Exchange-based Incentive Mechanisms for Peer-to-Peer File Sharing *

Kostas G. Anagnostakis and Michael B. Greenwald
CIS Department, University of Pennsylvania
200 S. 33rd Street, Philadelphia PA 19104, USA
{*anagnost,mbgreen*}*@dsl.cis.upenn.edu*

Abstract

Performance of peer-to-peer resource sharing networks depends upon the level of cooperation of the participants. To date, cash-based systems have seemed too complex, while lighter-weight credit mechanisms have not provided strong incentives for cooperation.

We propose exchange-based mechanisms that provide incentives for cooperation in peer-to-peer file sharing networks. Peers give higher service priority to requests from peers that can provide a simultaneous and symmetric service in return. We generalize this approach to n-way exchanges among rings of peers and present a search algorithm for locating such rings. We have used simulation to analyze the effect of exchanges on performance. Our results show that exchange-based mechanisms can provide strong incentives for sharing, offering significant improvements in service times for sharing users compared to free-riders, without the problems and complexity of cash- or credit-based systems.

1. Introduction

Peer-to-peer systems provide a powerful infrastructure for large-scale distributed applications, mainly because of the wide-spread cooperative resource sharing among participants. Cooperation and the existence of a critical mass of participants with sufficient resources are key elements for enabling a variety of novel applications such as file sharing, large-scale content distribution, and distributed data processing. Performance in such systems depends on the level of cooperation by the system's participants. While most existing peer-to-peer architectures have assumed that participants are generally cooperative, there is growing evidence from widely deployed systems suggesting the opposite. For instance, one study of the Gnutella file sharing system shows that almost 70% of the peers only consume resources but do not provide any files [7]. The result of this non-cooperation can vary between tolerable service degradation and complete system collapse depending on design goals and performance requirements.

Such problems have recently motivated work on incentive mechanisms for peer-to-peer systems that stimulate cooperation between self-interested participants. Systems such as KaZaA [3] attempted some rather naive methods where each peer announces its "participation level", computed locally as a function of uptime, download and upload volume, and give priority to remote peers that claim high participation levels. However, this is easily subverted because peers can *claim* anything. Simple software modifications to do this are easily accessible [2] and widely used [1]. Other proposals to date require the use of a credit system which can be either centralized or decentralized. Centralized mechanisms [4, 17] (*e.g.,* using micropayments issued by a trusted server or a centralized transaction clearing center) inherit the typical disadvantages of centralized designs: they introduce a single point of failure, they may put a significant burden on a single peer, and, perhaps most importantly, it may be hard to design the right incentives for one or more peers to take up such a demanding and sensitive role. Recent proposals for decentralized credit mechanisms [24, 18] are based on distributed hash tables (DHTs) [25, 23, 21] and therefore inherit another set of problems. For instance, heterogenous node capabilities make efficient allocation decisions hard, transient peer participation may significantly stress reconfiguration performance, and there are known classes of attacks that are likely to be directed against the credit system given its importance [13].

As an alternative, we examine a lighter weight approach that avoids most of the complexities of credit mechanisms. Rather than building a system based on principles of monetary or credit economies, we structure the system as a more primitive *exchange* or *barter* economy. Users directly trade resources between themselves, so little or no long-term bookkeeping is required. Requests from peers that can provide a simultaneous, symmetric, service in return (*exchange*

* This work was partially supported by the DoD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795.

transfers) are given higher priority. The service need not be directly to the provider (a *two-way exchange*), but more generally priority is given to peers who participate in *N-way exchanges* to which the provider currently belongs. N-way exchanges are implemented as *rings* of N peers, where each peer is served by its predecessor and serves its successor in the ring. Non-exchange transfers are only served if no other exchange is possible and peers have spare capacity. The preference given to exchange transfers provides a strong incentive for participants to cooperate.

