted's group, there is generally strong IP address proximity between the center peer and its leaf-peers. All of this indicates a high likelihood of collusion. Peer Ted, however, turns out to be the Maze user with the highest uploads of the month (3.8TB). Since its 8 edges carrying duplicate traffic shows very little IP address similarity, Ted is likely not a colluder.

While zero-cost identities are easy to generate, physically separate machines are expensive to obtain. This means spam

[2] Note that spam accounts can be produced by performing the Sybil [4] attack.

TABLE III

PEERS SUSPICIOUS OF DOING SPAM ACCOUNT COLLUDING, AS FOUND BY REPETITION DETECTOR



source peer U/D	upload traffic	client IP	Client id
Ted 3.8TB/ 124MB	12GB	A.B.220.148	C1
	6.0GB	C.D.98.169	C2
	6.5GB	C.E.135.202	C3
	14GB	F.G.14.35	C4
	6.6GB	C.H.110.166	C5
	6.9GB	A.B.167.140	C6
	6.7GB	A.B.121.135	C7
	4.3GB	I.J.157.156	C8
Mary 73GB/ 5.2GB	31GB	C.H.97.140	C9
	9.6GB	C.H.97.140	C10
	8.0GB	C.H.97.140	C11
	10GB	C.H.97.140	C12
Sam 47GB/ 0.5GB	17GB	H.U.8.26	C13
	13GB	H.U.8.26	C14
	9.7GB	H.U.8.207	C15
	5.8GB	H.U.8.101	C16
Ingrid 78GB/ 5.8GB	29GB	K.L.0.150	C17
	16GB	K.L.0.150	C18
	11GB	K.L.0.165	C19

accounts can be many in number, but are likely to reside on few machines. We define the *PM ratio* as the ratio of number of peers to number of machines to describe how densely a peer's clients are distributed across different physical machines.

Detector 3: (Spam account detector) high Peer to Machine ratio can indicate spam account collusion.

We use the method described in Section II-B to associate a peer with its machine. One problem with the PM value is the signal to noise ratio. A single upload to a random peer counts as an additional peer-machine pair, and significantly reduces the PM value. We remove these noisy values by discarding the bottom smallest uploads that, in aggregate, holds less than 20% of all upload traffic.

Figure 5 plots all peers as points with the total uploads made by the peer on the X-axis, and the peer's PM ratio on the Y-axis. Most peers have PM ratio slightly above 1 and below 2. This is statistically normal because many users perform whitewashing. However, a number of peers have exceptionally high PM ratios (up to 7). These peers are likely machines generating new user accounts to help a peer collude. Table IV lists the peers whose upload $>$ 10GB and have PM ratio $>$ 3. The temporal column shows when each client generated its peak loads of Maze traffic. Consistent temporal collisions between virtual nodes on the same machine may signal collusion.

We check for three additional properties of likely colluders. First, we use IP address proximity to infer whether these accounts have good connectivity to the center peer as expected.

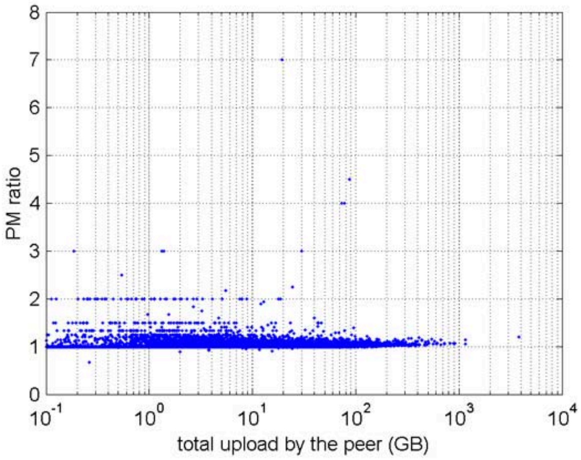


Fig. 5. Spam account detection by PM ratio

Second, we check if these accounts only download data from the center peer. Finally, we examine whether the spam accounts perform a large amount of downloads in a relatively short life-span. All of these heuristics confirm the likelihood that these peers are colluding. Most spam accounts live on the same machine; they generally download exclusively from the center peer; and they are only active for short life spans (1-2 days).

D. Upload Traffic Concentration

Pair-wise colluding and spam account colluding share one common trait: a high volume of uploads to a small number of machines. We now shift our focus from the flow among peers to among physical machines. We define the traffic concentration degree (TC degree), as the ratio of a peer's highest upload traffic to a single machine to his total upload traffic. For instance, if X uploads to 10 clients for a total of 100GB, and with 90GB going to one machine, then the TC degree of X is 0.9. The higher the TC degree, the more likely that the peer is performing either pair-wise or spam-account colluding.

