Robust Incentives via Multi-level Tit-for-tat

Qiao Lian[‡], Yu Peng[§], Mao Yang[§], Zheng Zhang[†], Yafei Dai[§], and Xiaoming Li[§]
[‡†]Microsoft Research Asia, Beijing, P. R. China
[§]Peking University, Beijing, P. R. China
[‡]lianqiao@gmail.com, [†]zzhang@microsoft.com, [§]{py, ym, dyf, lxm}@net.pku.edu.cn

ABSTRACT

Much work has been done to address the need for incentive models in real deployed peer-to-peer networks. In this paper, we discuss problems found with the incentive model in a large, deployed peer-to-peer network, Maze. We evaluate several alternatives, and propose an incentive system that generates preferences for well-behaved nodes while correctly punishing colluders. We discuss our proposal as a hybrid between Tit-for-Tat and EigenTrust, and show its effectiveness through simulation of real traces of the Maze system.

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, there are a lot of free-riders who consume without sharing. 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.

Maze[18][19] is a large and deployed P2P file sharing network and it has a simple incentive mechanism that imposes service differentiation with a simple point system. This point system tallies a peer's net contribution in its entire history as its point. By examining detailed logs over a long enough period of time, it is evident that colluders use variations of Sybil attack to exploit the weakness of the incentive system. This, in addition to the longstanding problem of free-riding, propels us to design the next generation incentive system for Maze.

We borrow the core principle from the Tit-for-Tat strategy and link reputation and incentives together: by collaborating with more reputable peers it is possible to improve a peer's own reputation and thus the service it gets in return. However, mechanisms based on purely private history do not scale: coverage will be so small that cheaters are indistinguishable from good peers. A practical solution is to leverage other peers' opinions and gradually move from private history to shared history. Our study shows that if shared history is employed in its uttermost form as proposed in EigenTrust[9], the system is vulnerable to other problems. Our multi-trust mechanism is designed to achieve the best balance between the two extremes. We have performed detailed simulation that validated our algorithm. The new mechanism is implemented and ready to be deployed.

In the remaining of this paper, Section 2 discusses related work. Section 3 introduces Maze system and the collusion cases we have found, followed by discussions of potential problems of Tit-for-Tat and EigenTrust. Section 4 discusses the design of the multi-trust algorithm, from its mathematical form to implementation consideration. Section 5 evaluates the multi-trust algorithm using trace driven simulation and verifies its effectiveness. We conclude in Section 6.

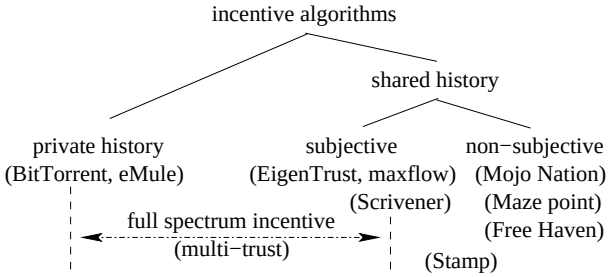

Figure 1: The category of incentive algorithms and examples

2. RELATED WORK

Nielson et al. [14] classifies incentives into two categories. The first type is *genuine incentives* as exemplified by incremental block exchanges in BitTorrent[3]. This mechanism works well when the file is hot and there are many concurrent users in a session. However, our experience with the Maze system has been that the distribution is heavily long-tailed, on one typical day we found around 80% downloading involves at most one downloader. We conjecture that this is common for other file sharing networks. Such long-tail behavior requires the incentive system to record current behavior for future reference. This type of incentive is termed as *artificial incentives*, and must be based on observed history. History can be harnessed in different ways, Feldman et al. [6] classify into private history, shared history, and subjective shared history(Figure 1). Basically, private history means a peer rewards other peers with which it has good experience. Shared history looks at the overall contribution of a peer by sharing private history among peers. In subjective shared history, a peer's reputation is given by the local trust value assigned by other peers, weighted by the reputations of the assigning peers.