2. Exchange mechanisms

In this paper we consider a file sharing system where each peer has fixed upload and download capacity. The upload capacity is more likely to be the resource bottleneck than the download capacity. To manage the upload link, we respond to all requests in relatively large, equal, fixed-size, blocks. We assume that the system supports partial transfers and that peers can download different parts of the same object concurrently from multiple sources. To focus on the main point of this paper, we ignore the details of object lookup. We note that our approach can work with several known search mechanisms including broadcast in Gnutella-like networks or a DHT query in systems like Chord. When a peer is interested in an object it can use one of these methods to locate up to a certain fraction of peers that currently have the object.

Each peer has an *incoming request queue* (IRQ) where remote peers register their interest for a local file. A transfer to satisfy a request is initiated if two conditions are met. First, the local peer must have upload capacity (at least one open fixed-size slot on the upload link)[1]. Second, either the transfer is an *exchange transfer*, or else no other request in the IRQ is both an exchange transfer *and* satisfies the first condition.

All exchanges are performed one fixed-size block at a time. Transfers are terminated if one of the two communicating peers disconnects, if the source deletes the object, or if the transfer is completed. The transfer terminates when the *first* peer completes its own download, because object sizes may differ and the system allows partial and concurrent transfers.

Non-exchange transfers are only served if no exchange is possible and the peer has a free upload slot (although these slots are preemptively reclaimed by exchanges, as they become possible). Peers who share more are more likely to be able to participate in an exchange, directly rewarding them with faster transfers. Thus, the power of the proposed approach is derived from the priority given by the system to exchange over non-exchange transfers.

1 Inadequate download capacity terminates the transfer when the remote node cannot receive its incoming request, it terminates its outgoing upload, and issues the request again when a download is feasible.

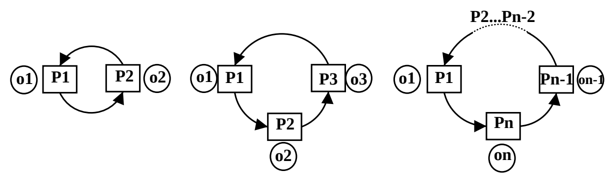

Figure 1. Two-way, 3-way and n-way exchanges

2.1. Exchange transfers

Peers must give priority to exchange transfers. It is therefore imperative that feasible exchanges be identified.

Two-way exchanges are easily detected. Each peer A regularly examines its incoming request queue and determines if, for any pending request, the remote peer B has some object that A is interested in. Although two-way exchanges are simple, unfortunately requests frequently do not resolve into convenient pairs.

Fortunately, it is easy to compute feasible N-way exchanges. Let G be the directed graph whose vertices are nodes in the peer-to-peer system, and whose labeled edges represent requests. An edge from node p_1 to p_2 with label o_e represents a request from p_1 to p_2 for object o_e. It is clear that any cycle of length n in G represents a feasible n-way exchange.

How can we compute cycles in G, a potentially enormous graph? First, we have empirically determined that n-way exchanges, where $n > 5$, do not substantially improve the likelihood of successful exchanges over exchanges where $n = 5$ (see Section 3). Therefore, it is sufficient to search for cycles in chains of up to 5 predecessors. Second, we note that a request from A in the incoming request queue of B represents an edge in G from A to B, and therefore peers already have information about a partial local subgraph of G.

Each peer maintains a *request tree* as follows. A peer with no incoming requests has an empty Request Tree. For peers with non-empty incoming request queues, let each request in the IRQ include the contents of *its* request tree (pruned to a depth of 4). A's Request Tree consists of an implicit root, A, as the parent of the set of Request Trees accompanying each entry in the IRQ. Then, A can initiate an n-way exchange if any peer in the Request Tree owns any object currently desired by A. If a suitable peer is found, and that peer appears at depth n in the tree (the depth includes A as the root), then we can construct a ring of n peers, P_i for $0 \leq i < n$, each carrying object o_i and requesting object $o_{(i+1) mod n}$. Each peer provides an object to their predecessor and gets an object from their successor.

A inspects the Request Tree before transmitting any request and after receiving each request. Prior to transmission of a request for object o_e, A inspects the entire Request Tree to see if any peer provides o_e. On receipt of each request, r, A need only inspect the incoming Request Tree associated with r — but it checks the peers in the incoming tree for *any* object that A still wants.

