Improving Cooperation in Peer-to-Peer Systems Using Social Networks

Wenyu Wang[1] Li Zhao[2] Ruixi Yuan[3]

Center For Intelligent and Networked Systems
Tsinghua University, Beijing, China, 100084
{[1]wangwengyu, [2]zhaoli04}@mails.tsinghua.edu.cn
[3]ryuan@tsinghua.edu.cn

Abstract

Rational and selfish nodes in P2P systems usually lack effective incentives to cooperate, contributing to the increase of free-riders, and degrading the system performance. Various attacks such as whitewashing, collusion, and software cracking pose great challenges on distributed reputation management. To tackle these problems, we propose to build a social network on P2P system, and use the strength of social connections to facilitate transactions in P2P system. The' small world' character of social networks makes it feasible for nodes to locate resources and conduct transactions while maintain limited local memory history. Such distributed memory combined by relationship between peers constructs a powerful reputation management network, which could have better performance than shared history system and is more robust under various attacks. Our simulation and analysis show that the social network model can greatly incent cooperation in P2P networks and enormously reduce the memory cost.

1. Introduction

The peer-to-peer model may facilitate more efficient resource sharing in the networks. Applications based on such model include file-sharing system (such as Gnutella[1],Kazaa[2] and BitTorrent[3]), discussion boards[4] and overlay routing[5] etc. The key problems that confront all these systems come from the disincentives of peers to cooperate. Cooperation consumes peers' resource and degrades their performance. Since rational and self-interested peers [6] always try to maximize their own interest, most of them would refuse to supply service to others without incentives (either direct or indirect). This behavior may benefit themselves in the short term, but the "tragedy of commons" [7] appears when most of peers in the network choose to avoid cooperation, which leads to the collapse of the whole system.

This work is partially sponsored by Tsinghua University through Grant No. 052301001

Incentive mechanism in P2P networks have been intensively discussed in recent researches [3][8][9]; both centralized and distributed mechanisms have been proposed to incent cooperation between peers. The free-riding [10] in P2P network is the main cause of "tragedy of commons", and it could avoid punishment with the aid of other misbehaviors such as whitewashing and collusion. Difficulties of discovering these "bad guys" have been summarized in [8], including: large population and high turnover; asymmetry of interest and zero-cost identity.

The memory/resource limitation of individual nodes further aggravates the problems, so the direct causes of these misbehaviors should also include:

Lack of history: Since the population of peers in P2P network is usually very large [11], and transactions between peers are so frequent and dynamic, it is not feasible for peers to remember all those who have had transactions with them in the history.

Unawareness of others: Peers in the network have no idea of transactions of others. So "bad guys" could succeed in exploiting resource from different "good guys".

Though some previous works [3][8][9][12][13] have proposed promising solutions to these problems, they have to rely on some information such as globally shared history [8] or centralized trust mechanism[12][13]. Shared history requires that each peer keeps records about all of the interactions that occur in the system, no matter whether it involves or not. It successfully solves the problems brought by the lack of history and unawareness of others. However, it would be difficult to have each node notified of all the transaction it is not involved with, and the maintenance cost of such information with large population can be very high. Centralized trust mechanism costs as high as that of shared history, and reduces the robustness of the whole system as well.

Interaction in peer-to-peer networks, in some aspect, is similar to interaction in real world communities, where people could make friends with those who have the same interest or helping each other with common good. The "six degree between two Americans" effect [14] indicates that such social network is advantageous for information spread and relationship maintenance. Therefore we propose an incentive mechanism based on social network, using the "small world" character to tackle the problems encountered in P2P networks. This approach allows peers to establish their own circles of friends and make use of these friends to share resources. Social networks have been shown to perform well in resource searching in P2P networks [15], and here we use it to incent cooperation between friends. When most pairs of friends are willing to cooperate, the "small world" character [17][18] will lead to a favorable environment for resource sharing. This distributed history stored in each node costs little space and we will show that the whole system is robust even when the memory stored locally get lost or being modified. The social network approach provides a promising solution to the defects of unstructured P2P networks. Some of the beneficial properties of this approach include:

Consignable requests: When one peer has a request, it will ask its friends first, if the resource is available (at least one of its friends has the resource and is willing to serve), it will get the resource from the server; if not, the requesting node will consign the requests to its friends who will look for the resource in their coteries. Though transactions between fixed peers still rarely repeat, the consignable requests between two fixed friends frequently happen.