Systems based on private history generally do not scale, although it has already been deployed in successful systems such as eMule[10] and BitTorrent[3]. In a large network, each peer can only interact with small percentage of peers[4]. Scrivener[15] proposes to use transitive trading to scale the relationship, but the complexity is to find a valid path, which becomes increasingly difficult when the system scales up. A related problem is that rigorous schemes as such could also degrade system performance in general, as pointed by Ganesan[8]. Many systems thus propose to let peers share their

own experiences to rank other peers [4][12][16][18][19].

However, shared history introduces the collusion problem. Colluding peers can forge shared history to increase each other's points or rankings. We have found active collusion in Maze[11], the result of which motivates this work. In general, a more sound solution is to use subjective shared history, as proposed by maxflow[6] and EigenTrust[9]. Peers in maxflow rank other peers using its own perspective, while the entire system ranks all peers globally in EigenTrust. Maxflow is ideal, but is prohibitively expensive for real systems. As we have experimentally verified, subjective shared history has its own problem of false negatives and false positives.

3. BACKGROUND: MAZE AND EIGENTRUST

3.1 The Maze Peer-to-Peer System

Our work builds on the Maze[18][19] P2P file-sharing system. Maze is a large deployed file sharing system with more than 1.4 million registered users. An average of 30,000 peers are online at any time, and roughly 13TB of data are exchanged daily. We have included the transfer log upload mechanism in our client side software, so that our log collection server can get the complete trace data we need for this paper. We begin with some background on the current Maze incentive system and the observed collusion behavior.

3.1.1 The point 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. Since the Maze central server audits all transfers and tallies points accordingly, Maze's incentive policy is in the shared-history category.

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 is 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 peers to download at least 3GB of data before its downloads are throttled at 300 kbps.

3.1.2 Existent cheating

To understand the impact of these incentive mechanisms on user collusion, we analyzed the complete log of all transactions in the Maze system over a one month period. By examining every transaction between all peers, besides the expected free-riding behavior[19] and user whitewashing[7], we found empirical evidence of user collusion as below. More details can be found in [11].

Pair-wise collusion: Two colluders mutually exchange large amounts of data to increase points for each other. This behavior exploits net point gain from mutual transactions, and is the simplest case used by most of the colluders.

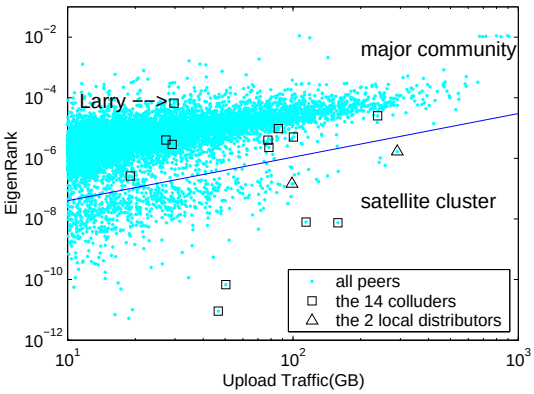

Figure 2: Result of applying EigenTrust to Maze logs. The figure shows two bands of users: those in the satellite cluster (lower half) and everyone else. Satellite users are punished as one group of colluders, even if some user (local distributor) have high contribution to those within.

Spam account collusion: A colluder registers a number of "spam" accounts, uses them to download data from the main account, thereby transferring spam accounts' initial points to the colluder's main account. This collusion is a form of whitewashing that allows whitewashed points to be collected at a single user.

3.2 The limitation of other incentive systems

To reduce user collusion, we examined a number of alternative incentive designs, mainly focusing on variations of Tit-for-Tat and EigenTrust. We briefly describe our thoughts on the strengths and weaknesses of each that motivated us to design an alternative incentive system.