Note that at the time A decides to request object o_e it "discovers" a (possibly incomplete) set of peers who provide object o_e, but it only issues requests to a subset of those peers. Later, it can use the original provider list to compute a cycle containing a peer, p_i, even if it did not originally transmit a request to p_i. At the initial request time, A had no preference for p_i because A had no way of knowing that p_i was a potential participant in an n-way exchange.

In practice, A must circulate a token through the proposed ring to determine whether everyone is still willing to serve. The ring can be invalid for several reasons. First, in the time between the original requests and the ring initiation attempt, some peers may have gone offline, or crashed. Second, other peers may have already constructed rings of their own, including some of A's intended participants (it is possible that several peers along the intended cycle will attempt to create the same ring roughly simultaneously).

The token circulation also negotiates transfer rate. Each node computes its available upload capacity (managing the allocation between exchanges), and ensures that the proposed transfer rate does not exceed it. The exchange is performed at the minimum of all proposed transfer rates, and each peer can redistribute excess capacity to the next exchange.

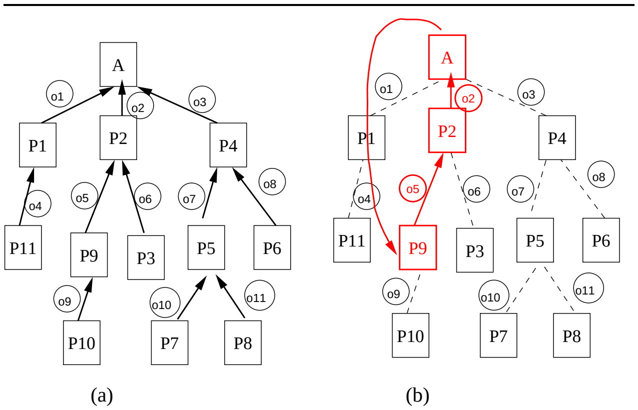

Figure 2. A can be served by some object on P9. The entire request tree is shown in (a). The cycle for the 3-way exchange that A tries to initiate is shown in (b). P3 may have simultaneously discovered a cycle through a target of P2 other than A. Both rings need P2, so only one will be initiated successfully.

An interesting question is how to choose from different feasible exchanges that can satisfy a given request. In principle, a preference for larger rings should improve overall performance, as more peers are served. On the other hand, peers prefer smaller rings as the search cost is lower, and the expected exchange volume is also higher for smaller rings, as the probability of a peer either disconnecting or completing is higher for larger rings. Assuming peers care less about global performance and more about their own benefit, there is no clear incentive to put additional effort into looking for larger rings when even a two-way exchange has been located.

2.2. Preventing cheating

Malicious peers may attempt to cheat the system into giving equal priority to non-exchange transfers. For example, a peer could claim that there is an an exchangeable object available and serve junk in exchange for real data.

Several mechanisms can be used to address this problem with varying degrees of effectiveness. Local blacklists can be used to refuse service to cheating peers, but in a large and dynamic system this is likely to be ineffective because cheaters may perform well enough even if they can cheat each peer only once. Cooperative blacklisting is an improvement, but it requires additional mechanisms which may themselves be subject to attacks. In both cases, the problem persists if it is easy for a peer to assume a new identity that is not blacklisted, as is the case in most file-sharing systems today [14].

It is possible to limit the damage done by cheating by exchanging blocks synchronously and validating each received block before transferring the next one. This requires a trustworthy source of information of valid checksums for the blocks being transferred. The maximum benefit for a cheater in this case would be equal to the block size. If the block size is $b_{exchange}$ bytes and the round-trip time between the two peers is t_{rtt} seconds, this limits the maximum exchange rate to $b_{exchange}/t_{rtt}$ bytes/second. As this may be less than the slot capacity, peers may want to use a window protocol and increase the window size to fill up the slot capacity-delay product, at the expense of increasing risk. A reasonable approach would be to start the exchange with a small window and increase after a number of rounds. A cheater would need to have at least a few real blocks in order to increase the window. It is very likely that even this level of cooperation would have a positive effect on the system as a whole.

