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

Qiao Lian[†], Zheng Zhang[†], Mao Yang [†§], Ben Y. Zhao[‡], Yafei Dai[§], Xiaoming Li[§]
[†] *Microsoft Research Asia, Beijing, China*
[‡] *U. C. Santa Barbara, Santa Barbara, CA, USA*
[§] *Peking University, Beijing, China*

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 collusion, our lack of understanding of user collusion in existing systems makes evaluating such mechanisms difficult. 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 point-based incentive policy. We search for the existence of colluding behavior by examining complete user logs of the entire system, and use a set of collusion detectors to identify several major collusion patterns. In addition, we evaluate how proposed reputation policies would perform in Maze, and identify reasons why they might miss their objectives. Our results are generally applicable to large-scale peer-to-peer systems, and can help guide the design of more robust incentive schemes..

1. Introduction

File-sharing networks such as Kazaa and Gnutella have popularized the peer-to-peer (P2P) resource sharing model. In these large and distributed networks, selfish users often "free-ride", or act in their self interests to exploit the system. Numerous research efforts have focused on the use of incentive systems to encourage sharing among users. Despite the effectiveness of these incentive systems, they are generally vulnerable to variants of the Sybil Attack [5]. In a Sybil attack, users take advantage of the zero-cost nature of online identities to create multiple identities. These online identities can then actively collude to cheat the incentive system.

While incentive systems are designed to discourage or prevent collusion [12], very little is known about how users actually collude in real systems, making collusion prevention difficult. This information is extremely difficult to gather, simply because it requires a very complete view of the inner workings of the network. Most measurements to date are performed by logging traffic at the edge nodes while performing queries or membership operations [14][15].

In this paper, we present measurements of actual user collusion activity in the Maze peer-to-peer file-sharing network. 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, the Maze software is fully controlled by our research team, making it possible to deploy and embed measurement code inside clients. Second, Maze's centralized architecture means we have access to all control and query traffic. Maze uses a simple incentive system based on a points system that increases with uploads and decreases with downloads. A trusted central server audits file transfers and adjusting user points accordingly.

For our purposes, we define *collusion* as collaborative activity within groups of users that gives to group members benefits they would not be able to gain as individuals. We note that it is impossible to determine users' intent. Our study focuses purely on observable action patterns that produce results similar to those produced by colluding users. In addition, our work is a first step towards understanding collusion behavior. Quantifying all forms of collusion is a topic to be addressed in ongoing work. It is also difficult to determine definitively whether multiple identities belong to the same person. Issues such as DHCP, NATs and mobile laptops prevent us from reliably detecting the use of multiple virtual identities.

This paper makes three key contributions. First, we gathered a month-long log of Maze that includes details of every file transfer performed in the system. For each transfer, we record the end peers, the file ID, transfer size and other data. We analyze this dataset for user collusion behavior. Second, we derive several *collusion detectors* based on our analysis of these logs, and use them to quantify the number and types of collusion in Maze. Finally, we apply the EigenTrust reputation system [8][9] to our dataset, and show that colluding users are difficult to detect using traditional reputation systems. 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 [4][8][9], but also show that certain types of collusion are difficult to detect and deter.

The rest of the paper is organized as follows. Section 2 gives a brief description of Maze and the data used for this study. Then, Section 3 describes each of our collusion detectors in detail, along with their results when applied to the Maze dataset. Section 4 compares and contrasts the detectors. Then in Section 5, we apply the EigenTrust algorithm to the Maze dataset and analyze the results. Finally, we discuss related work in Section 6 and conclude in Section 7.

2. The Maze Peer-to-Peer system

Before we present our analysis of the Maze logs, we begin by giving background information on the Maze system. The Maze file-sharing system was originally deployed to address issues of data location and load-balancing on FTP servers as part of the T-Net Web search project [20]. As T-Net became popular, its limited number of FTP servers led to significantly degraded performance. Maze provided a way to distribute content without incurring further infrastructure costs.

At login, Maze peers[1] authenticate to a central server, and upload an index of its locally available shared files. The central server maintains heartbeats with all online peers, and its index supports full-text queries across all of their shared files. In addition to searching the central index, users can browse three peer lists: 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 form a continuously adaptive social network. In contrast, the server provides each peer with a neighborhood-list of other peers sharing the same B-class IP address. These provide a list of local hosts with likely high-bandwidth, low latency links to the local host. Finally, the altruistic list is a collection of peers with the highest "Maze points" provided by the server. These peers 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 [18].

[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*.

A peer can recursively browse the contents of the Maze directories of any level of these lists, and initiate downloads when they find interesting content. 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.

Peers download from each other directly in a P2P fashion. Peers behind NATs can only connect to non-firewalled peers. Peers perform "swarm downloads" whenever possible by simultaneously retrieving different file chunks from multiple users, with priority given to peers that share IP address prefix. After each transaction, peers involved report to the central server, which adjusts their points accordingly.

2.1 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:

$$requestTime - 3 \cdot \log_{10} P$$

where P is the requestor's point total. Frequent uploads provide peers with higher points and faster downloads. While simple, this system faces two issues. First, do we keep the assignment of points as a zero-sum game, where the points lost by one peer are gained by the other? Enforcing such a policy imposes hardships on peers with slow links and those who hold many unpopular items. As a result, the Maze community discussed and voted for a rule which gives uploading more points than downloading in order to encourage uploading. This enables more flexibility, but has the side effect of allowing two interactive peers to create a net gain in points after mutual interaction. The other issue is bootstrapping points for new peers. Peers must have sufficient initial points to download content for it to share later. In the current system, Maze allows a peer to download > 1GB of data before its downloads are throttled at a rate of 300 kbps.