3.2.1 EigenTrust

EigenTrust works similar to the PageRank[2] algorithm used by Google. The page link in the PageRank algorithm becomes traffic flow in EigenTrust. EigenTrust falls into the subjective shared-history category, and assigns each peer a global EigenRank value by computing the left principle eigenvector of the trust matrix transit M^T. $M_{i,j}$ is the rank of j from i's perspective. Our offline calculations show that EigenTrust helps to punish colluders in Maze, but suffers from both false negatives and false positives.

False negatives: We observe that the distribution of points across users is highly skewed. A number of super peers provide a large number of uploads to many users while only downloading infrequently. Because of their high reputation values, their random downloads from a colluder immediately boost the colluder's reputation. Larry is a spam account colluder that we detected. , e.g. Most of Larry's 30GBs uploads are to other colluders, but 200MB uploads to some reputable peers boosts his rank nearly 100 times from $7.4 \cdot 10^{-8}$ to $8.2 \cdot 10^{-6}$(Fig. 2). Another 734KB upload to one super peer further promotes its rank to $6.6 \cdot 10^{-5}$. With these "leg-hugger" uploads, Larry gets a higher rank than most of peers who upload more than 30GB to legitimate users.

False positives: Systems like EigenTrust can unfairly punish peers inside satellite networks such as college networks. As shown in Figure 3, a college network as a whole consumes by downloading much more than it uploads, but still performs

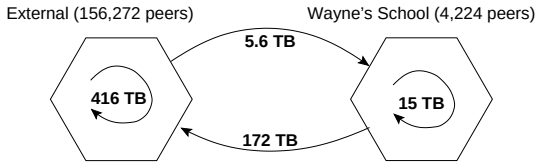

Figure 3: Satellite cluster example.

a large amount of exchange among members. EigenTrust treats the entire college as a collusion group. Roughly 5% of users in our dataset are inside these "satellite" clusters. EigenTrust generally ignores internal transactions and ranks most peers similarly. As a result, those that contribute heavily within the cluster – we call such peer *local distributor* – will be unfairly punished. For example, two local distributors who upload more than 300GB are ranked equally to an outside peer who uploads less than 10GB (Fig. 2). This implies internal peers should trust outside peers instead of nearby peers. In addition to being incorrect, this result destroys any attempts to leverage network locality for more efficient bandwidth utilization. Transfers will traverse the bandwidth limited wide-area links rather than higher throughput internal links.

3.2.2 Private history coverage problem

Solving the satellite cluster problem requires that peers utilize personalized rankings. The simplest and best-known personalized rank algorithm is the private history based Tit-for-Tat. Basically, a peer gives higher priority to those that it has successfully downloaded before. However, purely private history does not scale. In a large network, peers may interact on an infrequent basis with many others. If private history covers a very small percentage of peers, a cheater is indistinguishable from a good peer.

Our simulation results confirm these expectations. It shows that even with long private history it is difficult to improve coverage. A one month download log only enforces Tit-for-Tat to only 2% of a peer's upload, and the other 98% uploads are just blind upload, i.e., peer have no opinion of the requesting peer. The blind uploads bring a lot of opportunity for free-riders. In a very large p2p sharing system, it is enough for one free-rider to get enough benefit by cheating every peer only once[8]. This encourage peers to create more accounts than establishing long term trust relations.

4. DESIGNING MULTI-TRUST INCENTIVE

Tit-for-Tat [1][3] links incentive mechanisms with reputation: by collaborating with more reputable peers it is possible to improves a peer's own reputation and thus the service it gets in return. The main problem of purely private history is its coverage, and resolving it requires leveraging other reputable peers' history. Pushing this direction to its extreme, however, arrives at the EigenTrust mechanism. Our multi-trust solution attempts to achieve a balance in between. We will now discuss the core algorithm and its implementation.