Detector 4: (Traffic concentration detector) peers with exceptionally high TC degree.

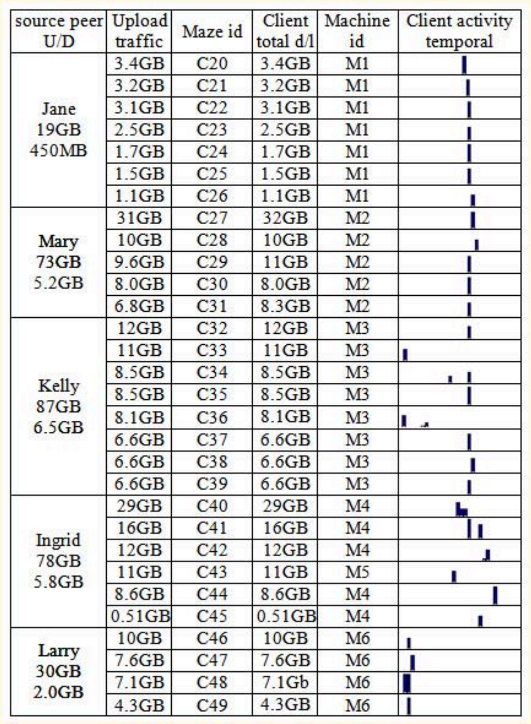
We plot the results from this detector in Figure 6 where we plot the total upload of each peer on the X-axis and its TC degree on the Y-axis. For non-colluding users, we expect an increase in upload traffic to correlate with more destinations and thus a lower TC degree. Figure 6 confirms this in our data set. Most peers who upload around 10GB have TC degrees around 10%. For heavy uploaders who upload around 1TB, the TC degree drops to about 1%.

We identify a number of potential colluders who have TC degrees significantly higher than their peers. For example, seven peers have uploaded more than 50GB and have TC degrees higher than 0.6. This means that more than 60% of the 50GB uploaded is going to a single machine. We list these

27th International Conference on Distributed Computing Systems (ICDCS'07)
0-7695-2837-3/07 $20.00 © 2007 IEEE

TABLE IV

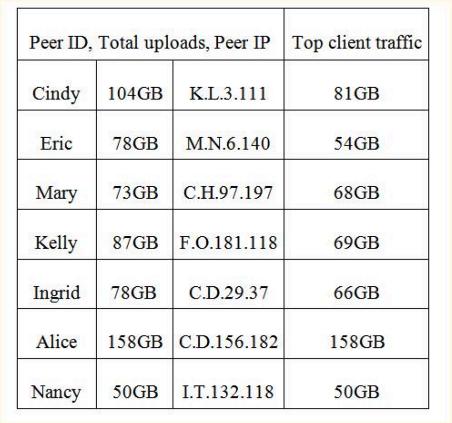
SOME TOP SPAM ACCOUNT COLLUDERS



source peer U/D	Upload traffic	Maze id	Client total d/l	Machine id	Client activity temporal
Jane 19GB 450MB	3.4GB	C20	3.4GB	M1	
	3.2GB	C21	3.2GB	M1	
	3.1GB	C22	3.1GB	M1	
	2.5GB	C23	2.5GB	M1	
	1.7GB	C24	1.7GB	M1	
	1.5GB	C25	1.5GB	M1	
	1.1GB	C26	1.1GB	M1	
Mary 73GB 5.2GB	31GB	C27	32GB	M2	
	10GB	C28	10GB	M2	
	9.6GB	C29	11GB	M2	
	8.0GB	C30	8.0GB	M2	
	6.8GB	C31	8.3GB	M2	
Kelly 87GB 6.5GB	12GB	C32	12GB	M3	
	11GB	C33	11GB	M3	
	8.5GB	C34	8.5GB	M3	
	8.5GB	C35	8.5GB	M3	
	8.1GB	C36	8.1GB	M3	
	6.6GB	C37	6.6GB	M3	
	6.6GB	C38	6.6GB	M3	
	6.6GB	C39	6.6GB	M3	
Ingrid 78GB 5.8GB	29GB	C40	29GB	M4	
	16GB	C41	16GB	M4	
	12GB	C42	12GB	M4	
	11GB	C43	11GB	M5	
	8.6GB	C44	8.6GB	M4	
	0.51GB	C45	0.51GB	M4	
Larry 30GB 2.0GB	10GB	C46	10GB	M6	
	7.6GB	C47	7.6GB	M6	
	7.1GB	C48	7.1Gb	M6	
	4.3GB	C49	4.3GB	M6	

