An Approach to Combating Free-riding in Peer-to-Peer Networks *

Victor Ponce†, Jie Wu, and Xiuqi Li
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

April 7, 2008

†Corresponding author

*This work was supported in part by NSF grants ANI 0073736, EIA 0130806, CCR 0329741, CNS 0422762, CNS 0434533, CNS 0531410, and CNS 0626240. Email: vponce@fau.edu, {jie, xli}@cse.fau.edu

***Abstract*—Due to the dynamic nature of P2P systems, it is impossible to keep an accurate history of the transactions that take place while avoiding security attacks such as whitewashing and collusion, and abuse such as freeriding. This is why it is important to develop a mechanism that both rewards cooperative peers and punishes misbehaving peers. Modeling P2P networks as social structures can allow incentive mechanisms to be developed that prevent the negative behaviors mentioned. In a social structure, peers make and receive payments for services provided to and from each other. In this paper, we extend a social network algorithm to include the transfer of credit between peers to reduce the path length in queries. We also develop a selection strategy that involves different aspects of peer interactions in P2P networks and a credit transfer mechanism that helps to discourage misbehaving peers by taking away credits that they have with good peers and transferring them to more cooperative ones. The simulation results show that our algorithm is effective in reducing the amount of debt between peers, meaning that peers become more cooperative, and shortening the average path length to a satisfied query, while increasing delivery ratio.**

Keywords: Balance, credit, friendship, incentives, peer-to-peer (P2P), proximity, social network, trust.

I. Introduction

Peer-to-peer (P2P) networks are computer networks consisting of ad hoc connections. These ad hoc connections are formed between individual peers and each peer is both a client and a server. P2P networks are primarily used for the efficient, widespread distribution of files known as file-sharing. The ability to establish ad hoc connections between individuals makes P2P networks popular. An increase in their usage has led to more advanced forms of P2P networks used in a wider range of applications. P2P networks can be classified as structured or unstructured depending on how the data is stored within the network. In this work we focus on unstructured P2P networks. No special network structure needs to be maintained in unstructured P2P networks, therefore joining the network is simple. They are also resilient to node join/leave (commonly referred to as turnover). Protocols such as BitTorrent [4], Gnutella [7], and eDonkey [1] are just some examples of these file-sharing systems. To find desired files, queries are flooded or forwarded randomly or intelligently in unstructured P2P networks. This type of network is also vulnerable to attacks by malicious peers since the free nature of the network makes it difficult to enforce security.

A peer in a P2P network that consumes many resources but provides few is referred to as a freeloader, leecher, or freerider. It was found that the algorithms used in the original implementation of BitTorrent were not able to effectively reduce the amount of freeriding in the system unless it had few seeds [8]. As shown in [2], more than 70% of the population in the Gnutella network consume the resources of the network without contributing in return. Another type of malicious behavior is to willfully cheat other peers. Cheating, in this sense, involves disrupting network traffic or knowingly providing corrupt or harmful files. Whitewashing is a term used to describe the action that a peer performs when it leaves the system (discarding its ID) and rejoins the network at a later time (with a new ID). This behavior is used by peers to "wash" away their previous bad actions.

Social networks are one way to alleviate the problems that are inherent in unstructured P2P networks. Social networks are structures made of nodes that are individuals or organizations. They are used to define relationships (links) between individuals. The relationships depend on the characteristics of the individuals and their interactions. The characteristics of a peer determine its behavior, which in turn determines the quality of the relationship. P2P networks modeled as social networks include some sort of incentive mechanism so that peers are more inclined to be cooperative. Some measure of contributions, such as credit, is used to quantify the quality of the relationship between peers. Incentive mechanisms reward good peers and punish those that misbehave by giving or taking away credit.

In this paper, we extend an existing incentive mechanism that is successful in isolating freeriders [13]. We include the transfer of credit between peers to reduce the path length in queries. We also develop a selection strategy that involves different aspects of peer interactions in a P2P network. The credit transfer mechanism also helps to discourage misbehaving peers by taking away credits that they have with good peers and transferring them to more cooperative ones. We model a P2P network as a social structure where each peer behaves as a person in a society, making judgmental decisions about other members in the society.

The rest of this paper is organized as follows: Section 2 gives some background information on some social network models and defines the model we are extending. Section 3 presents our idea in detail. Section 4 describes some security issues and how they are solved. Section 5 describes the simulation environment and the results, and Section 6 concludes the paper and discusses ideas for future work.

II. Related Work

There has been much research in P2P networks using a social structure to improve cooperation by providing good incentives. The idea of modeling a computer network as a society of peers is introduced to solve one of the principle problems of ad hoc networks; due to the lack of authority or structure, peers may behave selfishly. This is why criteria are introduced to score peers. All related works that are mentioned in this paper are similar in that they use history information and some manner of credit between peers. Improving cooperation is also an important issue in other areas of computer networks, such as in power-aware systems found in Mobile Ad Hoc Networks, or MANETs.