4.1 A Mathematical Discussion of Multi-trust Incentive

Let M be a $N \times N$ matrix that defines a one-step rank among peers, i.e., $M_{i,j}$ is peer j's rank from i's perspective. In practical terms, this can be measured as the normalized download volumes that i has received from j during a period of T. e.g. k has uploaded ten times more than j does to i, then $M_{i,k} = 10M_{i,j}$, and $\sum_j M_{i,j} = 1$. This is actually Tit-for-Tat, and the matrix is sparse because it only covers a peer's immediate friends as non-zero entries.

Similarly, the two-step rank matrix (one-level indirect trust) can be expressed as M^2. The entry $(M^2)_{i,j}$ aggregates other peers one-step rank to yield the rank of j from i's perspective. For instance, if $M_{i,j}$ is 0.5, and $M_{j,k} = 0.1$, then $(M^2)_{i,k}$ is added by the value of 0.05, and this is to be performed for all such j. This produces two effects. First, the coverage of the rank matrix gets larger as it becomes exponentially less sparse. Second, the rank starts to mix in more and more peers opinions.

In general, we can obtain the n-step rank matrix by continuing the above steps. The coverage continues to increase and the rank becomes more and more global, moving towards shared history based algorithm. In fact, M^∞ is exactly the EigenTrust matrix, and the entries in each column are the same and the matrix can be collapsed into the EigenTrust vector: every peer has the same rank on any other peer, and the vector offers complete coverage.

The above discussion reveals that this series of rank matrix gives a full trust spectrum, with Tit-for-Tat and EigenTrust as two extreme ends. The insight here is that, when deriving incentive metric for service differentiation, we need to consider – ideally – all these matrices instead of one. The immediate friends form the first tier, friends' friends form the next, and so on. Each matrix precisely represents the trust a peer imposes on others at a different level, and as the levels go deeper, the ranking become more global and less private. We can not use M since its coverage is too small, and likewise M^∞ is also insufficient since, as we have discussed, EigenTrust is vulnerable to a number of issues.

Our multi-tier incentive scheme essentially imposes service differentiation by looking at which tier j falls into when its downloading request arrives at i. The smaller level it belongs to, the higher priority it is given. Within the same tier, two peers will be ranked according to their values in the matrix of that tier. Obviously, i only needs the i-th row of the matrix series, and for all practical purposes we only need small number of levels.

4.2 Implementation

The multi-level trust model can be easily implemented in a distributed manner. For a duration of t, peer i computes $M_{i,*}$ by normalizing all the downloads it has had. Periodically, it will ask those immediate friends (i.e. non-zero entries in i's M row) for their $M_{j,*}$ row, and thus enables it to compute its own $(M^2)_{i,*}$ row. This procedure can be repeated iteratively, in the sense that as long as i can get $(M^k)_{j,*}$ from its immediate friends, it can compute $(M^{k+1})_{i,*}$.

This fully distributed algorithm is very practical when the level is small. From the Maze log, we found that the average size of a peer's immediate friends for one day is about 36. Getting all the 36 friends $M_{j,*}$ rows thus amounts to less then 32KB total. It is still manageable for level two, where it becomes about 1MB and daily update does not impose a significant overhead. However, cost progressively grows when moving to higher levels. Nevertheless, as our simulation shows later, using two level (i.e. M and M^2) already covers more than 60% of total traffic, because the download traffic exhibits small world pattern. For all peers that miss these matrix, we will use M^∞ to approximate. Since a peer's

EigenTrust rank is global, the value is carried along with the request instead.

From the engineering perspective, we would like to introduce complexity into the real system in a progressive manner. Thus, we have implemented the scheme by calculating $\{M, M^2, M^\infty\}$ in the Maze central server. These rows are then signed digitally and sent to the peers when they start a Maze session. The computation takes roughly two hour to complete. At the Maze peers, the service differentiation will be imposed as before, but now on this new metric instead of the original points.We plan to measure the effects and then move on to the distributed version of the algorithm.

We select t to be one day in our implementation. This duration represents a snapshot and can be too short. Thus, we aggregate the values of the past two weeks.