Transferable interest: When a successful resource sharing transaction is completed, all the peers that participates the transfer will know the contribution of each other. In this case, the effect of asymmetric interest is limited.

Friendship rebuilding cost: It is costly to build a coterie for a new-comer. An effective coterie calls for much effort, so the cost for whitewashers who always change their identities will be very high, hence discouraging such misbehavior.

Distributed/Local historical memory: Though the global memory of history is not available in our design system, each peer in the network has remembered the transactions between him and his friends. Since the requests and service of the peers mostly happen inside the coterie in our algorithm, peers could use the history for reference. The problem "unawareness of others" could also be alleviated.

The rest of this paper is organized as follows. We describe our model in Section 2. Then we discuss the evolution of social networks in Section 3. Possible attacks and preventions are proposed in Section 4. Simulation experiments are used to validate our mechanism in Section 5.

2. Social network model

2.1 Assumptions

First, we assume that each peer in P2P networks is rational but has no complete information: It also has no idea of the topology of the whole network. All the peers will act rationally to maximize their own interest according to the information it has. The whole network is dynamic, which allows existing nodes to exit and new nodes to join. Once a node chooses to exit, its ID is destroyed and won't be used again. All the new comers have no friends at the beginning; new connections could be established during the transactions. Once two nodes become friends, the peers can properly authenticate each other. We do not need any centralized authority to manage the identity and trust among peers. Hence it is also possible for the data on a node to be unreliable (either due to software cracking, or misbehaving).

2.2 Model

Our model is built upon graph theory, which is widely used in modeling social networks [14]. Let $\mathbb{S} = \{1, 2 ... N\}$ be the set of peers in the P2P system, which is also the nodes in a graph. We use directed graph to characterize the whole system. The friendship between two peers is represented by the arcs (links) between the two nodes. As shown in Figure 1, each node in the directed graph assigns two directed links to each of its friends, called *Credit Degree* and *Payment Degree*. Let C_{ij} denote the *Credit Degree* (the credit **i** has obtained from **j**, which is equivalent to the total service performed by **j** for **i**) which node ***i*** assigns to its friend node ***j***, and P_{ij} denote the *Payment Degree* (the payment made by **i** to **j**, which is equivalent to the service performed by **i** for **j**) from node ***i*** to node ***j***.

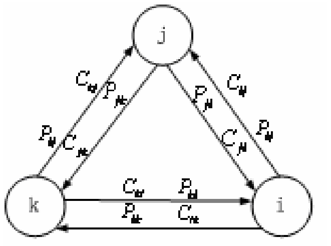

Figure 1. **The relationship between nodes, represented by the Credit Degree and the Payment Degree.**

2.3 Incentive mechanism

When one node in the graph has requests, it will ask its friends for help. Whether it could be satisfied or not depends on how effective its coterie is. Two main steps are proposed to incent and maintain cooperation between peers.

1) Consignable request

Similar to the "expanding ring" concept introduced in [19] in Gnutella, when a node in the network has a request, it will first send it to its friends. These friend nodes check if they own the requested resource and wish to serve. They will notify the requesting node if so and the transaction will occur. If none of the friends could serve the requests (no one have the resource or the owners refuse to cooperate), consignable requests occur.

The node will consign the requests to its friends, and set a count parameter (denoted as TTL) equal to 2, which allows the search within two-hop range. The friends who accept the consignation will search the resource in their own coteries. If the resource is available, the friend will route the data between the initial requesting node and the server. If none could serve the requests, the initial requesting node will increase the TTL to 3, and then the requests are iteratively consigned. The value of TTL will increase until the demanded service is available or the TTL reaches a maximum value which has been set beforehand. Since long distance iterative consignation is not robust and it may bring heavy burden to the overall performance of the whole network, the maximum value of TTL is used to limit the largest length of transaction path. When the TTL has reached the maximum value and the resource is not available, the initial requesting node has to turn to the index servers or so to locate the resource. The increase of TTL guarantees the shortest path for transaction, and when more than one path have the same TTL; the requesting node has to decide which one to use. We propose the following server selection strategy:

We define strength of friendship from node ***i*** to node ***j***

$$F_{ij} = C_{ij} + P_{ij} \quad (1)$$

Then the balance of friendship from node ***i*** to node ***j*** is

$$B_{ij} = (P_{ij} - C_{ij})/(P_{ij} + C_{ij}) \quad (2)$$

The strategy makes node ***i*** to select the node ***j*** with the highest B_{ij} score to serve the request. High B_{ij} score means that node ***j*** owes much to node ***i***, so choosing the node ***j*** with high B_{ij} will give node ***j*** a chance to pay back, and increase the strength of the friendship. At the same time, a high value of F_{ij} is favorable because of a stronger relationship.