1. New users are initialized with 4096 points.
2. Uploads: +1.5 points per MB uploaded
3. 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:
    - Each peer orders download requests by T = requestTime - 3log*P*, where P is the requester's point total.
    - Users with P < 512 are limited to 200Kb/s.

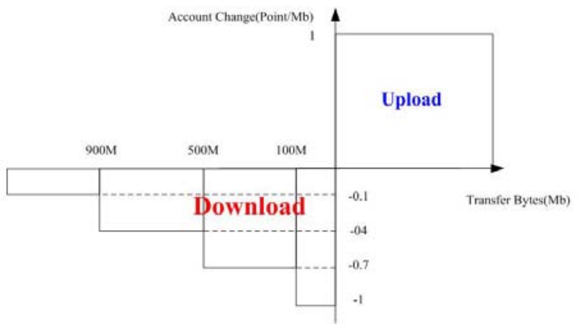


Fig. 1. The Maze point system

Maze uses service differentiation to reward and punish users for their behavior. The point system gives downloading preference to users with high scores. These users add to their request time a negative offset whose magnitude grows logarithmically with their score. The parameters of the credit system are designed to optimize for large downloads in a user population where the majority of users have access to high-bandwidth links. Since the majority of bytes exchanged on Maze are part of large multimedia files, download point deductions are graduated to weigh less heavily on extremely large files. For instance, a user can use up her initial points by downloading 4GB worth of 1MB-sized files, 5.2GB worth of 400MB-sized files, or 6.8GB of 800MB files.

Note that Maze policies award at least 50% more points for uploading than downloading. This system rewards uploaders and encourages additional participants to join the system. We recognize that this allows a net points gain as a result of a symmetric operation. This property is likely a source of user collusion. We discuss evidence of this later in Section 3.1. While the total points in the system will increase over time, we have yet to observe any negative impact on the overall system.

2.2 Data collection

While continuous logs of Maze traffic are maintained, we perform our analysis on a log segment gathered during the span of one-month period from 2/19/05 to 3/24/05. During this period, more than 161,000 active users participated in a total of more than 32 million file transfers. Data traffic totaled up to more than 437 Terabytes.

The data gathered for this study consists of a collection of user points during this month and the detailed traffic log. When two peers report the completion of a file transfer to the server, our log keeps only the data from the uploading peer. Each traffic log entry contains the following: *uploading peer-id, downloading peer-id, log upload time (server), transfer start time (source), transfer end time (source), bytes transferred, file size, downloader IP, file md5 hash, and full file path*. The bytes transferred can be different from the file size if the transfer was interrupted, or if the transfer is sourcing from multiple peers.

Note that the log only records the downloading peer's IP address as seen from the uploading peer. Thus, if both peers are behind the same firewall, the IP address of the downloader can be an internal IP (e.g. 192.168.*.*, 10.*.*.*). Otherwise it will be either a public IP or the NAT address, depending on whether the downloader has a public IP or not. A single machine can thus be tracked as a list of different IP addresses, including changes due to DHCP and host mobility in the case of laptops. In analyzing the colluding behavior, we frequently need to infer the network vicinity of the peers based on IP. To simplify this, we use the peer's most frequently used IP address.

As we discuss later, we also need to associate online identities with the physical machine the peer uses. We began by to using the hash of the hard drive serial number, first reported when the client logs onto Maze. We later discovered, however, 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.

We anonymize our logs to protect the privacy of Maze users. User identities are hashed into random strings. In this paper, we refer to distinct users using common names from a dictionary (e.g. Alice and Bob), and random alphabetic letters to represent 8-bit blocks of an IP address (e.g. C.H.97.140).

3. 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, and later summarize their strengths and weaknesses.

3.1 Repetition-based collusion detection

Our first attempt starts by drawing a crude picture of colluding activities in Maze by looking at how users use uploading to generate Maze points. Given that Maze does not explicitly guard against collusion, and 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.

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

We examine all transactions recorded in the one-month 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 contain duplicate files in the transfer traffic. This accounts for roughly 4.9% of all peer relationships. We define *duplication degree* be the ratio of total upload traffic in bytes over the size of the unique data in bytes. A high duplication degree means a low proportion of all traffic across the link is unique.

We plot the duplication degree of all edges against their total upload traffic as a scatter plot in. For duplication degrees of 5, 10 and 20, there are 890, 148, 27 edges with duplication degree greater than each respective threshold. Given that this data is generated from activities performed over the span of a month, it is highly likely that a good fraction of these peers are actively colluding. We also note that colluders are likely to use nearby machines to perform the transfers. Such network locality will maximize 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. The IP affinity data also confirms that edges with larger amount of repeat traffic are more likely to be across peers with similar IP addresses.

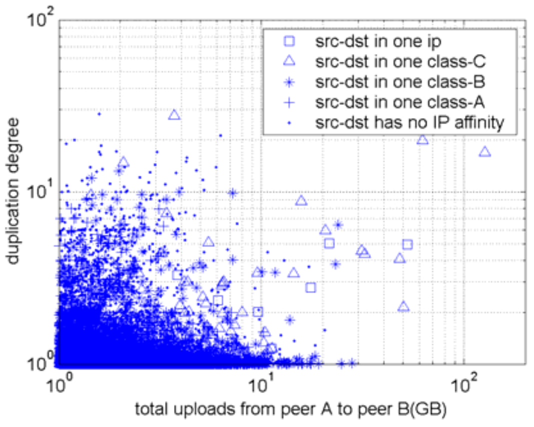


Figure 2: Duplication degree in uploading from peer A to peer B

To better understand this behavior, we take a closer look at the temporal distribution of duplicate traffic by individual users. Table 1 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.