Another problem is that a peer could act as a middleman between two peers that could perform an exchange directly with each other, and obtain an object without doing any useful work for the system. Specifically, let us assume that peer A has object x and wants object y, and peer B has object y and wants object x. The cheating peer C, interested in object x claims that he has object y and wants object x when talking to A, and that he has object x and wants object y when talking to B. Peer C would start getting blocks of y from B and exchanging them for blocks of x with A which in turn are passed to B for more blocks of y. In this scenario, peer C does not contribute any useful work to the system, and can still get high-priority service. If this is allowed to happen, then the exchange-based incentives break down.

Tighter control is needed to address this problem, involving the use of a trusted peer as a mediator. Both directions of the transfer can be encrypted, each with a secret key only known to the sending peer and the mediator. In the control header of each transfer block, the sending peer also includes a *peer-of-origin* identifier. The control header is also encrypted, so that a middleman cannot modify it. When the transfer is completed, the trusted peer mediates the exchange of the secret keys, after ensuring that neither side of the exchange has cheated. The mediator can do this by verifying the validity of a small number of randomly chosen blocks from each side of the transfer. The keys are sent to the peers indicated in the control header of the test blocks. In this way, a middleman would not be able to decrypt the blocks he peddled between the two peers in the scenario discussed above, and his participation in the transfer would offer him no benefit.

One remaining issue with this approach is that the middleman can initially obtain two blocks, one for each object peers A and B are interested in, and carry out small, one block transfers with each peer, and then presenting the newly acquired block for an exchange with the other peer. Since the peer can start this process with *real* data that is not encrypted, the protection offered by the mediator is not sufficient. We do not have a similar solution for this problem but we can argue that this way of increasing performance without doing any useful work is unlikely to be possible at a large enough scale to be practical for cheaters as a general strategy. First, the cheating peer needs to wait in low-priority queues to get the "bait" blocks anyway, for both files, adding some latency to the process. Second, the number of potential "victim" peers decreases with the number of blocks the cheater has available. Third, since the cheater needs to have two blocks, one for each peer, he is also constrained by the number of peer-pairs interested in those blocks. Fourth, the cheater wastes his own resources because he uses part of his upload capacity for an object that is totally useless to him. (This is a waste unless he is interested in both objects. Otherwise, he may be better off using this capacity for real exchanges.) Fifth, the peers he is targeting are likely to be talking to each other already so they may be uninterested in what he has to offer, and they may have already committed all of their upload capacity to each other. Finally, additional constraints could be designed into

peer	upload	has	wants
A	10	-	x
B	5	x	y
C	10	y	x
D	10	y	x

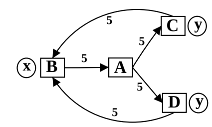

Figure 3. Example middleman scenario resulting in non-ring exchange

number of peers	200
download capacity	800 kbit/s
upload capacity	80 kbit/s
ul/dl slot size	10 kbit/s
content categories	300
objects per category	uniform(1,300)
categories/peer	uniform(1,8)
category popularity	f=0.2
object popularity	f=0.2
object size	20 MB (all objects)
storage capacity per peer (nr. of objects)	uniform(5,40)
queue for incoming requests	1000
max pending objects	6
fraction of freeloaders in system	50%

Table 1. Basic simulation parameters

the system to discourage this behavior, such as giving higher priority to longer exchanges.

Since users are considered to be *self-interested* rather than malicious, the best way to discourage this behavior is to offer an alternative that gives them better performance at a lower cost, is useful for the system as a whole, and respects their desire not to store or share objects. For instance, consider the scenario of Figure 3.

Although peer A has no exchangeable object, it is possible to substitute a pure object exchange with a mixed object-capacity exchange as shown in Figure 3: peer B sends x to A (5 upload units), peer A forwards x to C and D (5 upload units each for a total of 10), and C and D send y to B (5 upload units each for a total of 10). In this scenario, the result is the same for C and D compared to a pure object exchange, but both A and B increase their utility, since B gets object y at a rate of 10 when he would normally only be able to get it at a rate of 5, and peer A gets object x at a rate of 5 when he would not be able to participate at all in a pure object exchange. Of course, this requires a generalization of the exchange mechanism to non-ring topologies, which we do not discuss or analyze further in this paper.