A case of Consignable request is showed in Figure 2

Node 1 has a request and Node 9 could serve. First, Node 1 asks its friend Node 3, and Node 3 has no such resource. So Node 1 consigns the request to Node 3 and set the TTL equal to 2. If Node 3 accepts the consignment, it will ask its friends including Node 2, Node 4 and Node 5. Since none of them have such resource, Node 3 has to again consign the requests to Node 2, Node 4 and Node 5. In our case, Node 2 and Node 4 refuse to accept the consignable requests, and Node 5 accepts it. As a result, TTL is set to 3, and Node 6 and Node 7 are asked about the requests by Node 5. If the maximum TTL is equal to 3, the consignable requests fail because Node 6 and Node 7 have no such resource, and Node 1 has to turn to other strangers for help. If the maximum TTL is larger than 4, Node 5 will then consign the requests to Node 6 and Node 7. Suppose both of them accept the consignment, Node 6 and Node 7 will visit Node 9 where the requested resource is available. If Node 9 agrees to serve for both nodes, two feasible shortest paths have been found.

Path_1 = {1,3,5,6,9}, Path_2 = {1,3,5,7,9}

It is up to Node 5 to decide which path to take ultimately, if the *Credit Degree* and *Payment Degree* are $C_{56} = 30$, $C_{57} = 25$; $P_{56} = 25$, $P_{57} = 30$, our strategy will choose Path_2, for $F_{65} = F_{67}$ and $B_{57} > B_{56}$

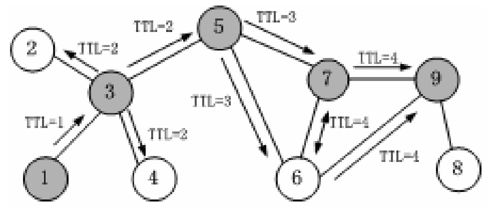

Figure 2. **A typical process of consignable request.**

2) Transferable interest

When a shortest path has been found, all the nodes on this path have to route the data. Though a direct data transfer from the serving node to the requester seems more effective from the aspect of bandwidth, the Credit and Payment Degree between nodes is hard to handle in our algorithm if they do not participate the transfer. We will further discuss the performance problem in Section 6. Consignable requests make the interest between the initial requesting node and the server visible to all the nodes on the path, and the interest becomes transferable in this case. For instance, in Figure 2 we choose $Path\_2$ to transfer the data, and the Node 3, Node 5 and Node 7 will route the data for this transaction. In this case, Node 1 will acknowledge Node 3 for routing the data, and the same kind of acknowledgement happens from Node 3 to Node 5, Node 5 to Node 7 and Node 7 to Node 9. As a result, the interest between Node 1 and Node 9 has been transferred to interest between each pair of nodes along the path. So the effect of asymmetry of interest is turned around to strengthen the friendship along the path. Once a transaction is finished, all of the nodes on the path will update their own *Credit Degrees* and *Payment Degrees*. The amount of increase depends mainly on the size of data (it will be discussed in Section 3).

3) Decision function

Peers decide whether to provide resource or accept consignable requests according to the output of decision function when requests happen. A balanced relationship between node ***i*** and node ***j*** means that $B_{ij} = (P_{ij} - C_{ij})/(P_{ij} + C_{ij})$ is close to 0.

We define a decision function for each peer, and the output of the function is a probability deciding whether node ***i*** should supply service to node ***j*** if it could. For a rationale node, this function needs to have two properties: the probability lies within [0,1]; and the probability is a decreasing function of B_{ij}, as it tries to repay the service provided by others. We choose a decision function as follows:

$$D_{ij} = \frac{1}{2}(1 - \sin\frac{\pi B_{ij}}{2});\ B_{ij} \in [-1,1],\ \text{so}\ D_{ij} \in [0,1]$$

D_{ij} is a decreasing function of B_{ij}. In addition, $B_{ij} = 0$ which means a complete balance between two nodes, results in a probability of 0.5 to serve. Our decision function proves to be effective in the simulation result outlined in section 4.

3. Evolution of social networks

As people in society, each peer in P2P system has no fixed coterie. They will strengthen the relationship with those who are effective in providing resource, and weaken the ties with those who rarely supply service. In this way, "good" nodes will receive a good reputation through the transactions and extend their coterie; while "bad" nodes will be discredited by their friends (if any) for their bad behavior. The main evolution of social networks comes from two aspects: creating new connections and severing old connections.