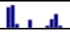
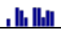
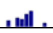
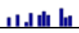
Src ID, U/D (GB)		Unique data on max edge	Total traffic on max edge	Temporal locality (x: date, y: upload)
Alice	158/76	7.5 GB	126 GB	
Bob	251/12	6.0 GB	98 GB	
Cindy	104/31	1.9 GB	81 GB	
David	114/149	3.1 GB	62 GB	
David	114/149	10.1 GB	52 GB	
Eric	78/18	7.4 GB	44 GB	

Table 1: Top 6 edges with the most redundant traffic

The temporal locality provides strong evidence that all 5 of these peers are colluding aggressively. The maximum duplication degree is close to 43 by peer Cindy. Peer 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, peer Alice uploaded the MSDN DVD image (~3GB) repeatedly for 29 times.

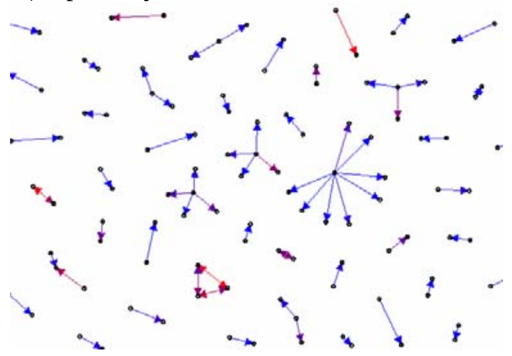


Figure 3: Collusion link topology of 100 links with the highest ratio of duplicate transfers

Locating nodes that carry large amounts of duplicate traffic has given us a starting point in detecting collusion. The next step is to better understand collusion topologies. For this, we built a visualization tool that draws edges with highest ratio of duplicate traffic. Figure 3 gives a snapshot of the top-100 duplicate traffic links. This figure shows graphically the collusion patterns. There are pair-wise collusions, which is a result of the asymmetric point system (upload earns more than download for the same amount of bytes). There is also a more sophisticated 3-party topology. Interestingly enough, it also shows a number of star-shaped topologies, which is not what we have expected. The following two sections deal with these two kinds of collusion in greater detail.

3.2 Group-based collusion detection

After examining collusion based on duplicate traffic, we turn our attention to mutually colluding peers. In Maze, group collusion occurs when peers exchanging large amount of data among themselves to earn points. This is a consequence of the asymmetric point assignment in Maze. If two peers upload 10GB data to each other, each of them will acquire at least 5 thousand points. The asymmetric point system was a result of extensive discussions and voting on the Maze forum, where users wanted to encourage uploading more than downloading. Through this and other studies, we are trying to quantify the impact of this incentive policy.

Examining topologies in Figure 3 shows three pair-wise colluding groups and one 3-party colluding group. We refer to them as (Fred, Gary), (Olga, Pam), (Harry, Cindy), and (David, Alice, Quincy). Our data shows that most group collusions are pair-wise groups, and groups of three or more are rare. Intuitively, the traffic pattern for a pair-wise colluding group is where two peers upload a relatively large amount of their total upload traffic to each other. To quantify this, we define the property *pair-wise degree*. For two peers A and B, pair-wise degree is the sum of all bidirectional traffic between A and B, divided by the sum of total traffic uploaded by A and B. This gives rise to the detector of group collusion.

Detector 2: (Pair-wise detector) *large amounts of mutual upload traffic compared to total uploads.*

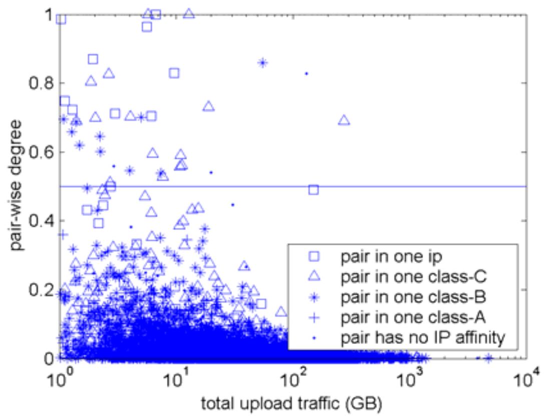


Figure 4: Pair-wise collusion detector by the ratio of mutual upload traffic over total traffic.

Figure 4 shows the statistic results of applying this detector to our dataset. There are 28 thousands pairs of peers with mutual uploads, every point in the figure stands for such a pair. The x-axis is the total uploads by these two peers, and the y-axis is the corresponding pair-wise degree. The horizontal line denotes pair-wise degree equals to 0.5. Above that line are pairs whose mutual upload exceeds uploads to peers external to the pair, and there are 73 of them. While it is possible for two friends to share large amounts of mutually interesting content, but the highly concentrated nature of these uploads appear indicative of collusion. As Maze becomes popular, it may even be the case that a user uses Maze to transfer personal files between two of his machines. Regardless of the actual reasons, whether there is intent to collude, such behavior still 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 laborious to transfer large amounts of data through a narrow pipe just for the purpose of colluding. One may argue that colluders might anyway do pair-wise colluding across wide-area if they are truly desperate. But they can more easily achieve sufficiently high points simply by whitewashing. Thus, we expect that good connectivity between peers is a requirement for pair-wise collusion. To verify this, we again analyzed the IP affinity of peer pairs, labeled with different symbols in Figure 4. We see that most colluding peers have similar IP addresses. IP address vicinity implies they are likely physically close to each other in the network and therefore are connected using a higher bandwidth connection.

Peer 1 external	Peer 1	Mutual upload	Peer 2	Peer 2 external
1.7GB	Fred	24GB→←23GB	Gary	5GB
23GB	Cindy	81GB→←27GB	Harry	0GB
52GB	David	62GB→←126GB	Alice	32GB

Table 2: Top-3 big mutual upload pairs. Peer 1/external includes all traffic from peer 1 not going to 2.