In [9], Nandi et al. implemented a transitive trade system where credit is transferred throughout the entire path of a transaction. In their schemes, peers' interactions are described by a relationship where peers have credit and confidence values with those with which they have interacted. A Distributed Hash Table (DHT) is used to find paths to a data source. Their protocol uses the Pastry [11] routing constraint to find nodes with keys that most closely match the requested data key.

In [13], Wang et al. use a social network to model a P2P system. They model a P2P network using a directed graph, where the nodes are peers and the edges are connections

between peers. They define a friendship between two peers which is represented by the directed edges in the graph. Each edge has a credit and a payment weight assigned to it, where the credit from one node to another is the payment from the other node to itself, i.e., $C_{ij} = P_{ji}$ where i and j are two nodes in the graph with an edge connecting them. Each node assigns each direct (1-hop) neighbor a credit and payment, and these neighbors are called friends. This information about the data transferred between peers is then used to describe the strength of the friendship. Then, a balance of friendship is used in a decision function to determine routing paths. This has the drawback of not allowing peers to choose a path based on direct interaction with their neighbors or on the location of the data they seek. This paper lacks an important decision criterion: location. Disregarding the location of the destination peer implies that all paths have the same cost regardless of their lengths, which also prohibits the selection of the best possible server.

In [5], Feldman et al. develop a model in which users decide whether to contribute to a system based on the number of other contributors in the system. If there are too few contributors, then the deciding peer will be less willing to participate because of the increased load on itself. The cost of contributing, to a peer, is the inverse of the total percentage of contributors in the system. This research differs in that different properties of the network are taken into account in routing decisions.

In [15], Zhang et al. define a scheme in which each node is associated with two parameters: money and reputation. Peers exchange money for service and increase their reputations while doing so. There is a central authority that settles disputes between peers when one believes it overpaid or did not receive enough service. The central authority is a set of randomly chosen nodes in the network. Similar to other schemes, they classify peers into three different types: honest, selfish, and malicious.

In [3], Buttyán et al. introduce virtual currency, called nuglets, into a mobile ad hoc network to stimulate cooperation among nodes that are self-interested. They describe two distinct models and a hybrid model. In one model, the node sending the packet pays for the service while in the second model the node receiving the packet is charged. The hybrid model is a combination of the first two models where the packet is paid for partially by the sending node, and the remainder of the fee for forwarding is paid by the receiving node. Simulation results show that the performance of these models is worse than the standard method due to increased packet size and higher likelihood of packet-dropping.

Another paper focusing on improving cooperation in mobile ad hoc networks is [16]. Zhong et al. design a cheat-proof, credit-based system called Sprite to improve cooperation in MANETs. In this mechanism, a central service is used to manage payments to serving nodes.

Similar to the aforementioned papers, we have a credit system that is used to keep track of the interaction between peers. We introduce two new elements: the transfer of credit between peers and the proximity of a peer's neighbors to the data source in a query. Transferring credit will promote proximity routing by allowing paths that were once inaccessible (because of debt) to be taken.

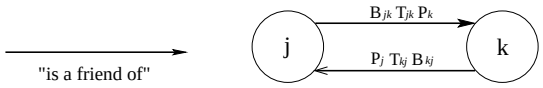


Fig. 1. Representation of balance and trust between peers in a p2p social network. The absolute balance between two nodes is the same in both directions, i.e., $B_{jk} = -B_{kj}$.

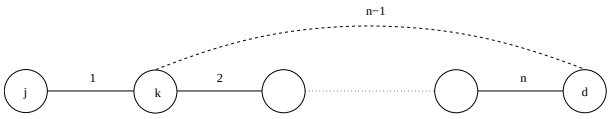


Fig. 2. The proximity of k to d, the destination, from j's perspective is written as P_k and is equal to the number of hops from k to d, if the total number of hops in one path from j to d through k is n then $P_k = n - 1$.

III. Incentive Model

The unstructured P2P network is modeled as a directed graph where each vertex, or node, represents a peer in the network. The edges of the graph represent the relationship between two peers, which may also be referred to as a friendship. There are exactly three arcs from one node to another friend node, whose weights are described below. A peer that initiates a search, or query, is referred to as the source. The peer which stores the data that is being queried is called the destination. 1-hop neighbors of a peer are called friends.

There are several criteria that peers use in the selection of friends to query. They are *balance*, *trust*, and *proximity*, which are shown in Figures 1 and 2.

1) balance - B_{jk}, the difference between the total amount of service provided by a node j to a node k and the total amount of service provided to node j by node k. This amount is cumulative from the first interaction between the two nodes and is used in server selection and decision to serve.
2) trust - T_{jk}, the quantitative measurement of the interaction between the nodes j and k. The longer the relationship, the higher the trust. This means that a long-term friend will have a better chance of serving and, likewise, of being served. Trust is used in server selection and decision to serve.
3) proximity - P_k, the relative distance of a friend node k from the destination node d from current node i's perspective. Proximity is measured in terms of the number of hops from the friend node to the destination and is used only in server selection.