3.1 Creation of new connections

Since each individual node could not know all the peers in the networks and have no idea of the topology of the system as a whole, the creation of new connections is not arbitrary, but mainly depends on who they meet. We propose two ways to discover new friends: create new connection through successful transaction and through strangers.

1) Through successful transaction

Successful transaction between two peers is a main way to create new connections. It is favorable compared with creating new connections with strangers. The reasons are as follows:

Low risk of accepting free-riders. Successful transaction serves as an evidence of generosity. For the requesting node, it believes that the server node is generous because of the service; for the server node, the consignable requests carried by its friends prove a good reputation of the initial requesting node (if the initial requesting node is a free-rider, consignable requests will hardly be accepted according to the decision function). As a result, the risk of accepting a free-rider is greatly reduced after a successful transaction for both sides.

Shortcut of transactions Long path in the transaction brings about vulnerability, because any failure of cooperation between two nodes on the path could result in the failure of the transaction. For instance, in Figure 2, if any cooperation between Node 1 and Node 3, Node 3 and Node 5, Node 5 and Node 7 or Node 7 and Node 9 fails, the transaction could not succeed. If Node 1 and Node 9 could establish a tie after the transaction, the length of path will be reduced to 1 when transactions happen again between Node 1 and Node 9, and what is more, the distance between Node 1 and Node 7 is also reduced to 2 (through Node 9) from 3 (through Node 3 and Node 5). This shortcut will also facilitate further transactions through this path.

Though it is advantageous to establish new connections after successful transaction, they are not always profitable. New connections cost resource to maintain and are relatively weak at the beginning. At the same time, each node could have a limitation of the maximum number of friends it could maintain. We suggest a probability function to decide whether the new connection could be built or not. This function is:

$$u_{ij}(g) = \min(1 - \frac{curF_i}{\max F_i} + \frac{0.5 \times (h_{ij}(g) - 1)}{\max Hop}, 1) \qquad (3)$$

$u_{ij}(g)$ denotes the probability of building a new connection between Node i and Node j; $\max F_i$ denotes the maximum number of friends node i could maintain; $curF_i$ denotes the current number of friends node i has; $h_{ij}(g)$ denotes the distance between Node i and Node j. $\max Hop$ denotes the maximum number of TTL allowed in Consignable requests

As we know, there are two factors which could affect the probability of building new friendship. The length of transaction path decides to what degree the new friendship is valuable and benefit to the whole system. The new friendship contributes more when the transaction path is long. The current number of friends decides the willingness of nodes to build new friendship. When the friends of node ***i*** are few, it is eager to make new friends to enhance the efficiency for resource sharing. But when the number of friends nearly reaches the maximum number, it may have a disincentive to make more friends. Our heuristic proposal considers both of the two parameters: $u_{ij}(g)$ is a decreasing function of $curF_i$, and an increasing function of $h_{ij}(g)$. In order to make the probability function proper, we limit the output to values between [0,1] . While we recognize such formulation is relatively simple, it works surprising well in our simulation.

2) Through stranger policy

Punishing strangers has been proven necessary to constrain white-washers [16]. However, sometimes strangers are unnecessarily punished heavily. Hence, we adopt an adaptive policy to strangers using the information from the coterie. Each node has a list of new connections established recently with strangers, recording the number of services and requests of these new friends. When a node receives a request of new connection from a stranger, it will ask its old friends for reference. Each of these old friends will send a value of stranger contribution to it, and the node calculates a mean contribution of strangers according to the formula as follows:

$$V_i = \frac{1}{2} w_i + \frac{1}{2}[(\sum_{j=1}^{J} C_{ij} w_j) / (\sum_{j=1}^{J} C_{ij})] \qquad (4)$$

V_i denotes the mean contribution of recently encountered strangers; w_j denotes the mean contribution of strangers sent to Node i by its friend Node j. So the first item on the right of the formula means the contribution of strangers calculated by node ***i*** itself according to the stranger list it keeps. The second item means a weighted mean value of contribution sent from its coterie. The weighted mean strategy makes generous nodes' suggestions more convincing, while free-riders' suggestions always have little impact. In this way, collusion could be effectively confined. Each w_j is calculated as follows in Node j:

$$w_j = \frac{1}{n}\sum_{k=1}^{n}[C_{jk}/(C_{jk}+P_{jk})] \qquad (5)$$

Where n is the total number of strangers the node ***j*** has interacted within a period of time recently, and w_j is the mean contribution of these n nodes. Clearly, $w_j \in (0,1)$ .This method of using the weighted mean contribution of strangers to evaluate the current requesting stranger helps the node to decide whether to accept the strangers' requests or not. When the number of free-riders is large in the system, most of friends may report a low value of stranger contribution, so the V_i will decrease to a low value too, suggesting that refusal is a wiser choice.