We take a closer look at some specific examples of possible colluding peer-pairs. Table 2 lists three top 3 pairs ranked by pair-wise degree. Considering the asymmetry of the point system, even the most unbalanced peer (Harry) will end up with a net point gain (after uploading 27GB and then downloading 81GB). Recall that our repetition-based collusion detector found a 3-party collusion. Two of the peers in this collusion are detected as a pair-wise collusion (peer David and Alice). However, this match does not necessarily mean that the pair-wise detector can detect larger collusion topologies. If a group of N colluding peers collude by perfectly balancing their traffic across mutual links, the pair-wise degree between any two peers can drop as low as 1/(N-1). With the possibility of creating much more sophisticated colluding topologies, designing a robust group-collusion detector clearly remains a great challenge.

3.3 Spam account collusion

In our repetition-based topology in Figure 3: Collusion link topology of 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. Compared to pair-wise colluding which involves a limited number of peers, this is counterintuitive: what would motivate these leaf-peers to download duplicate content from the central peer without any benefit in return? The traffic is only unidirectional. Even more puzzling is how the central peer found so many "selfless" peers to cooperate?

The truth is that the center peer is the colluder, and the leaf-peers are also controlled by the colluder. We call these leaf-peers *spam accounts*, peers that are created and then discarded when they become useless. Using spam accounts is a creatively way of leveraging the zero-cost nature of identities in Maze. This strategy is similar to the link spam [17] problem in search engines using page rank to sort results. Why would a Maze user use this strategy rather than just using a single machine to restart with new identities, or whitewash? One possible reason is that users typically want to maintain one primary account for social status or to maintain Maze-related state such as. the Maze point total or the friends-list. To do this, a Maze user needs to maintain a persistent primary account active. This type of collusion is also efficient because it earns points much faster than pair-wise collusion for the same amount of traffic. In order for this to work, the colluder must have more than one machine. If a user has access to only one machine, then she can cheat the system only through account whitewashing.

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 3. Except for Ted's group, there is generally strong IP address similarity 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 accounts can be large in number, but live on relatively few machines. We define *PM ratio* (number of peers / 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 colluding.*

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

Table 3: Peers suspicious of doing spam account colluding, as found by repetition detector

We use the method described in Section 2.2 to associate a peer with its machine. One problem with the PM value is the signal to noise ratio. A single upload to some random peer will count as an additional peer-machine pair and significantly reduce the PM value. We remove these noise values by discarding the bottom smallest uploads that, in aggregate, holds less than 20% of all upload traffic. Figure 5 presents the statistical result with each point representing one peer.

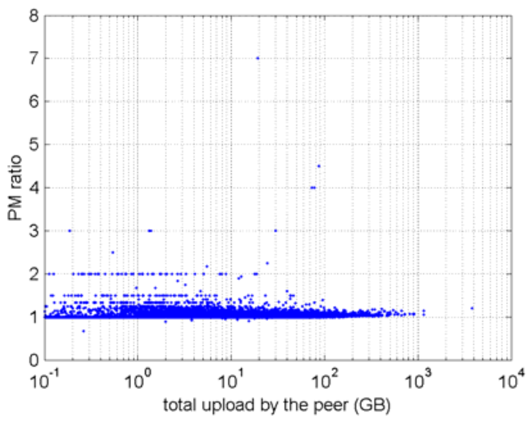


Figure 5: Spam account detection by PM ratio.

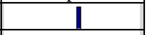
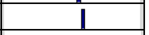
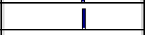
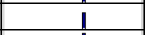
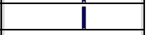
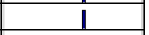
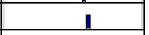
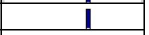
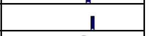
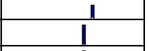
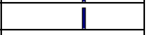
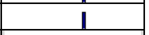
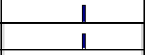
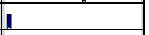
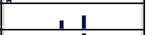
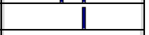
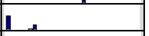
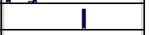
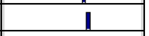
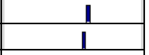
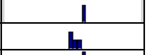
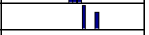
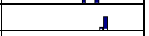
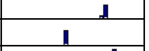
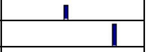
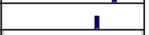
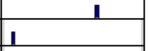
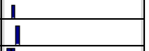
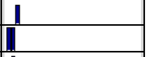
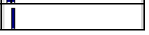
source peer U/D	Upload traffic	Maze id	Client total d/l	Machine id	Client activity temporal
Jane 19GB 450MB	3.4GB	C20	3.4GB	M1	
	3.2GB	C21	3.2GB	M1	
	3.1GB	C22	3.1GB	M1	
	2.5GB	C23	2.5GB	M1	
	1.7GB	C24	1.7GB	M1	
	1.5GB	C25	1.5GB	M1	
	1.1GB	C26	1.1GB	M1	
Mary 73GB 5.2GB	31GB	C27	32GB	M2	
	10GB	C28	10GB	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 4: Some top spam account colluders

The x-axis of Figure 5 is the total uploads made by the peer, and the y-axis is the peer's PM ratio. We can see that most peers have PM ratio slightly above 1 and below 2. This is statistically normal because of a good portion of Maze users do whitewashing [19], which means that on average, every machine hosts more than 1 peer. However, there are peers with exceptionally high PM ratios (the highest reaches 7). It means that these peers are mostly whitewashers acting as spam accounts helping a peer collude. Table 4 lists the peers whose upload > 10GB and have PM ratio > 3 (and thus some of the peers in Table 3 are not included). 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 cannot be certain they are actually colluding. To dig deeper, we make use of three heuristics: 1) The spam accounts should have good connectivity to the colluder (the center peer). We use IP address similarity to infer this. 2) Spam accounts only download data from the center peer. 3) 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). One exception is Mary's spam accounts (C27, C28, C29, C30 and C31). It turns out that they download from another peer with closely related content (several chapters of a Korean television episode). The center peer (Mary) also downloaded related content from the same source. Note that peer Ted, which was identified as a center of a star with 8 fan-outs each carries large duplicate upload, is not included. As we mentioned earlier, Ted is the largest uploader of the month and seems to not be a colluder.