TABLE V

TOP 7 COLLUDERS DETECTED BY TC DETECTOR



Peer ID, Total uploads, Peer IP			Top client traffic
Cindy	104GB	K.L.3.111	81GB
Eric	78GB	M.N.6.140	54GB
Mary	73GB	C.H.97.197	68GB
Kelly	87GB	F.O.181.118	69GB
Ingrid	78GB	C.D.29.37	66GB
Alice	158GB	C.D.156.182	158GB
Nancy	50GB	I.T.132.118	50GB

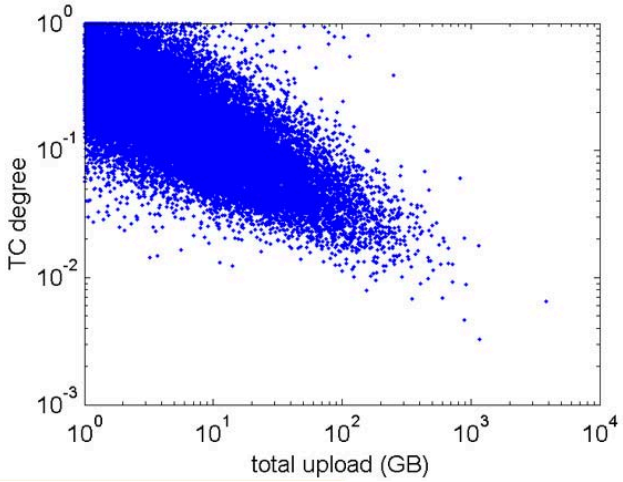
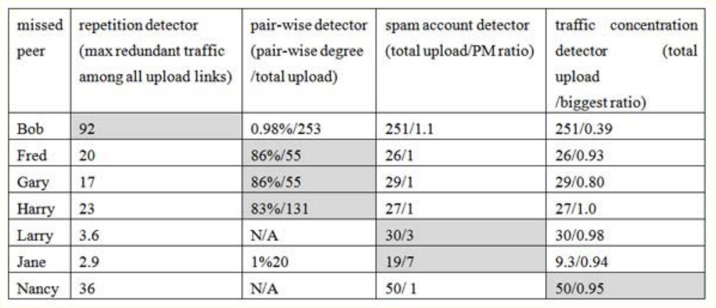


Fig. 6. Upload traffic concentration detector result

TABLE VI

SEVEN TOP COLLUDERS AND HOW OUR DETECTORS HAVE FOUND AND MISSED THEM



missed peer	repetition detector (max redundant traffic among all upload links)	pair-wise detector (pair-wise degree /total upload)	spam account detector (total upload/PM ratio)	traffic concentration detector (total upload /biggest ratio)
Bob	92	0.98%/253	251/1.1	251/0.39
Fred	20	86%/55	26/1	26/0.93
Gary	17	86%/55	29/1	29/0.80
Harry	23	83%/131	27/1	27/1.0
Larry	3.6	N/A	30/3	30/0.98
Jane	2.9	1%20	19/7	9.3/0.94
Nancy	36	N/A	50/ 1	50/0.95

peers and their traffic in Table V. Six of these have been detected by previous detectors. The pair-wise detector missed the new peer Nancy because it has no pair-wise traffic with any other peer. The spam account detector missed Nancy because it mainly uploads to only one peer and its PM ratio is close to one. It turns out that it ranks #7 by the repetition detector (we listed only the top 6 in Section 3.1).

E. Detectors Compared

After presenting four different collusion detectors, we summarize the top colluders discussed in earlier sections in Figure 7, and graphically show how they were detected by each of our four detectors. Table VI lists the top seven colluders according to their total upload traffic. The detectors responsible for detecting each colluder is shown in shaded cells. Our first observation is that spam-account and pair-wise colluders do not overlap. This is logical, because the two detectors are designed specifically with these two patterns in mind. While colluders can engage in both activities simultaneously to evade potential detectors, the fact that spam-account colluding is more "cost effective" and that these detectors were applied to logs after the fact, leads us to believe that this does not happen in practice.