A threshold V_T is set beforehand, if V_i is greater than V_T, the request of new connection is accepted and if V_i is small, the request will be turned down.

3.2 Severance of old connection

The need to sever old connections comes from many aspects. We propose a method using both history record and short-term transactions to address this problem. Each time a node ***i*** initiates a request, it will update the *Credit Degree* for its friends according to the formula as follows:

$$C_{ij} = \begin{cases} C_{ij} - \Delta C & j \in S_1 \\ C_{ij} + \Delta S & j \in S_2 \end{cases} \qquad (6)$$

S_1 denotes the set of nodes which do not supply service to node ***i*** . S_2 denotes the set of nodes which supply service to node ***i***, and it is empty if no one serves. ΔC denotes the decrease of *Credit Degree* for these nodes which have not supplied service , and it is called "leak rate of *Credit Degree*". ΔS denotes the increase of *Credit Degree* for the node which has supplied service. We have $\Delta S > \Delta C$ to make a relative stable friendship in a short term. The value of ΔS could depend on the size of resource node ***j*** has supplied to node ***i*** or set to a constant value just as we did in this paper.

The *Payment Degree* has similar "leak" mechanism as *Credit Degree*. Each time node ***i*** serves its friend node ***k***, it will update the *Payment Degree* as follows:

$$P_{ij} = \begin{cases} P_{ij} - \Delta P & j \neq k \\ P_{ij} + \Delta S & j = k \end{cases} \qquad (7)$$

ΔP denotes the decrease of Payment Degree for these nodes which do not request, and it is called "leak rate of Payment Degree". ΔS denotes the increase of Payment Degree for the node which have received service. A threshold C_T is set to monitor the friendship between nodes, when the *Credit Degree* C_{ij} falls below C_T , that is $C_{ij} < C_T$, node ***i*** will sever the connection ***ij*** and remove node ***j*** from its friend list. We note that it is possible for a node ***j*** to consider ***i*** as its friend, but ***i*** may not consider ***j*** as its friend.

4. Robustness against attacks

Our incentive mechanism is robust against wide range of attacks including free-riding, whitewashing, collusion, self-boasting etc. It is a distributed reputation management scheme using social networks model.

Disincentive of free-rider and whitewasher

It is difficult for free-riders and whitewasher to create friendship links under our system. In the unlikely event a connection is made, it will quickly be severed because the credit of free-rider or whitewasher will be exhausted under our "leak" mechanism.

Prevention of self-boosting

Self-boosting is first reported in [11]. This problem is serious and difficult to detect. However, in our system, it is useless to do so. Self-boosting can not change any values of Credit Degree and Payment Degree, so has no effect on our system.

Prevention of collusion

The most popular collusion is the case that several "bad guys" claim to receive service from each other. This kind of collusion is simple, but very difficult to prevent in most P2P systems. The social networks we built can effectively prevent the collusion. See the example in Figure 3. The gray nodes are "bad guys" and the white nodes are common ones. Collusion makes the link between "bad guys" strong, but it cannot strengthen the link between the normal nodes and the "bad guys". For example, in Figure 3, C_{12}, C_{74}, C_{84} are all very small with our mechanism. As a result, the normal nodes will not supply resource to these colluded "bad guys", and the latter are isolated as a whole finally. In some cases, there can be a mole node [8] that serves as a link between the normal nodes and the group of colluders. Suppose Node 2 in dark gray is a mole node, and vouches for the good reputation of Node 3, Node 5 and Node 6. In this case, Node 3, Node 5 and Node 6 could exploit the resource of Node 1 through Node 2(the mole). However, the exploitation is based on the increase of P_{12}, and Node 1 will soon refuse to further supply resource when P_{12} becomes large while C_{12} remains the same. lates the colluders quickly.

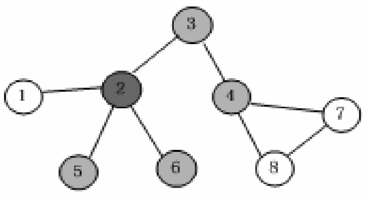

Figure 3. Nodes with different characters in a network. All nodes in gray are free-riders who collude.