5. EVALUATION

Our preliminary evaluation focuses on two aspects of the multi-trust algorithm. First, we validate that the two-level rank metrics is capable of boosting the coverage significantly beyond what a private-history solution can offer. Second, we show that our mechanism deals with the colluders as effectively as EigenTrust does while reducing the side effects such as leg-hugger and unfair punishment of local distributor within satellite users.

5.1 Coverage experiment

We used one month traffic log to study how the multi-trust algorithm improves coverage. Applying such policy will alter the traffic pattern, as source peers will give preference to those fall in M, and then in M^2. The downloading peer will likewise modify their rank matrix accordingly. Also, the complete log is too large to be simulated. The amendments we made are:

- We simulate the 0.84% most popular files which account for more than 88% of total traffic. This makes in-memory simulation possible.
- The indexing is perfect and all the replicas are always available when the request is made. A more detailed simulation will correlate the request time with the node churn log.

These simplifications will bring in noises, however our goal is to have a qualitative understanding of the net effect. The simulation picks one request, sends it to all available replicas, each of which will upload equal share to the requester. The metrics that we use is called the *trust upload ratio*, which is the volume of traffic that are served by either a M friend, or a friend that is either a M or M^2 friend, over the total traffic in the system. And we plot both curves as time progresses. The difference between the two curves give an idea on how much coverage is expanded by one more level.

In Fig. 4, the x-axis is the wall clock, whereas the y-axis gives the upload traffic ratio for both M and $\{M, M^2\}$. As expected, both curves rise with time. However, M stays mostly at around 3%, showing the problem of purely private-history based approach. M^2, on the other hand, increase to about 60%. What that means is that the rest of the 40% traffic is left to be handled by EigenTrust and subject to its problems.

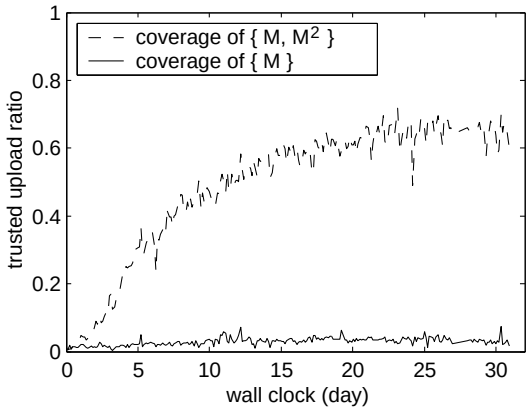

Figure 4: Coverage experiment for one month long simulation.

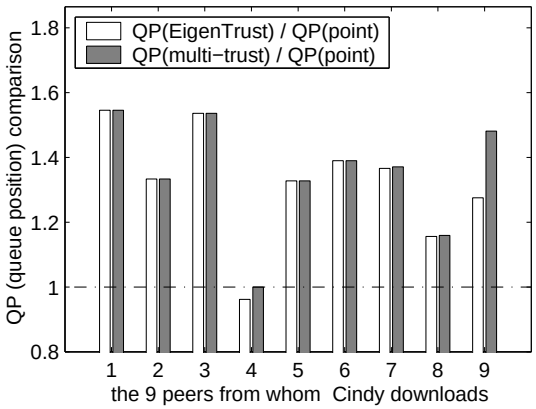

Figure 5: Pair-wise colluder Cindy.

5.2 Leg-hugger and satellite cluster experiment

Since our algorithm does not give a global value like EigenTrust, we must construct our own metric to quantify preference between peers. In this experiment, we illustrate peer preference by subjectively ranking the values of downloaders from the perspective of the uploader. We generate the ranking value in two steps. First, we use the completed transactions in our one month traffic log to calculate a set of ranking values for each peer. Next, the evaluator node extracts download requests for the following two week period, and statistically sorts the requesters into a service queue according to their local subjective ranking. Peers with lower queue positions are served first, i.e., higher ranked. Our result uses the peers' queue positions as a metric to compare each incentive algorithm. Our expected results are that true colluders will have queue positions no earlier than EigenTrust, whereas local distributors will move up.