As discussed earlier, the traffic concentration detector relies on the observation that colluders generally control relatively few machines. Thus it examines how a peer's upload traffic is spread across its partners. Figure 7 shows that this detector detects the majority of both spam-account and pair-wise colluders. We now examine the colluders that TC degree failed to detect. With the exception of "Bob," who was missed due to a low TC degree (0.39), the other colluders it missed all have high TC degrees (>0.8), and avoided detection only because their total upload traffic was low. Lowering our traffic

TABLE VII

SUMMARY OF STRENGTHS AND WEAKNESSES OF COLLUSION DETECTORS.

Detector	Heuristic	Strength	Weakness
Colluders	Small colluding working set	General	Cannot detect randomized colluding group
Pair-wise	Uploads: pair-wise > external	Accurate	Limited to pair-wise topologies
Spam	One uploads to many	Accurate	Limited to spam collusion topologies
Traffic Concentration	Colluder controls few hosts	General	Difficult to optimize parameters

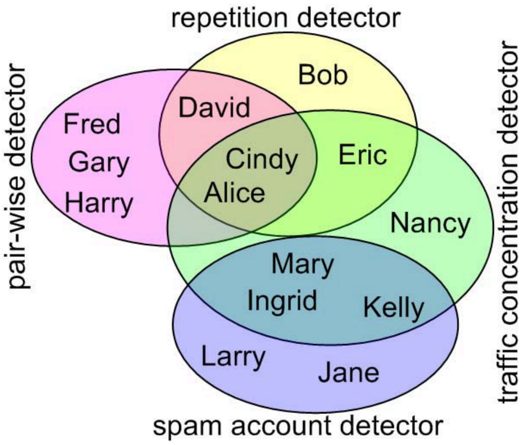


Fig. 7. Venn diagram of collusion detectors

```
$\vec{t}^{(0)} = \vec{p}$;
repeat
    $\vec{t}^{(k+1)} = C^T \vec{t}^{(k)}$;
    $\vec{t}^{(k+1)} = (1-a)\vec{t}^{(k+1)} + a\vec{p}$;
    $\delta = ||t^{(k+1)} - t^{(k)}||$;
until $\delta < \epsilon$;
```

Fig. 8. The basic EigenTrust algorithm for computing trust.

threshold from the current value of 50GB would allow us to detect all of our colluders, showing that the TC detector is in fact quite effective. However, because we still cannot guarantee perfect results when mapping peers to specific machines (in the presence of DHCP and NATs), we cannot yet deploy the TC detector as an online collusion detector. Finally, we are still investigating how to choose reasonable traffic thresholds to avoid false positives.

Table VI shows that the repetition detector misses six colluders. All six peers have too small redundant traffic on a single link to be noticed. For example, peers Larry and Jane have a lot of collusion traffic, but their traffic is scattered across multiple upload links. The repetition can only be found if we aggregate multiple links together (Figure 3). The pair-wise detector also missed four colluders. Two among the four colluders have little number of mutual upload with other peers. The other two have no mutual upload with any peer at all. Spam account detector missed five colluders. All the five colluders evaded the spam account detector because they have very low PM ratios.

Figure 7 shows that the repetition detector also works quite well. However, the reason that it works at all is because the current version of Maze has no explicit defense mechanism against collusion. This detector can easily be circumvented by a colluder if it simply modifies the content slightly, even by just flipping one single bit. Also, differentiating legitimate repeated downloads from colluders will be a challenging task. For example, peers could lose their local cache and be required to repeat previous downloads. We have used this detector in the study to lead the ways to other more robust detectors, taking advantage of the very fact that colluders today do not bother to cover their tracks by randomizing their colluding working set. We summarized the four detectors in Table VII.

IV. EIGENTRUST AND COLLUSION

Our access to a complete view of all transactions in Maze gives us a unique opportunity to evaluate how proposed reputation systems from research would perform on real world systems. In this section, we will evaluate EigenTrust [6], a well-known reputation system that generatess a global ranking. The EigenTrust ranking can be used for both reputation management [6] (clients choose trustworthy download sources), and free-rider detection [7] (uploaders choose trustworthy clients). Ideally, the ranking algorithm would assign low scores to malicious colluders.

A. An Overview of EigenTrust

We first give a high level description of the EigenTrust system [6]. EigenTrust calculates global trust values for all peers based on Power iteration in peer-to-peer file-sharing systems. The algorithm is similar to the PageRank algorithm. First, peer i assigns peer j trust values C_{ij} based on its downloading experience from j. Trust values for all j are normalized locally by each peer i. We obtain from this a matrix C containing a measure of trust across all peer pair in the network. The trust vector t is defined as the left principal eigenvector of C. The component t_i is called the EigenRank of peer i, and represents the peer's global reputation.