These properties are used by each node in selecting a path to the destination so that the following occur: 1) the network bandwidth is utilized efficiently by routing through paths that are closest to the source, and 2) by using balance and trust between direct neighbors, peers will initially choose the friend that owes it the most, relative to the debt of the other direct neighbors.

We make the following assumptions in our work: First, peers in the network are dynamic, joining and leaving as they please. For ease of programming, when a peer exits the network it is replaced by another peer with the same characteristics, i.e., strategy and minimum and maximum number of friends. Second, the content of the data being stored and transferred in the network is not the issue in this paper. Therefore, it is possible for a node to send incorrect data to a requesting node, whether it is due to maliciousness or corrupt data.

A. Interaction Between Peers

The direct interaction between two peers is captured from two aspects: balance and trust. The balance of the interaction depicts the give-receive relationship between two nodes j and k, where the amount G_{jk} given and the amount R_{jk} received can be measured in actual data amounts transferred in bytes or packets. The balance also reflects which peer contributes more. The trust value between two peers shows the amount of interaction between them up to that point in time. In other words, although two peers may have only known each other a relatively short period of time, they may have a high trust value if they have exchanged large amounts of data within that time. As in real-life situations in which people are more willing to trust their helpful friends, peers in a P2P network should also be more willing to trust peers with which they have had more positive interaction. The balance and trust are defined as follows for current node j and friend node k:

$$B_{jk} = G_{jk} - R_{jk}$$

$$T_{jk} = G_{jk} + R_{jk}$$

where G_{jk} (give) is the total amount of service provided by j to k and R_{jk} (receive) is the total amount of service provided to j by k.

B. Path Discovery

Searching is done by flooding the network with a path discovery message. However, similar to [13], we use an iterative deepening approach in which the combination of depth-first and breadth-first search is used to minimize network bandwidth consumption. In order to govern the extent of network bandwidth that is consumed in the path discovery phase, a time-to-live (TTL), in number of hops, is set by the source node and the request is sent to all of its friends. At each hop, the TTL is reduced by one. If the friends do not have the data, they forward the path discovery message to their friends until the TTL parameter is zero or the data is found. Each time the data is not found, which is detected by a timeout, the source node increases the TTL, if it is less than a specified limit, by one and the query is sent again to all of its friends. Again, the message is forwarded until the TTL is reached or the data is found. All of the peers which are the next hop in a path to the data source are put in a separate list L of friends which is used in the server selection phase.

C. Server Selection

The path discovery phase may result in some or many paths being found, or none at all. After possibly multiple paths have been discovered, the source decides which path to take based on the relationship it has with the friends through which paths exist. Source j computes a value Q_{jk} for each friend k in L and chooses the peer with the highest Q_{jk} value to request service:

$$Q_{jk} = \left(\frac{B_{jk}}{\sum_{i=0}^{f} B_{ji}} \right) w_1 + \left(\frac{T_{jk}}{\sum_{i=0}^{f} T_{ji}} \right) w_2 + \left(\frac{\frac{1}{P_{jk}}}{\sum_{i=0}^{f} \frac{1}{P_{ji}}} \right) w_3$$

where $w_1 + w_2 + w_3 = 1$ and f is the total number of friends with paths. The weight assigned to each term may vary depending on what is more important in a particular application. If, for example, you want to heavily punish freeriders, then w_1 should be higher. If you want to give more importance to the amount of interaction between friends, then w_2 should be higher. Lastly, if you want to give more importance to the relative distance to the data you are looking for then w_3 should be higher. We experiment with different values for the weights in our simulations to see how system performance differs.

The idea of choosing the next hop based on its proximity to the destination is similar to a mechanism used in [12], [14] where nodes compute heuristic values for their neighbors based on iterative forwarding to choose the least cost path, in terms of energy consumption, to the destination. The cost in P2P networks refers to the delay in finding the query.

D. Decision Function

When a request for service is received by a peer, it decides whether to provide service in the form of supplying the data or forwarding the request according to the output of a decision function. The balance and trust of the friendship are used to calculate the value of the decision function.

The output of the decision function, F_{kj}, is a probability deciding whether node k should supply service to node j, the requesting node. The properties of this decision function are:

- the probability lies within [0,1].
- the probability is a decreasing function of the balance/trust ratio (Figure 3), indicating repayment of services by others because if the ratio is low (balance$\ll$trust) the probability of supplying service will be high.

The decision function is as follows:

$$F_{kj} = \frac{1}{2}\left(1 - sin\frac{\pi}{2}\frac{B_{kj}}{T_{kj}}\right)$$

This decision function was proved to be effective in reducing collusion in [13]. We propose a transfer-of-credit mechanism to achieve improved network balance and delivery ratios. Balancing the network means that we want the friendship balance of all peers to be close to the mean friendship balance of the network so that there are few peers who owe significantly more to others. Simulation results show that the network is more balanced using the credit transfer mechanism.