5.2.1 Colluder punishment

In our first experiment, we choose one pair-wise colluder and spam account colluder, and verify that multi-trust performs as well as EigenTrust. This experiment includes two target peers, pair-wise colluder Cindy and spam account colluder Ingrid.

Pair-wise colluder: (Fig. 5) Cindy downloads from 9 peers during the next two weeks. As we see from the figure, there are 7 peers punish Cindy in multi-trust as well as Eigen-

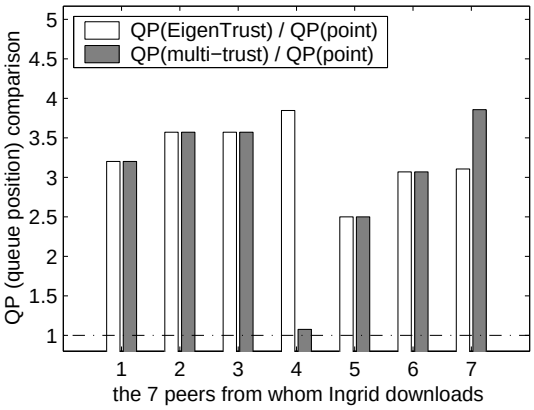

Figure 6: Spam account colluder Ingrid.

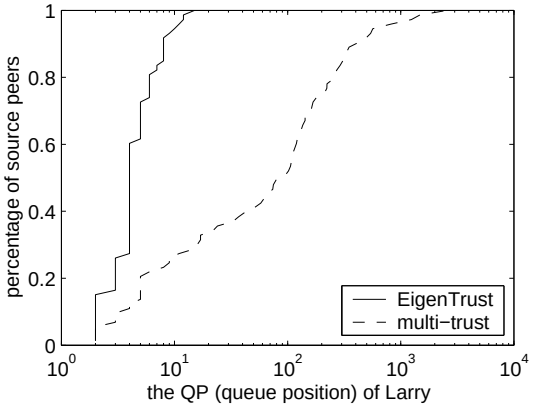

Figure 7: CDF of queue position for the leg-hugger Larry, for both EigenTrust and multi-trust.

Trust. On the other hand, multi-trust allows for both peer 4 and peer 9 to punish Cindy more than EigenTrust. Detailed analysis show that friends of these two peers moved ahead, downgrading Cindy as a result.

Spam account colluder: (Fig. 6) Ingrid downloads from 7 peers during the next two weeks. There are 5 peers punish Ingrid in multi-trust as well as in EigenTrust. Peer 7 punishes Ingrid more in multi-trust than in EigenTrust. Peer 4 is one exceptional case. Peer 4 has downloaded 29MB data from Ingrid, as such it is reasonable for it to get ahead.

5.2.2 Curing the problem of EigenTrust

As we discussed earlier, EigenTrust suffers from false-negative (i.e. the problem of leg-hugger) and false-positive (i.e. unfair punishment of local distributor). We select one leg-hugger peer Larry and another satellite cluster local distributor peer Wayne. EigenTrust gives them unexpected high and low rank.

Leg-hugger (Fig. 7) Larry downloads from 73 peers in the two weeks following our one month traffic log. It gets high rank in the EigenTrust because of his high rank friend. In multi-trust his high reputable friend still helps him getting into 16 (22%) peers' trust list, but overall he is punished in all the other peers. Although multi-trust does not completely solve this problem, the problem is alleviated by 78%. In the case it is punished, its queue position is increased by more than a factor of 10.