The basic algorithm can be further improved to enhance its robustness against malicious users. This is done by incorporating the notion of pre-trusted peers in the set P. So, for peer i, we define $p_i = 1/|P|$ if $i \in P$, and p_i = 0 otherwise. The algorithm is summarized in Figure 8. Parameter α is a constant less than 1 that is used to control the level of trust

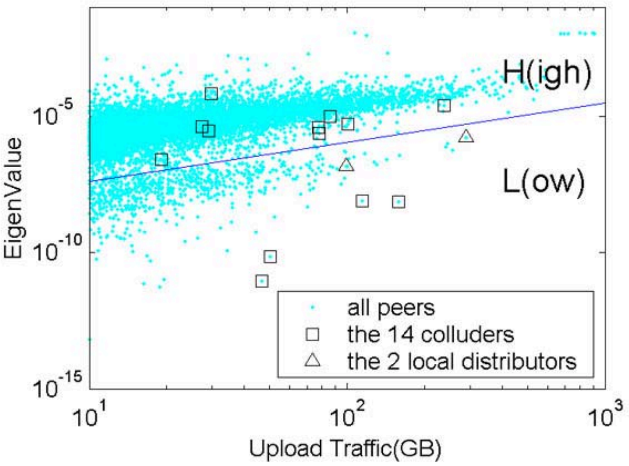


Fig. 9. The EigenRank of peers

each peer places on the pre-trusted peers. A higher value of α implies more confidence on the pre-trusted peers.

B. Applying EigenTrust to Maze

We map the EigenTrust algorithm to Maze system as follows. First, we define the trust value c_{ij} be proportional to the total traffic peer i downloads from peer j during the log period. We then normalize the local trust value c_{ij} such that: $\sum_{j=1}^{N} c_{ij} = 1$. Next, we select 10 pre-trusted peers from the Maze population by choosing ten well-known users who we have directly interacted with on the Maze user forum (*i.e.* $|P| = 10$). This model gives us the matrix C and the set of pre-trusted peers P. Finally, we set α to a reasonable value of 0.1.

C. Experimental Results

Figure 9 shows the EigenTrust values for the 9568 peers whose upload traffic total more than 10GB. We can make two interesting observations. First, generally speaking, the more uploads a peer has, the higher its score will be. Thus, if Maze had no colluders or whitewashers, its primitive point system should work well. Second, peers are spread in two noticeable bands. We drew a line and partitioned peers into two (High and Low) regions, as shown in Figure 9. Out of roughly 9600 peers, 551 are in the Low region. If we focus on peers with similar amounts of upload traffic, reputation values of peers in the High region are far higher than those in the Low region (about 10^3 times). This seems to imply that peers in the Low region are misbehaving colluders.

To test this assertion, we label the positions of the fourteen colluders detected from earlier sections using square symbols (see Figure 9). Peer Mary is absent because its EigenTrust value is 0 and outside the scope of the Y-axis. The result is surprising. While some colluders have low scores in the Low region, many others have higher scores and reside in the High region. Therefore, there is no clear cause for the delineation between the two regions of Eigenvalues.

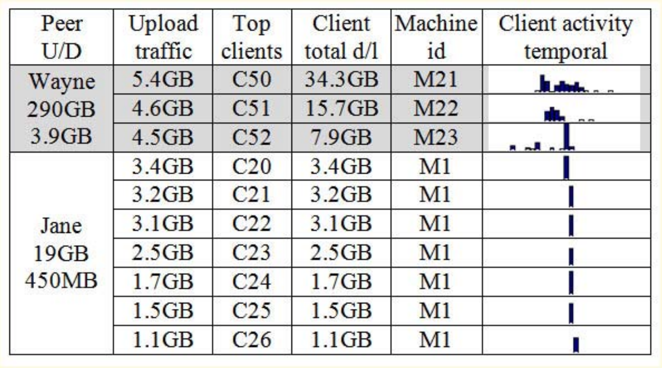
TABLE VIII

CLIENT IP DISTRIBUTION OF REGIONS H AND L

Region	Average # of distinct IPs in clients	Avg. # of Class-B spaces with clients
H	299.5	59.38
L	98.323	2.18

TABLE IX

A COMPARISON OF A NON-COLLUDING REGION-L PEER (WAYNE) WITH A REGION-H SPAM-ACCOUNT COLLUDER (JANE)