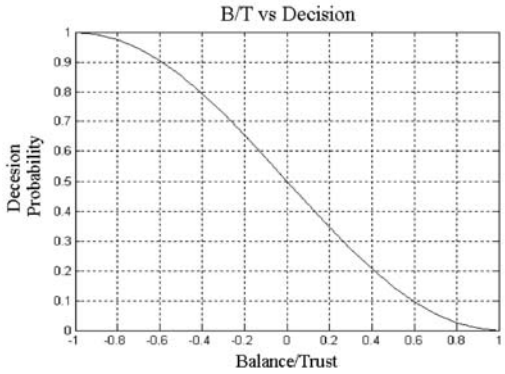


Fig. 3. In the figure, as the balance/trust ratio between two peers k and j increases ($\frac{B_{kj}}{T_{kj}} \rightarrow 1$) the probability of peer k serving peer j decreases ($D_{kj} \rightarrow 0$), and vice-versa.

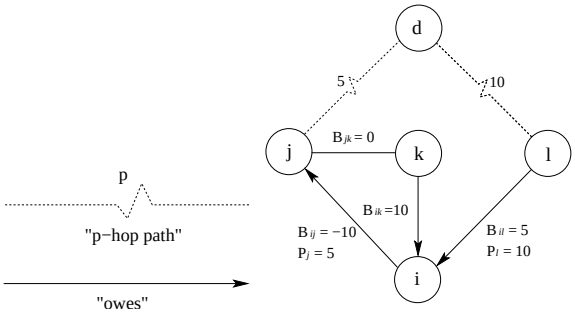


Fig. 4. In the figure, the dotted lines represent the paths from nodes j and l to the destination d and the weight is the hop count, so the proximity values of j and l from i's perspective are P_j and P_l, respectively. In this example, $G_{ij} = 0$ and $R_{ij} = 10$, so $B_{ij} = -10$ and $G_{il} = 10$ and $R_{il} = 5$, so $B_{il} = 5$ to illustrate the effect that the credit transfer has on the routing decision. If credit transfer is not used, $Q_{ij} = 0.47$. But, if credit transfer is used $Q_{ij} = 0.79$, so j will be chosen by i to provide this service.

E. Credit Transfer

In this work, we propose to allow direct transfer of credit from one peer to another. This transfer of credit aids in balancing relationships between peers. It also helps to make transactions possible between two peers that have direct information about each other but one is indebted to the other and helps peers choose the best path with the criteria mentioned earlier. Credit can only be transferred from a 1-hop peer to another 1-hop peer. The three peers involved in the credit transfer are friends, so each knows about the other. The following is a scenario in which credit transfer will allow a peer to serve another, which would not occur if credit is not transferred.

The three peers in the credit transfer are already friends. A simple example of how the credit transfer works is shown in Figure 4. Peer i owes peer j an amount 10, so $B_{ij} = -10$. Peer k owes peer i an amount 10, so $B_{ik} = -B_{ki} = 10$, and l owes i an amount 5, so $B_{il} = -B_{li} = 5$. Peer k and peer j are even, so $B_{jk} = B_{kj} = 0$. Through the path discovery stage peer i finds that peer j can provide it with the best path. Without credit transfer, the request values of j and l are $Q_{ij} = 0.47$ and $Q_{il} = 0.53$. With these values i would normally choose to query l, even though the path through j is closer. So peer i can use the amount owed to it by k to utilize peer j's services. Peer i zeros the balance that k owes it by adding the difference between G_{ik} and R_{ik} to R_{ik} (to zero the balance) and also adding that amount to G_{ij} (to increase the balance), so that now $Q_{ij} = 0.79$ which will result in choosing peer j. All three peers benefit from the transfer: i is able to use the best path, j is paid what it is owed and an additional amount if B_{ki} was higher than B_{ij}, and k does not owe i anymore and it has also gained some trust with i. This method of transferring credit retains the history information between the peers so that in future routing decisions the zero balance does not have a great impact. Using this method, all three peers have to acknowledge the credit transfer.

The credit transfer mechanism is used by the requesting peer when the server denies its service based on the decision function. The requesting peer looks through its list of friends and finds ones that are indebted to it. It then selects a common friend between itself and the serving peer. From this peer, the debt is removed and transferred to the serving peer. Then the serving peer acknowledges this transfer and increases the requesters credit by the same amount. In this way, the requester increases the balance from it to the server and is provided the service.

F. Establishing and Breaking Connections

1) New Connections: After a successful transaction, the peer that initiated the query (requester) may ask the peer that provided the data (server) to become its friend if its friend list is not full. There are two factors that are considered in this case: the number of current friends of the server and the distance between the two peers. The following function, which is similar to the one in [13], describes the decision C_{new} to form a new connection:

$$C_{new} = min\left[1 - \frac{N_{cur}}{N_{max}} + \frac{r \times (H_{cur} - 1)}{H_{max}}, 1\right]$$

where N_{cur} is the current number of friends, N_{max} is the total number of friends allowed, H_{cur} is the number of hops that were taken to get to that peer, H_{max} is the maximum number of hops allowed in a search, and r is a random number, $r \in$ [0,1]. This probability function is used after a successful transaction, meaning that the data was transferred from the server back to the requester, hop by hop. If the distance in hops is further, the probability to make this new friendship is higher, and vice versa. If the server has a smaller number of friends then the probability is also higher, and vice versa. The random value r is included because sometimes the server has few empty slots available in its friend list, so the probability of choosing the requester is mostly influenced by the distance between them.