3.4 Upload traffic concentration

Pair-wise colluding and spam account colluding share one common trait: there is a high volume of upload to relatively a few destination machines. This observation is important, because we now shift our focus from the flow among peers to among physical machines. In pair-wise colluding, two machines upload towards each other. In spam account colluding, uploads flow to a few machines on which the colluder repeatedly generate new spam accounts. This is intuitive because colluders, in general, control a limited number of machines.

We define the *traffic concentration* degree or *TC degree* in short, 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 the machine receiving the most traffic receives 90GB, then the TC degree of *x* is 0.9. On the other hand, if *y* has 100 clients each residing on a different machine, and each of them downloads 1GB from *y*, then *y*'s TC degree is only 0.01. The higher the TC degree, the more likely that the peer is performing either pair-wise or spam-account colluding. This is our fourth detector:

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

The results are summarized in Figure 6, where each dot represents a single peer. The x-axis is the peer's total upload, while the y-axis is the TC degree. In general, the more uploads a peer has, the more likely that uploads are scattered across a wider range of machines (and peers), resulting in a lower TC degree. Figure 6 confirms this in our data set. For peers who upload around 10GB, their TC degrees are roughly 10%. For heavily uploaders who upload around 1TB, the TC degree drops to about 1%. However, colluders show up differently with exceptionally high TC degrees:

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 /ratio of biggest)
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

Table 6: 7 Top colluders and how our detectors have found and missed them

they are located in the middle towards the top of the graph (total uploads about 100GB and large ratios close to 1). There are seven peers that have uploaded more than 50GB and their TC degree is larger than 0.6. This means that more than 60% of its 50GB uploads is going to a *single* machine). We take a closer look at these specific peers in Table 5.

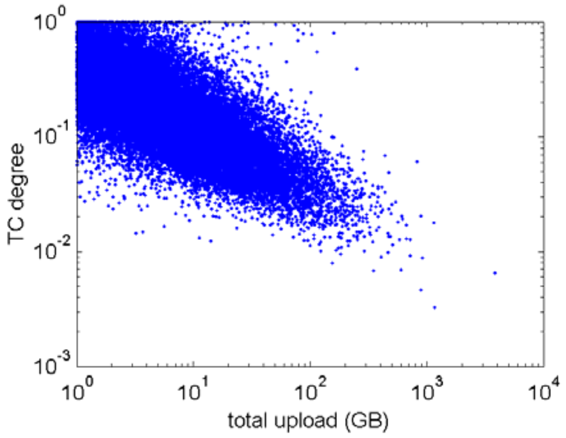


Figure 6: Upload traffic vicinity detector result

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

Table 5: Top 7 colluders detected by TC detector

We list seven peers in Table 5. For each peer, we list the total traffic going to the beneficiary node. Six of them are also detected by previous detectors, but there is a new peer: Nancy. The pair-wise detector missed it because Nancy 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 almost one. It turns out that it ranks #7 by repetition detector (we listed only the top 6 in Section 3.1).

4. Comparing collusion detectors

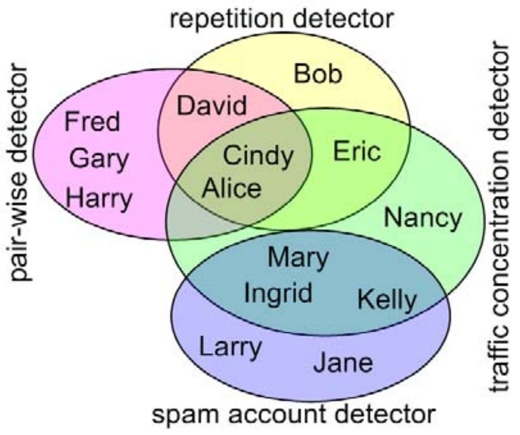


Figure 7: Venn diagram of collusion detectors.

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 6 lists the top-7 colluders according to the total upload traffics. It also shows (with shaded cell) which detector is responsible for finding out the colluder.

The first observation we make 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. It does *not* mean that there are no colluders who engage in both activities simultaneously. Doing so can potentially evade our detectors. However, given that we have no active collusion detection mechanism and that spam-account colluding earns points in a more “cost-effective” way, there is reason to believe that this is does not happen in practice.

As we discussed earlier, the traffic concentration detector is reasoned out of one straightforward observation, that colluders generally control relatively few machines. Thus it looks only at how “concentrated” a peer’s upload traffic goes to other machines. Figure 7

Detector	Heuristic	Strength	Weakness
Repetition	Colluders use a small colluding working set	General detector, and helps us find complex collusion topology	Easily defeated by randomized colluding working set
Pair-wise	There is more mutual upload than upload to external	Can pin-point the pair with high accuracy	Specialized for pair-wise colluding, and does not work for more complicated group colluding topology
Spam	Upload to a large number of whitewashed accounts located on small number of machines	Works fine for finding spam collusion specifically	Specialized for spam-account colluding
Traffic concentration	Colluder control relatively small number of machines	General detector, and works fine for most collusions we found	Setting the right parameter will be tricky

Table 7: Detector strength and weakness summary

shows that indeed, this detector covers a good portion of both spam-account and pair-wise colluders. To understand this further, we take a closer look of the cases that this detector missed. The colluders missed by traffic concentration detector are Bob, Fred, Gary, Harry, Larry, and Jane (last column of Table 6). We can see that Bob is missed because its TC degree is not high enough (0.39). All others have very larger TC degrees (the smallest being 0.80), and they are missing from this table simply because of their relatively small total upload traffic (we only looked at those have upload traffic greater than 50GB). This shows that the TC detector is in fact quite effective at detecting colluders.