Peer U/D	Upload traffic	Top clients	Client total d/l	Machine id	Client activity temporal
Wayne 290GB 3.9GB	5.4GB	C50	34.3GB	M21	
	4.6GB	C51	15.7GB	M22	
	4.5GB	C52	7.9GB	M23	
Jane 19GB 450MB	3.4GB	C20	3.4GB	M1	
	3.2GB	C21	3.2GB	M1	
	3.1GB	C22	3.1GB	M1	
	2.5GB	C23	2.5GB	M1	
	1.7GB	C24	1.7GB	M1	
	1.5GB	C25	1.5GB	M1	
	1.1GB	C26	1.1GB	M1	

In EigenTrust, a peer's reputation depends on the reputations of its clients: if the clients have lower reputations, then this peer suffers as well. Therefore, we examine the respective client groups of these peers to find the cause for their difference in trust values. After analyzing the data, we find that there is a significant difference in the IP address distribution of the clients of the High (H) region and the Low (L) region peers. The data is shown in Table VIII. On average, region-H peers upload to about 300 distinct IP addresses scattered across 60 class-B spaces. This means that each class-B space contains on average 5 IP addresses used by clients of region-H peers. On the other hand, region-L peers upload to 98 distinct IP addresses scattered across 2.2 class-B spaces. Thus, each class-B space contains about 45 IP addresses used by clients of region-L peers. Therefore, the key difference is that region-H peers have more clients than region-L peers, and they are more widely spread geographically. Region-L peers seem to serve the role of "local distributors" that disseminate data only to nearby peers.

We take a closer look at well-behaved peers with low trust values and colluding peers with high trust values. Table IX lists one of the region-L peers (Wayne) and compares it with a spam-account colluder (Jane) we found earlier. Wayne is in region-L, whereas Jane is in region-H (the reason that Jane ranks high will be discussed shortly). Peer Wayne's 722 clients reside on 614 different machines, all of which have temporal activities that are vastly different from the colluder; we show only its three top clients. In contrast, Jane's download activities are well synchronized in time and strongly indicative of collusion.

A closer look at Wayne's uploading history reveals that

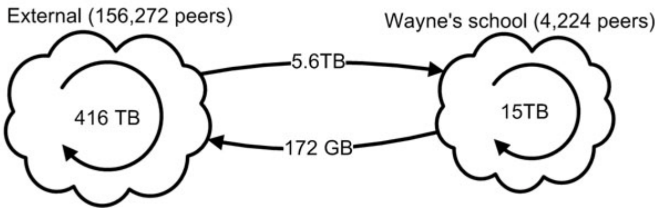


Fig. 10. Wayne's school is a satellite cluster

TABLE X
A PRE-TRUSTED PEER HELPS COLLUDER LARRY

Total upload by Larry	Total collusion	Uploads to Ted	Ted's total downloads
29.7GB	29GB	734KB	124MB

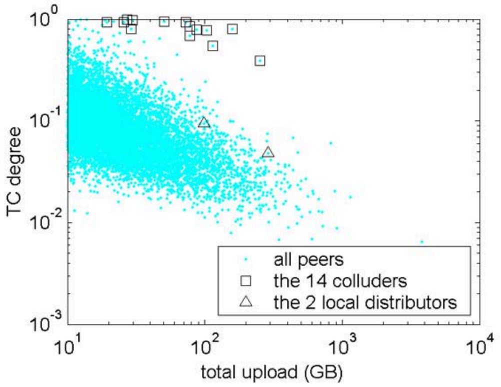


Fig. 11. The TC plot with local distributors and colluders

many of its clients are also region-L peers. Thus, we speculate that due to the nature of good network connectivity, peers in a subnet tend to cluster together in their downloading activities. Despite the fact that there is large volume of traffic within the cluster, because the amount of external traffic is low, EigenTrust compounds the lower scores in the group through its transitive trust computation, in effect treating these peers as part of a big misbehaving group. We query Wayne's IP address in the APNIC whois database, and find that Wayne belongs to a university, and most of his clients are from the same university. We calculate the internal and external traffic of this school and show the results in Figure 10.