Robustness against software crack

Some distributed P2P system meet difficulty in reputation management when crack occurs [20]. When the data used for reputation management is modified by the users, it fails to keep effective. The reputation of peers in our system are not kept locally, but kept by the coteries. So the modification of local data could bring no interest. For instance, in Figure 3, if Node 2 crack the software and modify local data C_{21}, P_{21}, the decision of Node 1 that whether to serve Node 2 will not be affected because this decision function is only rely on C_{12}, P_{12}, stored in Node 1 and could not be modified by Node 2.

5. Evaluation and result

5.1 Experiment framework and parameters

All the peers in the simulation have three kinds of strategies: 100% cooperate, 100% defectand using decision function (described in Section 2.3)). Similar to the learning behavior described in [8], we assume that a node can find whether its strategy is yielding profitable results and will switch its strategy probabilistically based on the comparison. Default values of parameters in simulation are presented in Table 1, and part of parameters will be set to special values in different kinds of attacks. We implement the model described in Section 2 and Section 3 in simulation, and evaluate the performance of our algorithm in different scenarios of attacks. We set the initial friend list of each node empty at the beginning for the worst case in simulation.

Table 1. Default values of parameters in simulation

Parameter	Default	Section
Population size	100	--
Run Time (rounds)	1000	--
Initial ratio of "100% cooperate"	1/3	5.1
Initial ratio of "100% defect"	1/3	5.1
Initial ratio using decision function	1/3	5.1
Learning Probability	0.05	5.1
Turnover Probability	0.0001	5.1
Credit Degree for new friendship	30	3.1
Payment Degree for new friendship	30	3.1
Leak rate of Credit Degree	2	3.2
Leak rate of Payment Degree	2	3.2
Increase of C or P (delta C and delta P)	40	3.2
Maximum number of friends	10	3.1
Maximum TTL allowed	5	2.3
Severance threshold	10	3.2

5.2 System Evolution and Performance

1) Population using different strategies

While initially setting the population using different strategies (cooperate, defect and decision function) as equal, the learning behavior changes the user's strategy overtime. Figure 4 shows the population evolution of users using different strategies. Though the random selection of learning behavior causes the population size to oscillate in the first 200 rounds, the strategy with decision function will dominate at last.

To illustrate the efficiency of our mechanism in isolating the free-riders, we let the peers use fixed strategy without learning behavior in Figure 5. We set 10 nodes with 100% defect strategy and 90 nodes with decision function strategy. The result after 1000 rounds of simulation is shown in Figure 5. Clearly the free-riders are isolated after several hundreds rounds.

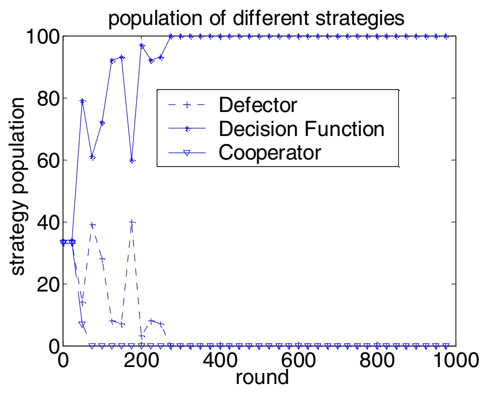

Figure 4. The strategy with decision function dominates after 300 rounds, and the population of 100% defector and 100% cooperator fall to zero. The evolution of population with different strategies may vary at beginning in repeated simulations, but the decision function always dominates at last.

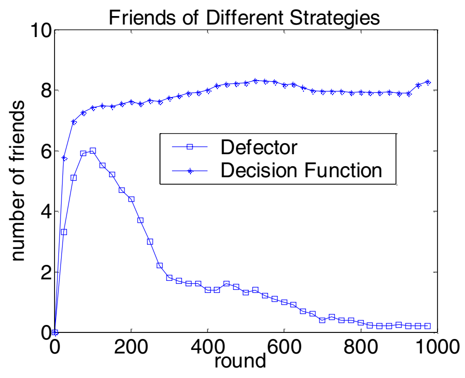

Figure 5. The mean number of friends grows up at similar rate for both defector and decision function strategy when most peers have little information of others at the very beginning. The defectors will quickly be isolated when most good peers have formed their own coteries.