While promising, it is still too early to conclude that the TC detector will successfully detect colluders in an online fashion. As discussed, identifying a peer's machine in the presence of DHCP, NAT and other issues is non-trivial. Choosing the threshold for collusion in TC degree detector is still quite ad hoc.

Table 6 shows that the repetition detector misses six colluders (column 2). 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 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.

5. EigenTrust and collusion

Another way to look at the Maze point system is that it uses a peer's upload amount as a way to calculate its global reputation score. While we have shown that it has been effective in encouraging sharing [19], the current scheme is flawed in the sense that a peer's contribution is measured only by its aggregate upload, but not by how widely its contribution benefits the community at large. This is why the TC degree, though quite crude, would have been a better way to capture a peer's contribution.

According to the taxonomy proposed in [6], incentive mechanisms can be categorized into those using private history, shared history, or subjective shared history. The authors pointed out that non-subjective shared history based schemes such as Maze are vulnerable to the collusion attack. To some extent, our results have validated their conclusions. The authors proposed a maxflow algorithm based on subjective shared history as the counter measure. The algorithm basically calculates the services that the downloading 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." While the algorithm is interesting, it also means that there will not be any global ranking of peer reputation. This, in turn, means that there will be no list of “celebrities” such as the “altruistic list” that Maze uses. Our experience with operating Maze thus far proves one thing: the P2P file-sharing system is much like a society, and things like the “altruistic list” is highly useful, even though it logically leads to a global ranking system and thus opens the door to collusion.

One of the best known algorithms of global ranking system is the EigenTrust proposal [8]. The EigenTrust ranking can be used for both reputation management [8] (used for clients to choose the trustworthy download sources) and free rider detection [9] (used for uploader to choose trustworthy clients). Although our primary focus of the paper is to understand collusion behavior, it would be interesting to run the EigenTrust algorithm over our logs. Intuitively, if the algorithm is robust, it should give the colluders low scores.

5.1 EigenTrust description

We first give a high level description of the EigenTrust system [8]. 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* can assign another peer *j* trust values C_{ij} based on its downloading experience from *j*. The trust values for all *j* are normalized locally by each peer *i*. At this point, we obtain a matrix *C* containing the trust value of the pairs of peers of the entire system. The trust vector *t* is defined as the left principal eigenvector of *C*. The component t_i is called the EigenRank of peer *i*, this value represents the peer’s global reputation. This achieve the goal: “the global reputation of each peer *i* is given by the local trust values assigned to peer *i* by other peers, weighted by the global reputations of the assigning peers.” [8]. The algorithm can also be explained by “random walk” as follows. Imagine a great number of ants randomly walking among peers, with probability C_{ij} to move from peer *i* to peer *j*. At the stable state, the number of ants at each peer will be proportional to its EigenRank.

The basic algorithm can be further improved to enhance its robustness against malicious users. To do that, it incorporates the notion of some 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 described in Figure 8.

The parameter *a* is a constant less than 1, used to primarily deal with malicious collectives including colluders. It means that, when calculating the trust vector, each peer will place some trust on the pre-trusted peers. A higher value of *a* implies more confidence on the pre-trusted peers.

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

Figure 8: Basic EigenTrust algorithm

5.2 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 downloading of peer *i* from peer *j* during the log period. Then, we normalize the local trust value c_{ij} such that:

$$\sum_{j=1}^{N} c_{ij} = 1$$

From the Maze forum, we select ten peers that we are confident that they can act as the pre-trust peers (i.e. |P|=10). At this point, we have obtained the matrix *C* and the pre-trust peer set *P*. Finally, we set *a*=0.1. We are now ready to run the EigenTrust algorithm.

5.3 Experiment results

Figure 9 shows the EigenTrust values for the 9568 peers whose total uploads are more than 10GB. There are two interesting observations we can make about the result. First, generally speaking, the more uploads a peer has, the higher its score will be. Thus, if the Maze system did not have colluders and whitewashers, its primitive point system should have been sufficient. Second, the peers are spread in two noticeable bands. We drew a line and partitioned peers into two (H and L) regions, as shown in Figure 9. Out of roughly 9600 peers, 551 are in region L. If we focus on peers with the same upload traffic, this means that the reputation values of those in region H are far higher than those in region L (about 10^3 times). Does that imply that peers of region L are colluders?

To answer this question, we label the positions of the fourteen colluders detected from earlier sections using squares (see Figure 9). Peer Mary is absent because its EigenTrust value is 0 and is outside the y-axis scope. The result is somewhat unexpected. While a portion of the colluders have low scores and belong to region L, many others are in fact in region H. Therefore, generally speaking, the split of the two regions can not be simply attributed to collusion. So what is the reason?