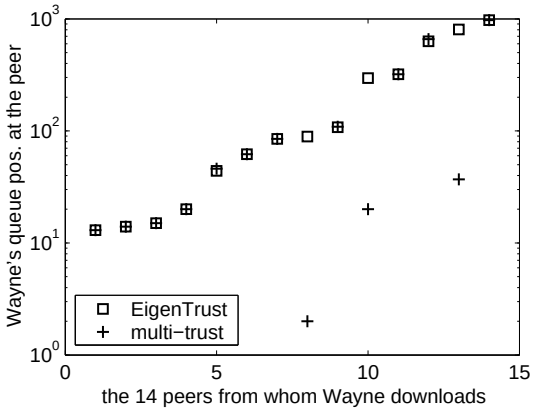

Figure 8: Local distributor Wayne. The x-axis is ordered so that EigenTrust QP is monotonically increasing.

Satellite cluster (Fig. 8) Wayne is the local distributor in one cluster. It downloads from 14 peers in the next two weeks, among them there are 11 external peers and 3 internal peers (peer 9, 10 and 13). He is punished by all download sources in EigenTrust because of his low EigenRank. With multi-trust, the two internal peers 10 and 13 give him higher priority. Peer 9 is new to the system and its M consists of only 3 peers, as such it has not had enough history to elevate Wayne from its perspective. Although all external peers rank Wayne fairly low, one of them (peer 8) has directly downloaded from Wayne from and thus gives its high rank, illustrating the advantage of the tiered design.

The simulation results show that, multi-trust performs no worse than EigenTrust in punishing pair-wise or spam account colluders. Meanwhile, it solves the two problems brought by EigenTrust, by dropping the high rank of leg-hugger peer and rising the low rank of local distributor inside its own satellite cluster.

6. CONCLUSION

Designing a robust incentive algorithm for P2P system is a challenging research issue. Using the real trace from the deployed and large P2P network Maze, we present several collusion patterns found in Maze. We believe that the existing solutions such as private history-based Tit-for-Tat and the EigenTrust algorithm each have their own pitfalls. We propose the multi-trust algorithm as a hybrid that achieves the best balance.

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 Proc. of WWW, April 1998.
[3] B. Cohen, "Incentives Build Robustness in BitTorrent," In Proc. of P2P-Econ, 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 Proc. of IPTPS, Cambridge, MA, 2002.
[6] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust Incentive Techniques for Peer-to-Peer Networks," In

Proc. of EC, May 2004.
[7] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, "Free-Riding and Whitewashing in Peer-to-Peer Systems," In Proc. of ACM PINS, August 2004.
[8] P. Ganesan, M. Seshadri, "On Cooperative Content Distribution and the Price of Barter", In Proc. of ICDCS, June 2005.
[9] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks" In Proc. of WWW, May 2003
[10] Y. Kulbak, and D. Bickson, "The eMule Protocol Specification", eMule project, http://sourceforge.net
[11] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li, "An Empirical Study of Collusion Behavior in the Maze P2P File-Sharing System," Microsoft Research Technical Report, MSR-TR-2006-14, 2006.
[12] T. Moreton, A. Twigg, "Trading in Trust, Tokens, and Stamps," In Proc. of P2P-Econ, Berkeley CA, June 2003
[13] A. Nandi, T-W. Ngan, A. Singh, P. Druschel, and D. S. Wallach, "Scrivener: Providing Incentive in Cooperative Content Distribution Systems", In Proc. of Middleware, Grenoble, France. Nov.-Dec. 2005
[14] S. J. Nielson, S. A. Crosby, and D. S. Wallach, "A Taxonomy of Rational Attacks" In Proc. of IPTPS, Ithaca, NY. February 2005
[15] A. Nandi, T-W. Ngan, A. Singh, P. Druschel, and D. S. Wallach, "Scrivener: Providing Incentive in Cooperative Content Distribution Systems", In Proc. of Middleware, Grenoble, France. Nov.-Dec. 2005
[16] B. Wilcox-O'Hearn. "Experiences deploying a large-scale emergent network." In Proc. of IPTPS, Cambridge, MA, 2002.
[17] B. Wu, Brian D. Davison, "Identifying link farm spam pages," In Proc. 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 Proc. 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 Proc. of IPTPS, Ithaca, NY. February 2005.