The total traffic consumed by Wayne's school is the sum of its internal traffic (15TB) and its download from external sites (5.6TB). On average, a peer in Wayne's school is responsible for 4.9GB of traffic during the log period. For the same period, a peer not in Wayne's school has an average of 2.7GB of traffic. While the difference is noticeable, it is not statistically significant. The key problem is that Wayne's school collectively uploads (172GB) far less than it downloads (5.6TB): the upload volume is only 3% of the download volume. Therefore, everyone in this cluster (Wayne's school) is punished by the EigenTrust algorithm, including Wayne. While it is interesting that EigenTrust has helped to identify this satellite cluster with asymmetric traffic flow, the scores assigned to individual peers do not seem justified. A non-colluding region-L peer such as Wayne has contributed to a great number of other peers, but nontheless receives a low score because of his client peers. It may be fair for peers outside of Wayne's school to treat Wayne as colluder, but peers inside his school should not. In this instance, a global ranking can clearly lead to a non-ideal representation of a peer's trust value.

We now try to understand how certain colluders have obtained such high trust values in EigenTrust. We take as example a colluder Larry, who has a much higher trust value than other normal peers with comparable uploads. Figure 9 shows many colluders located in region-H. While it is difficult to determine all contributing factors, one possibility is that pre-trusted peers may have unintentionally helped colluders elevate their trust values. Table X presents a case where a pre-trusted peer Ted helps colluder Larry. Larry uploads 29.7GB, mostly of which are uploads to colluding partners. Pre-trusted peer Ted has a total of 3.8TB of uploads, and it has also downloaded 124MB data. Out of these downloads, 734KB or only 0.59% are from Larry. This tiny amount of traffic raises Larry's trust value by nearly a factor of 10 (from 8.2e-6 to 6.6e-5). Removing this upload would have dropped the colluder's ranking from 334 to 1905. In addition, Larry has a small amount of uploads to other reputable peers. If we remove its top 200MB uploads to "celebrity peers," it drops into region L (trust value drops to 7.4e-8). Decreasing the value of α to place less trust on the pre-trust peers produces a similar effect. These findings seem to suggest another vulnerability of the EigenTrust algorithm.

Obtaining an endorsement from a highly reputed node is expensive in the context of web spamming, but is not difficult in file-sharing systems like Maze, since the primary factor in choosing downloading target is content availability. Such a transaction significantly boosts trust values of colluders. EigenTrust is highly sensitive to this effect. Additionally, since most colluders do not have downloads from highly reputed nodes, transactions between a single colluder and a highly reputed node would work to elevant the rankings of all colluders.

To put things into perspective, we conclude this section by redrawing the results of using the traffic-concentration detector in Figure 11, now annotated with the same colluders and local distributors as we did in Figure 9. Note that unlike EigenTrust, a lower value of TC degree means that a peer is not only contributing, but that its contribution flows are physically - not logically - diverse. It is interesting to see that all the 14 sample colluders have high TC degree, whereas the two sample local distributors have low TC degree. Thus, for the Maze system at least, the TC degree metric seems to be a more robust enhancement to our point-based trust system.

V. RELATED WORK

According to the taxonomy proposed in [5], incentive mechanisms can be categorized into those using private history, shared history, or subjective shared history. The choking algorithm in BitTorrent is a type of private history based on Tit-for-Tat. Its robustness and ability to deal with free-riders has been proven in practice [2]. However, systems based on private history generally do not scale well. In a large P2P network, peers will interact with a large amount of peers, most of which are new faces, and they will only interact once [3]. This limits BitTorrent to generally small groups in individual download sessions.

Many shared history solutions have been proposed to overcome this drawback. They can be generally categorized into two types [10]: virtual currency based and reputation based, *e.g.* Mojo Nation [15] or Free Haven [3]. Shared history introduces the collusion problem, where colluders can use forged shared history to increase the ranking of colluders [5]. One approach is presented in the Stamp algorithm [10], where peers issue stamps to use as virtual currency, and the value of each peer's stamps is maintained by exchange rates that act as reputation values.

A more general solution is to use subjective shared history. The maxflow algorithm [5] estimates the services that the requester has provided to the uploader in the past, whether directly or indirectly. The comparison is done from the uploader's point of view, and therefore is "subjective." The algorithm is computationally expensive and does not offer any global rankings of peer reputations. Our experience with Maze has shown that mechanisms like the "altruistic list" provide strong incentives for cooperation, even though it can encourage collusion for popularity's sake.

Finally, our results show that the collusion patterns found in Maze have analogous counterparts in web spam attacks. For example, link spam collusion [16] is a combination of our spam account collusion and group collusion. Other similarities exist. For example, TrustRank is an algorithm that is more spam-resilient and similar to EigenTrust, both of which are derived from the page ranking algorithm [1]. It would be worthwhile to apply some of their results to collusion in P2P file-sharing systems.