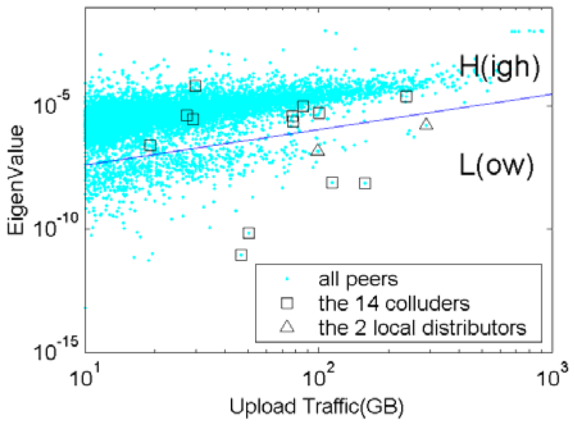


Figure 9: The EigenRank of peers

Region	Average # of distinct client IP	Avg. # of Class-B spaces clients located in
H	299.5	59.38
L	98.323	2.18

Table 8: Client IP distribution of regions H and L

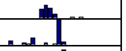
Peer U/D	Upload traffic	Top clients	Client total d/l	Machine id	Client activity temporal
Wayne	5.4GB	C50	34.3GB	M21	
290GB	4.6GB	C51	15.7GB	M22	
3.9GB	4.5GB	C52	7.9GB	M23	
	3.4GB	C20	3.4GB	M1	
	3.2GB	C21	3.2GB	M1	
Jane	3.1GB	C22	3.1GB	M1	
19GB	2.5GB	C23	2.5GB	M1	
450MB	1.7GB	C24	1.7GB	M1	
	1.5GB	C25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

Table 9: A comparison of a non-colluding region-L peer (Wayne) with a region-H spam-account colluder (Jane)

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, there must be something of the clients that tells these two groups apart. After analyzing the data, we found that there is significant difference in the IP address distribution of the clients of region-H and region-L peers. The data is shown in Table 8. On average, region-H peers upload to about 300 distinct IP addresses which are scattered in 60 class-B spaces. This means that each class-B space contains, on average, 5 IP addresses used by a region-H peer's clients. On the other hand, region-L peers upload to 98 distinct IP addresses scattered in 2.2 class-B spaces. Thus, each class-B space contains about 45 IP addresses used by a region-L peer's clients. Therefore, comparing against region-L peers, the key difference is that region-H peers have more clients, and they are more widely spread geographically. Region-L peers appear like a "local distributors" (marked as triangle in Figure 9)

Table 9 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.

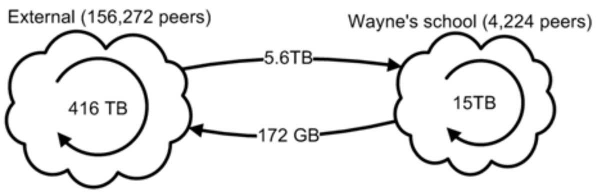


Figure 10: Wayne's school is a satellite cluster

A closer look at Wayne's uploading history reveals that 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. If this cluster does not upload heavily to external peers, even though there are many intra-cluster traffics, the net effect of the EigenTrust algorithm is to treat it as a big misbehaving group. We query Wayne's IP address in the APNIC whois database, and find that Wayne belongs to a university. Most of Wayne's clients are in the same university. We calculate the internal and external traffic of this school and the results are shown in Figure 10.

The total traffic that Wayne's school consumes is the sum of its internal traffic (15TB), plus 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 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, we are not certain that the scores assigned to individual peers are justified: a non-colluding region-L peer such as Wayne has contributed to a great number of other peers.

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

Table 10: A pre-trusted peer helps colluder Larry

Now let's turn to the problem of why some colluders have such high EigenRanks. The colluder Larry has a much higher EigenRank than many other normal peers who have comparable uploads. Figure 9 shows that there are many colluders located in region-H. It is difficult to determine all the contributing factors. One possibility is that pre-trusted peers may have unintentionally helped colluders to elevate their EigenRanks. Table 10 presents a case where a pre-trusted peer Ted helps colluder Larry. Larry uploads 29.7GB, mostly of which are colluding uploads. The pre-trusted peer Ted has a total of 3.8TB of uploads, and it has also downloaded 124MB data. Out of downloads totally 124MB, 734KB is from colluder Larry. That is to say, only 0.59% comes from this colluder. This tiny traffic raises Larry's EigenRank nearly 10 times (from 8.2e-6 to 6.6e-5). Removing this upload would have dropped the colluder's ranking from 334 to 1905. Further more, Larry also has some other uploads to reputable peers. If we remove its top 200MB uploads to "celebrity peers," it drops into region L (EigenRank further drops to 7.4e-8) Decreasing the value of *a* (which has a net effect of placing less trust on the pre-trust peers) produces similar effect. However, these colluders may simply be lucky. These findings seem to suggest another vulnerability of the EigenTrust algorithm. It may be fair for peers outside of Wayne's school to treat Wayne as colluder, but peers located inside his school should not. This implies that global ranking doesn't work well in this instance.

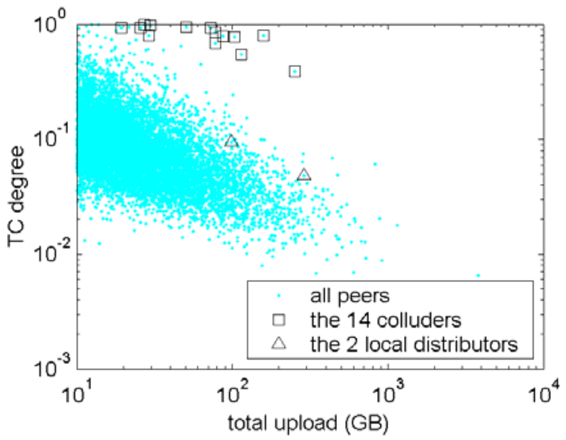


Figure 11: The TC plot with local distributors and colluders.