2) Collusion and false report

M. Feldman et. al.[8] uses MaxFlow to prevent collusion, and propose a proximate algorithm to reduce its running time. Our algorithm, in fact, has completed this goal when searching the service and no more calculation is needed. Collusion could not work in our system because of the friendship bottleneck between the common peer group and the free-rider group. We consider a worst case in which each free-rider node claim that other free-rider nodes are generous ones. At the same time, they do not supply any service to common nodes. Still, we set 10 nodes with 100% defect strategy and 90 nodes with decision function strategy, and request all nodes perform fixed strategy. We illustrate the total friends of these collusive defectors and the number of defectors in these friends in Figure 6.

In the case of collusion, we discover that the mean number of friends of these collusive defectors becomes large, but nearly all of the friendship is among collusive defectors. Hence the link between common nodes and defectors is very weak, making collusion unsuccessful. Clearly, the enhancement of friendship among defectors only could not change their payoff received.

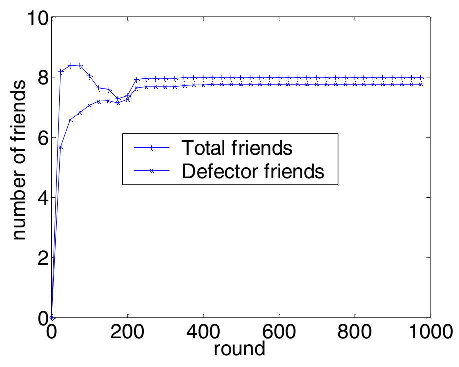

Figure 6. The mean number of friends of these collusive defectors could reach 8, but nearly all of these friends are collusive defectors as well.

3) Comparison with previous works

Algorithm using shared history and private history are compared with our algorithm here. We mainly focus on the performance of our algorithm in a P2P network with high turnover rate and large population. We use *the mean rate of satisfied request* (MRSR) as the indicator to reflect the overall cooperation. The mean rate of satisfied request is calculated as:

$$MRSR = \frac{Number\ of\ satisfied\ request}{Number\ of\ total\ request} \quad (8)$$

MRSR is the mean rate of satisfied request. MRSR reflects overall cooperation. We compare the MRSR as the population grows between different incentive mechanisms in Figure 7:

The private history fails to incent cooperation when the population grows to more than 200, and it is resulted from the asymmetric interest. Shared history and social network approach both perform well even when the population becomes large.

High rate of turnover is a serious problem in P2P networks. We compare the performance of different algorithms when the turnover rate becomes large.

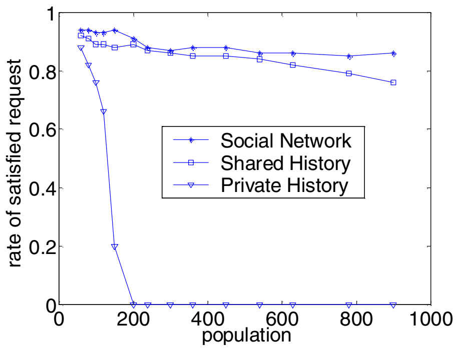

Figure 7. The MRSR with private history suffers a sudden drop when the population grows more than 200. MRSR with shared history or social network performs well when the network extends.

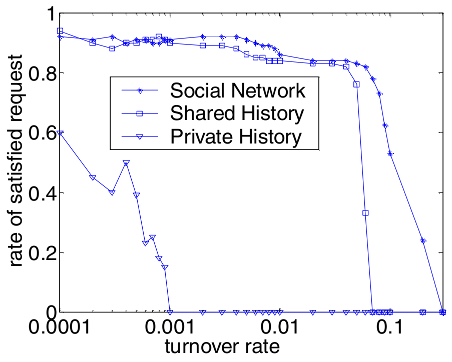

Figure 8. The private history could tolerate a very low turnover rate. Shared history and social network could keep a high MRSR with high turnover rate.

Figure 8 shows that private history could only tolerate a turnover rate lower than 0.001 and the shared history is much better than the private history when the turnover rate is <10%, but suffers a sharp drop-off afterwards. Our algorithm suffers a similar decrease when the turnover rate grows to more than 20%. It may be a little surprising that shared history which has global information performs not as well as our algorithm. In fact, the problem comes from the algorithm based on shared history instead of the shared history itself. Most algorithms based on shared history set the probability to serve strangers artificially low, so the newcomer would meet great frustration in joining the system. Our algorithm based on social networks alleviates these problems. First, the probability to accept strangers is also low in our algorithm, but once the newcomer has been accepted, the sequent requests will have a relative high probability to be met if the newcomer acts like a "good guy" with the help of consignable requests. Second, not only the nodes serving resource will get acknowledgement, but also those nodes that route the data. In this way, the newcomer can quickly join the system as long as it is willing to help route the data.