VI. CONCLUSION

Our work presents first-hand empirical analysis of colluding behavior in a real peer-to-peer file sharing system. With total access to the Maze network, we analyzed a complete user and traffic log segment representing all transactions during the course of a month. From our observations of collusion behavior, we build four different types of collusion detectors for file-sharing networks. While obtaining definitive proof of intent to collude is difficult, and our detectors can produce false-positives for unexpected user behavior, application of our detectors has provided substantial evidence of existing collusion-like behavior in Maze.

We also apply the popular EigenTrust reputation system to our data set, and compare the results to our knowledge of existing colluders. Using our lessons from this study, we are developing incentive systems that will provide stronger resistance against observed user collusion behavior [9].

Note that while Maze is a centralized system, most of our detectors can be implemented using distributed mechanisms and embedded inside client software. We believe that if client software can prevent user tampering of data, our detectors can quantify collusion behavior on an "online" basis. The question of whether knowledge of collusion detection mechanisms changes user behavior remains to be seen. Finally, the collusion patterns we observe are likely to occur in any system without conservation of points. Outside of BitTorrent's Tit-for-Tat scheme, Maze is one of the few peer-to-peer systems with an incentive system, and our lessons can guide the design and deployment of future distributed incentive schemes.

ACKNOWLEDGMENTS

Work at Peking University is supported by the National Natural Science Foundation of China under Grant No. 60673183 and the National Grand Fundamental Research 973 Program of China under Grant No. 2004CB318204. Ben Zhao is supported by NSF CAREER Grant No. 0546216.

REFERENCES

[1] BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual web search engine. In *Proc. of WWW* (Brisbane, Australia, April 1998).

[2] COHEN, B. Incentives build robustness in bittorrent. In *Proc. of P2P-Econ* (June 2003).

[3] DINGLEDINE, R., FREEDMAN, M. J., AND MOLNAR, D. The free haven project: Distributed anonymous storage service. In *Proc. of WDIAU* (July 2000).

[4] DOUCEUR, J. R. The Sybil attack. In *Proc. of IPTPS* (March 2002).

[5] FELDMAN, M., LAI, K., STOICA, I., AND CHUANG, J. Robust incentive techniques for peer-to-peer networks. In *Proc. of EC* (May 2004).

[6] FETTERLY, D., MANASSE, M., AND NAJORK, M. Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages. In *Proc. of WebDB* (June 2004).

[7] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in p2p networks. In *Proc. of WWW* (May 2003).

[8] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. Incentives for combating freeriding on p2p networks. In *Proc. of EuroPar* (June 2003).

[9] LIAN, Q., PENG, Y., YANG, M., ZHANG, Z., DAI, Y., AND LI, X. Robust incentives via multi-level tit-for-tat. In *Proc. of IPTPS* (Santa Barbara, CA, Feb. 2006).

[10] MORETON, T., AND TWIGG, A. Trading in trust, tokens, and stamps. In *Proc. of P2PEcon* (Berkeley, CA, June 2003).

[11] NGAN, T.-W. J., WALLACH, D. S., AND DRUSCHEL, P. Enforcing fair sharing of peer-to-peer resources. In *Proc. of IPTPS* (Berkeley, CA, Feb. 2003).

[12] SAROIU, S., GUMMADI, K. P., DUNN, R. J., GRIBBLE, S. D., AND LEVY, H. M. An analysis of internet content delivery systems. In *Proc. of OSDI* (December 2002), ACM, pp. 315–328.

[13] SAROIU, S., GUMMADI, P. K., AND GRIBBLE, S. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN* (January 2002).

[14] T-net. `http://e.pku.edu.cn`.

[15] WILCOX-O'HEARN, B. Experiences deploying a large-scale emergent network. In *Proc. of IPTPS* (Cambridge, MAa, Feb. 2002).

[16] WU, B., AND DAVISON, B. D. Identifying link farm spam pages. In *Proc. of WWW* (Chiba, Japan, May 2005).

[17] YANG, M., CHEN, H., ZHAO, B. Y., DAI, Y., AND ZHANG, Z. Deployment of a large-scale peer-to-peer social network. In *Proc. of WORLDS* (December 2004), USENIX.

[18] YANG, M., ZHANG, Z., LI, X., AND DAI, Y. An empirical study of free-riding behavior in the maze p2p file-sharing system. In *Proc. of IPTPS* (Ithaca, NY, Feb. 2005).