To put things into perspective, we conclude this section by redrawing the results of using the traffic-concentration detector in Figure 11, which is 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 appears to be a more robust and simpler add-on for the basic point system.

6. Related work

There is much work focused on incentive systems for peer-to-peer networks. They are categorized into three types according to the taxonomy in [6]: private history, shared history, and subjective shared history. Most of the early works focus on the free-rider problem. For example, 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 [3].

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 [4]. This limits BitTorrent to generally small groups in individual download sessions. To overcome this drawback, many shared history solutions are proposed. They can be generally categorized into two types (according to [11]): virtual currency based and reputation based, e.g., Mojo Nation [16] or Maze and Free Haven [4]. There is also a hybrid approach called Stamp [11], where any peer can issue stamps. These stamps act like currency. The value of each peer's stamps is maintained by its exchange rates, which acts as a reputation value.

The new problem introduced by shared history is collusion. Collusion can use forged shared history to increase the ranking of colluders [6]. There are two types of collusion. Group collusion builds mutually high ranking among group peers, and spam account collusion uses spam accounts to generate an artificially high ranking to a single colluder. The Stamp algorithm solves part of this problem by enabling exchange rate among various kinds of stamps. However, a more generic solution is to use subjective shared history [6], including both the maxflow [6] and EigenTrust [8][9] approaches. In maxflow each peer ranks other peer on its own perspective; while in EigenTrust the entire system ranks all peers globally. Maxflow is an ideal solution but is expensive to implement in a real system.

Most of the incentive schemes focus on curbing the free-riding behavior. In other words, they want to motivate users who otherwise do not want to share. Our experience is that there are some peers not following this assumption at all: some peers upload huge but seldom download. According to Maze forum, their goal is to climb social ladder for the celebrity effect, and the incentive system becomes the tool for them to leverage. Such exceptionally high ranked peers break the basic assumption, and their random downloads elevate the colluder's reputation as well, as we have showed.

EigenTrust is similar in spirit to the page ranking algorithm [2]. Many issues we discuss also exist in the world of webpage ranking. For example, link spam collusion [17] is a combination of our spam account collusion and group collusion. It would be worthwhile to apply some of their results to collusion in P2P file-sharing systems.

7. Conclusion

Our work seeks to present a first-hand empirical analysis of colluding behavior in a real peer-to-peer file sharing system. With total access to the popular Maze file sharing system, we analyze a complete user and traffic log collected during the course of a month on Maze. 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, application of our detectors provides substantial evidence of collusion-like behavior.

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. Finally, while our observations are made on Maze, the collusion patterns we observe are likely to occur in any system without point conservation. Outside of BitTorrent's tit-for-tat scheme, Maze is one of the few peer-to-peer systems with an active incentive structure, and our lessons should serve to guide the design and deployment of future distributed incentive schemes.

References

[1] R. Axelrod, "The Evolution of Cooperation", New York: Basic Books, 1984.

[2] S. Brin, L. Page, "The anatomy of a large-scale hypertextual Web search engine," In *Proceedings of WWW*, Brisbane, Australia, April 1998.

[3] B. Cohen, "Incentives Build Robustness in BitTorrent," In *Proceedings of Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[4] R. Dingledine, M. J. Freedman, and D. Molnar. "The free haven project: Distributed anonymous storage service." *LNCS 2009*, 2001.

[5] J. Douceur. "The Sybil Attack." In *Proceedings of IPTPS*, Cambridge, MA, 2002.

[6] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust Incentive Techniques for Peer-to-Peer Networks," In *Proceedings of EC*, May 2004.

[7] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, "Free-Riding and Whitewashing in Peer-to-Peer Systems," In *Proceedings of ACM Workshop on Practice and Theory of Incentives in Networked Systems (PINS)*, August 2004.

[8] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina, "Incentives for Combating Freeriding on P2P Networks," In *Proceedings of Euro-Par*, June 2003

[9] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks" In *Proceedings of WWW*, May 2003

[10] R. Ma, S. Lee, J. Lui, D. Yau, "A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks," In *Proceedings of Sigmetrics-Performance*, New York, NY, June 2004.

[11] T. Moreton, A. Twigg, "Trading in Trust, Tokens, and Stamps," In *Proceedings of 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley CA, June 2003

[12] T.-W. J. Ngan, D. S. Wallach, and P. Druschel. "Enforcing fair sharing of peer-to-peer resources." In *Proceedings of IPTPS*, Berkeley, CA, February 2003.

[13] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". In *Proceedings of ACM Middleware*, Heidelberg, Germany, November, 2001.

[14] S. Saroiu, K. P. Gummadi, R. Dunn, S. Gribble, and H. M. Levy, "An analysis of Internet content delivery systems," In *Proceedings of OSDI*, December, 2002.

[15] S. Saroiu, K. P. Gummadi, and S. Gribble, "A measurement study of Peer-to-Peer File Sharing Systems," In *Proc. of Multimedia Computing and Networking*, 2002.

[16] B. Wilcox-O'Hearn. "Experiences deploying a large-scale emergent network." In *Proceedings of IPTPS*, Cambridge, MA, 2002.

[17] B. Wu, Brian D. Davison, "Identifying link farm spam pages," In *Proceedings of WWW*, Chiba, Japan, May 2005.

[18] M. Yang, H. Chen, B. Y. Zhao, Y. Dai, and Z. Zhang, "Deployment of a Large-scale Peer-to-Peer Social Network," In *Proceedings of WORLDS*, San Francisco, CA, Dec. 2004.

[19] M. Yang, Z. Zhang, X. Li, Y. Dai, "An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System," In *Proceedings of IPTPS*, Ithaca, NY. February 2005.

[20] T-Net, http://e.pku.edu.cn