6. Conclusion and Discussion

In this paper, we propose to build a social network in peer-to-peer system to incent cooperation between peers. To our knowledge, this is among the first work to incorporate the concept of social network into the P2P system for incentive mechanism design. The proposed algorithm proves to be robust in a large population and relative high turnover rate with simulation analysis. The local storage memory in our algorithm is greatly reduced compared with shared history in many previous works, and is a truly distributed memory.

One potential problem for this incentive mechanism lies on the repeated transfer of data along the transaction path, as it may impose extra burden on system performance and network bandwidth if the mean length of path is long. To evaluate the influence, we measured the mean length of path in the simulation experiment. For totally random resource request and service distribution, the average length of path is about 2.5 due to the limitation of maximum TTL (section 2). This is also indicative of the small characteristic length of random networks. Furthermore, if we consider the small world property of social network, where the cluster coefficient is large compared with that of the random networks, we can expect further reduction to the average path length. Hence lessen the penalty for repeated transfer path. An obvious short-cut solution is to allow the serving node and the requester to perform direct file transfer. However, this will alter the credit and payment calculations and make the reciprocity hard to perform. Its overall effects on the formation of social networks and the incentive mechanisms can be further investigated.

7. Future work

Quantitative comparison of the maintenance cost in social network approach with other mechanisms may promise to yield more impressed results in future work. At the same time, the advantages of social network approach in both incentive mechanism design and resource search algorithm should be combined together to achieve higher performance.

REFERENCES

[1] Gnutella, http://www.gnutella.com.

[2] KaZaa, http://www.kazaa.com.

[3] The official BitTorrent home page, http://www.bittorrent.com

[4] Gu. B, and Jarvenpaa. S, "Are Contributions to P2P Technical Forums Private or Public Goods? – An Empirical Investigation," In 1st Workshop on Economics of Peer-to-Peer Systems, 2003.

[5] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and DS Wallach, "Security for Structured Peer-to-Peer Overlay Networks." In Proceedings of Multimedia Computing and Networking, 2002

[6] Jeffrey Shneidman, D. C. P, "Rationality and Self-Interest in Peer to Peer Networks," In Proc.2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03), 2003.

[7] Hardin, G. "The Tragedy of the Commons." Science 162:1243-7, 1968.

[8] Michal Feldman, Kevin Lai, Ion Stoica, John Chuang. "Robust Incentive Techniques for Peer-to-Peer Networks." ACM E-Commerce Conference (EC'04), 2004.

[9] J. Crowcroft, R. Gibbens, F. Kelly and S. Östring " Modeling Incentives for Collaboration in Mobile Ad Hoc Networks." In Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2003.

[10] E. Adar and B. A. Huberman, "Free riding on gnutella." 2002.

[11] Mao Yang, H. C., Ben Y. Zhao, Yafei Dai, and Zheng Zhang. "Deployment of a Large-scale Peer-to-Peer Social Network." First Workshop on Real Large Distributed Systems (WORLDS 2004), 2004.

[12] J. R Douceur, "The Sybil Attack." In Electronic Proceedings of the International Workshop on Peer-to-Peer Systems, 2002.

[13] Sepandar D. Kamvar, Mario T. Schlosser and Hector Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks. " In Proceedings of the Twelfth International World Wide Web Conference, 2003.

[14] D. J. Watts. "Small Worlds, The Dynamics of Networks between Order and Randomness." Princeton University Press, Princeton, NJ,1999.

[15] Yamini Upadrashta, J. V., Winfried Grassmann. "Social Networks in Peer-to-Peer Systems." Proceedings of the 38th Hawaii International Conference on System Sciences, 2005.

[16] Friedman, E. and P. Resnick, "The Social Cost of Cheap Pseudonyms. "Journal of Economics and Management Strategy, 2001.

[17] Milgram E, "The Small World Problem", Psychology Today, 60-67

[18] Newman, M. E. J, "Models of the Small World: A Review", Journal of Statistical Physics, 101, 819-841, 2000.

[19] Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S,"Search and replication in unstructured peer-to-peer networks", Proceedings of the 16th international conference on Supercomputing, New York, USA, 84 − 95, 2002.

[20] Leander Kahney. Cheaters Bow to Peer Pressure, http://www.wired.com/news/technology/0,1282,41838,00.html, 2001.