There is also a mechanism for creating a new connection between two peers that are strangers but have friends in common. In our mechanism, the peer that is asked for friendship asks its friends if they know the stranger peer and weighs their responses based on their relationships. The peer being asked also weighs its decision using the ratio of its current number of friends to the maximum number of friends. The function

to determine whether j will accept the new peer i as a friend is denoted as S_{ji}:

$$S_{ji} = 1 - \frac{N_{cur}}{N_{max}} \times \frac{1}{2} \sum_{k=0}^{f} \left(\frac{B_{jk}}{T_{jk}} \times \frac{B_{ki}}{T_{ki}} \right)$$

where f is the total number of friends of j. This function is used when a new peer enters the network and has no friends. It randomly chooses a peer in its vicinity to request friendship. Peer j, the one being asked for friendship, calculates its S_{ji} value about the requesting stranger peer i. Peer j takes into consideration its current number of friends as well as their opinions (if any) about i. The product of the two ratios is halved so that new connections are less likely to be accepted from stranger peers with common friends. It may be more advantageous to form friendships with peers that your friends do not already know so that the data shared by the circle of friends is less likely to be redundant.

2) Old Connections: There are times when there is little or no interaction between two peers that are friends. Since peers have a limit on the maximum number of friends they can have at any time, there must be a way to break ties with current friends. A simple solution to this is to decay the give and receive amounts by a predetermined value λ. The connection between two peers will be severed if the balance, B_{jk}, between nodes j and k exceeds a threshold. When the balance is higher than the threshold, it means that the amount given, G_{jk}, is much greater than the amount received, R_{jk}. The give and receive amounts are updated as follows periodically from peer j:

$$R_{jk} = R_{jk} - \lambda$$
$$G_{kj} = G_{kj} - \lambda$$

for every friend k in j's friend list that does not serve. Each friend k also updates its information about peer j.

The balance and trust amounts can be directly related to the amount of data that is exchanged. Therefore, the give and receive amounts are incremented by τ. The give and receive amounts are updated as follows:

$$G_{jk} = G_{jk} + \tau$$
$$R_{kj} = R_{kj} + \tau$$

for every friend k in js friend list that served. τ represents the amount of data or service provided by each friend. Again, both friends update their information about each other so as to keep an accurate record of their transactions.

IV. Security Model

We discussed how the credit transfer among friends works in the previous section. In this section, we discuss how the information about the peers involved in the credit transfer is kept secure. Currently, the exchange of credit among friends in the P2P social network is strictly blind. Assume that there are three peers involved in a credit transfer: peers i, j, and k. Peer i is the one that needs the credit so that it may obtain the service from peer k, and peer j is the one which will provide the credit for the transfer. Transferring debt between a triangle of peers requires that each peer trust the others. The problem here is that if peers j and k are not already friends, then k will have a hard time verifying peer j.

The assumption here is that when the network is first formed, the majority of the peers are honest peers. We do not discuss key distribution, therefore man-in-the-middle attacks are possible.

A. Case 1 - All Informed

In this case, all three peers know each other. That is, peer i has prior balance, trust, and proximity information for previous transactions with or through j and k each. Likewise, peers j and k have the same prior information about each other. In this scenario, all three peers have prior information about the others and therefore have proof of their existence. The only challenge now is to verify the amount that is being transferred from i to k.

One method of doing this is to have a digital signature triangle. Since peer i is the one with complete information (i has absolute knowledge about the amount being transferred since he is the one transferring), he is responsible for initializing the credit transfer authentication process.

The message exchange sequence is shown in Figure 5. Peer i generates message m which contains the IDs of each peer and the amount being transferred:

$$m = [ID_i, ID_j, ID_k, TA]$$

where ID_i, ID_j, and ID_k are the identifiers of peers i, j, and k, respectively and TA is the amount being transferred. The first ID is of the initiator, the second of the credit provider, and the third of the credit recipient. No additional information is needed about the recipient or giver of credit. Peer i then encrypts the message using the RSA encryption algorithm and signs it using the SHA-512 hash. The hash function chosen, however, is implementation-dependent. Therefore, another hash function may be used to reduce complexity. Peer i sends m first to peer k who then asks peer j to verify the amount. After receiving peer j's signature of confirmation, k sends a signed copy back to both i and j, so now all three peers have signed and agreed to the transfer. Then, as a final check, peer i confirms peer j's signature and the credit transfer can take place. If peer k's verification of the amount of credit being transferred from i to itself fails, then it sends a message back to i notifying him of the failure and the reason (if provided by j) and denial of service for that instance.

B. Case 2 - Partially Informed

In order for cooperation to exist among peers in a P2P network, there must be trust. As is, there is trust based on amounts transferred between peers. This works well for recording the behavior of your friends with yourself, but what if credit needs to be transferred between non-friend peers. This case is not as trivial as the first. Not all of the peers involved in this scenario know each other - they do not all have friendships with one another. In this case, peer i is a friend of both peers

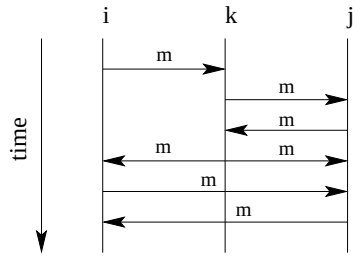


Fig. 5. Credit transfer message passing sequence

j and k but peer k is not a friend of peer j (Figure 6). Again, i wants service from k, but as it is peer k cannot provide i any service.

Here, a distributed authentication mechanism is used, similar to [10], to authenticate unknown peers. First, peer i sends a digitally signed message to k with j's public key so that k can authenticate peer j. Then, peer k sends a challenge message to k that contains the amount that will be transferred. Peer k then asks his friends (l, m, n) to send a message to j to verify its identity as well. The messages sent by k's friends are nonces encrypted with js public key. j returns the nonces in a signed response message. The challenge response message pair is j's proof of possession of the public key. All of k's friends send their verification messages to k and if all of the messages agree then k trusts peer j. If one of the messages does not agree, then either j or some of peer k's friends are malicious. Peer k then checks if peer j is malicious by sending it all of the verification messages and asking him to prove that it signed the messages. If j is able to prove that all of the messages were signed by k's friends then one of peer k's friends is malicious. This leads k to announce that a Byzantine fault has occurred. Each group member will send the Byzantine agreement message to others. At end of this phase, the honest peers will be able to identify the malicious peer. If the authentication process went well, peer k continues with the credit transfer. Peer k also adds j to a list *foaf* (friend-of-a-friend). This list can be used in the future by k if, for example, a query can not be satisfied by one of its own friends.

V. Simulation Setup and Experimental Results

In this section we present the simulation settings and demonstrate that our algorithm is successful in accomplishing our goals of reducing path length and debt between peers. First we describe the simulation framework. Then we show the initial system settings and the simulation results.

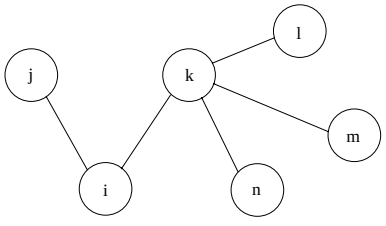


Fig. 6. Unknown third party credit transfer scenario. Solid lines indicate friendships.

TABLE I
Table of Parameters

Parameter	Value	Parameter	Value
Population Size	100	Run time (rounds)	1000
Max No. Friends	10	Ratio Cooperative Peers	1/3
Ratio Defective Peers	1/3	Ratio Decisive Peers	1/3
Learning Probability	0.05	Turnover Rate	0.01
Initial G_{jk}	30	Initial R_{jk}	30
λ (payment decrease)	2	τ (payment increase)	40
Max Hops	5	Cut-off Threshold	10

A. Simulation Framework

As in [6], our simulation consists of rounds. Each round is a logical time unit in which every peer plays two roles. There are two games: client and server. Every peer decides what role they want to play in the beginning of each round. After the choice is made, some will play as clients and some will play as servers. The peers which are clients may choose whether to query, and the peers which are servers may choose whether to serve. The credit scores are updated accordingly for serving and non-serving servers each time a query is made. The decision by a server to choose is based on its strategy. There are three types of strategies:

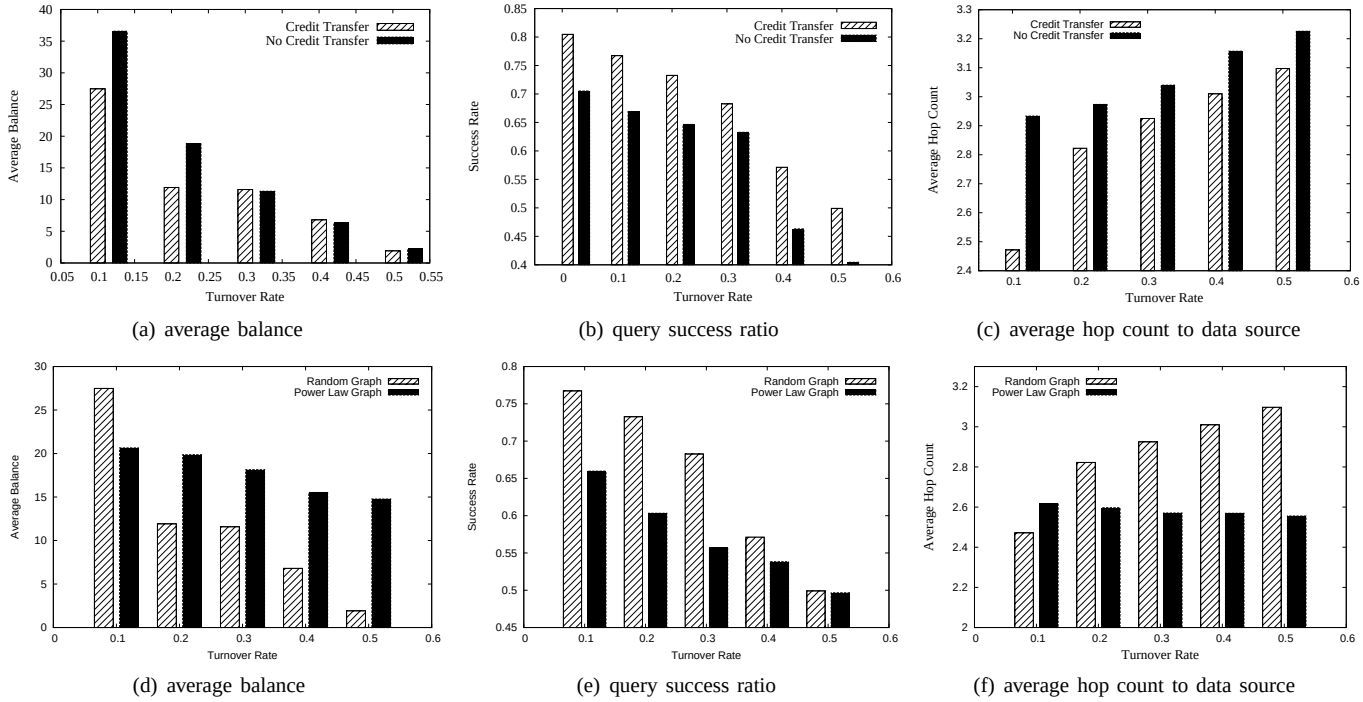
- Cooperative - A peer which always chooses to serve.
- Defective - A peer which always chooses not to serve.
- Decisive - A peer which uses the decision function described in this paper.

At the end of each round each peer will choose to take one of the following actions:

1) Learn - Each peer rates his own strategy. Every time a peer makes a query it updates its strategy's score. When a query is successful it increases the score and when the query fails it decreases the score. At the end of each round, if it chooses to learn, every peer ranks its strategy among its friends and chooses a new strategy with a probability proportional to the difference between its strategy's score and the average of its friends. This probability will be low if the peer's strategy is ranked relatively high, so it will more likely keep its current strategy.
2) Exit - If a peer chooses to exit the network, a new peer with the same ID and strategy will enter the network to replace it. The rest of the parameters of the peer, such as friends, strangers, and all other values are re-initialized.
3) Remain - A peer may choose to remain in the network and continue with its current strategy. Nothing happens in this case; all parameters for this peer remain the same.
4) Mutate - A peer may randomly choose to mutate to another strategy. All of the peer's parameters remain the same with the exception of its strategy.

B. Settings

Table I shows the default simulation settings. The network consists of 100 peers. The simulation consists of 1000 rounds, each of which is a logical time unit. The maximum number of friends that a peer can have is 10. Initially, the population in the network is equally divided into three categories of peers: cooperative, defective, decisive. These categories define the



(a) average balance (b) query success ratio (c) average hop count to data source

(d) average balance (e) query success ratio (f) average hop count to data source

Fig. 7. Simulation results

strategy of the peer, which changes throughout the simulation, as described in [13]. The probability to learn and change strategy is set to 0.05. The turnover rate, or rate at which peers exit the system and are replaced by new peers, is initially set to 0.01. The $Give$ and $Receive$ amounts are initially set to 30. These are always the initial values for new friendships. The payment for serving another peers (λ) is 2. The decay amount (τ) is set to 10. The maximum number of hops that a query can travel is set to 5. The cut-off threshold for friends who have accumulated too much debt is 10.

C. Results

The weights for the $balance$, $trust$, and $proximity$ were varied to find the optimal values. Several different values gave good results, but the best weights are: $w_1 = \frac{1}{2}$, $w_2 = \frac{1}{3}$, and $w_3 = \frac{1}{6}$. The following results were obtained using these values.

Figure 7(a) shows the average balance in the network for different turnover rates with all other values set to default. It shows that our algorithm does have an average balance closer to 0. This means that throughout the experiment the average debt of the peers in the network is lower. As the turnover rate increases the average balance for our algorithm is about the same as the algorithm without credit transfer. The reason for this is that when more peers leave and new ones come into the network the $Give$ and $Receive$ amounts are initialized and the balance is 0, which leaves little room for balancing the debt between friends using credit transfer.

Figure 7(b) shows the average query satisfaction rate for the two different algorithms. As the figure shows, the satisfaction rate using our algorithm is higher at all turnover rates. Even at a 50% turnover rate, which is realistic in a real P2P system, our algorithm with credit transfer and friend proximity consideration has a query satisfaction rate of close to 50%.

Figure 7(c) shows the average number of hops to the data source node in the query. Clearly, the number of hops to the data source is reduced by using credit transfer and the hop count metric. Our algorithm works exceptionally well at low turnover rates because it uses history information.

Power-law Distribution

In order to show a more realistic simulation, we ran the same algorithm on a network with a power-law distribution. The peer population stabilized to 100 at round 150. Then we ran the simulation for another 1000 rounds, as in the randomly distributed simulation. We compare the results for the algorithm with credit transfer with two types of distributions: random and power-law.

Figure 7(d) contains the results for the average balance in the network with different population sizes. The graph shows that the average balance between friends did not decrease much as the turnover rate increased. This may be due to the fact that because of the power-law distribution, more nodes are connected indirectly through the highly-connected peers. At a turnover rate of 0.1%, the average balance is higher in the random network, whereas at higher turnover rates the average balance is lower. The reason that the average balance for the power-law network did not decrease much is because the balance between the highly-connected peers and those that joined the network at later times kept increasing until the connection was severed by the highly-connected peer. As a result, the average balance does decrease slightly with higher turnover rates because the duration of a friendship is also decreased. A lower duration of friendship because of turnover means that the connection will not have to be severed due to lack of cooperation.

Figure 7(e) shows the average query success ratio of the algorithm with credit transfer. The success ratio is always

higher at any turnover rate with a random distribution of peer friendships. This is closely related to the reasons given for the average balance. Highly-connected peers in the power-law network have many friendships that accumulate debt. As a result, they are less willing to help those friends until the connection is severed. The success ratio is further inhibited by the high turnover rate, not giving friendships enough time to strengthen and become useful.

Figure 7(f) shows how the average number of hops to a data item differs between random and power-law distributions. The average hop count in the power-law graph decreases slightly as the turnover rate increases. This is due to the indirect connectivity of the network resulting from the power-law distribution. There are some highly-connected nodes that serve their friends to decrease hop count.

Although the query success ratio is lower for the power-law network, the existence of highly-connected peers reduces the average hop count and the hop count is maintained at a similar level with different turnover rates. The average balance is also affected by the different distribution. The type of peers that are the friends of the highly-connected peers greatly affects the balance. Defective peers will cause the balance to increase while decisive or cooperative peers will cause it to decrease. In the long run, the balance remains high because there is much traffic through a small portion of the network.

VI. Conclusion and Future Work

The problem of encouraging cooperation can be solved using a social network approach. We improved upon an incentive mechanism to include different metrics, such as distance in hops between peers and debt. Our algorithm accomplished two goals: 1) The overall balance of the network was reduced so that there were fewer peers who had higher debts than the average of the entire network, and 2) The average number of hops to the desired file was reduced. This is important because a lower debt in the network yields a higher level of cooperation.

The first goal can help prevent freeriders from taking advantage of those peers who cooperate. Peers that do not cooperate are eventually isolated from those that do cooperate. The second goal means that, by allowing the selection of servers to be two-sided, the data can be found faster. By two-sided we mean that when a peer requests service from a friend, both peers must agree to cooperate. The use of path length information as well as history about peer relationships increases the success ratio of queries and reduces the query path length.

We also specified a method of securing credit transfer. Using our message passing sequence, the information about a credit transfer can be authenticated and verified. We also acknowledge the case where one of the peers in the credit transfer is a stranger, which means it is more difficult to establish trust.

In the future, we plan to expand our algorithm to include security measures against man-in-the-middle (MITM) attacks. We will discuss key distribution and how groups of peers can be used in the authentication process.

References

[1] http://www.emule-project.net/home/perl/general.cgi?l=1.

[2] E. Adar and B. A. Huberman. Freeriding on gnutella. *First Monday*, 5(10), October 2000.

[3] L. Buttyán and J. P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical report, Swiss Federal Institute of Technology - Lausanne, January 2001.

[4] B. Cohen. http://www.bittorrent.org/ bittorrentecon.pdf.

[5] M. Feldman, M. Ahamad, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. In *Proceedings of the SIGCOMM'04 Workshop*, August 2004.

[6] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, May 2004.

[7] Gnutella. http://www.gnutella.com.

[8] S. Jun and M. Ahamad. Incentives in bittorrent induce freeriding. In *Proceedings of ACM SIGCOMM*, August 2005.

[9] A. Nandi, T. Ngan, P. Druschel, and D. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. November 2005.

[10] V. Pathak and L. Iftode. Byzantine fault tolerant public key authentication in peer-to-peer systems. *Computer Networks*, 50:580–597, 2006.

[11] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM Conference on Distributed Systems Platforms (Middleware 2001)*, November 2001.

[12] I. Stojmenovic and X. Lin. Power aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, November 2001.

[13] W. Wang, R. Yuan, and L. Zhao. Improving cooperation in peer-to-peer systems using social networks. In *Proceedings of IEEE IPDPS 2006*, 2006.

[14] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical report, UCLA Computer Science Department UCLA/CSD-TR-01-0023, May 2001.

[15] Z. Zhang, S. Chen, and M. Yoon. March: A distributed incentive scheme for peer-to-peer networks. In *Proceedings of IEEE INFOCOM'07*, May 2007.

[16] S. Zhong, Y. Yang, and J. Chen. Sprite: A simple, cheat-proof, credit-based system for mobile ad hoc networks. Technical report, Department of Computer Science, Yale University, July 2002.