3. Simulation

We simulate a small, 200-node file-sharing system where each peer has fixed and asymmetric upload and download capacity *e.g.*, the available capacity is not affected by other user traffic and there is typically much more download than upload capacity. We assume that the core network is sufficiently overprovisioned, delay and loss are negligible so that the only bottleneck in the system is a peer's connection.

The object popularity model is similar to the model presented in [22] and consistent with real-world measurements[16]. Objects are organized in categories. Each peer is interested in m categories, which are selected at initialization time. The popularity of a category of rank i is computed as $F_c^i = 1/(1 + i.f_c)$ *i.e.*, the probability of a request for an object in category i is $p_c^i = F_c^i / \sum_{j=0}^{m} F_c^j$. For each of the m categories assigned to each peer we also assign a local preference distribution with uniformly random weights for each category. The local preference distribution is independent of global popularity. When a peer issues a request, it chooses a category based on the local preference distribution, and then picks an object in that category, also based on a distribution where the popularity of an object of rank i is computed as $F_o^i = 1/(1 + i.f_o)$ and the probability of a request for an object in category i is $p_o^i = F_o^i / \sum_{j=0}^{m} F_o^j$. For $f = 0$ the distribution becomes uniform, and for $f = 1$ it becomes Zipf-like. Note that measurements of real-world file-sharing systems suggest zipf-like locality[9].

Each peer is allowed a maximum number of pending requests. The request rate is fast enough so that this maximum is reached, and held, at an early point in the simulation, and throughout each experiment a new request is issued as soon as a pending download is completed. Each peer can store up to a maximum number of objects. We initially place objects on each peer based on the peer's category preferences. In regular intervals, peers examine their storage and remove random objects if the maximum number of objects is exceeded. A peer postpones removing an object if it is used in an ongoing exchange. Although object popularity is zipf-like, the presence of a cache makes it likely that any given peer will request a file only once (afterwards, it hits in the cache). This is both consistent with real-world measurements (as in [16]), and conservative with respect to finding exchanges.

The system parameters for simulation are shown in Table 1.

3.1. Results

The key metric for peer performance in file sharing systems is object download time. We measure the mean object download time for sharing and non-sharing users in a non-exchange, two-way, 5-2-way (*e.g.*, choosing longer over shorter exchange rings), and 2-5-way (*e.g.*, choosing shorter over longer rings) exchange system. In Figure 4 we report behavior while varying system load. As expected, as the upload capacity is reduced, the mean download time

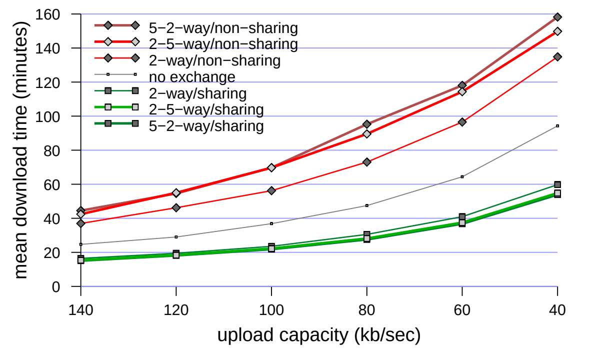

Figure 4. Mean download time vs. upload capacity and exchange policy.

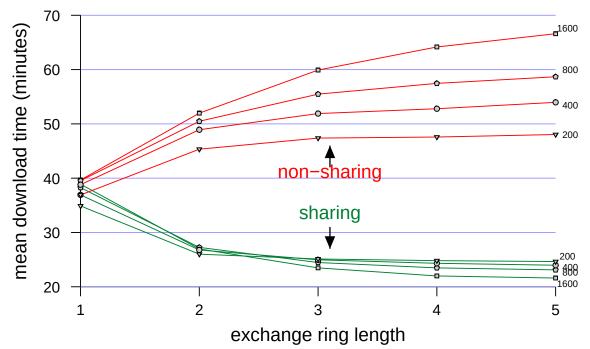

Figure 5. Mean download times vs. maximum exchange size and the number of peers in the network.

increases for both sharing and non-sharing users, but increases faster for non-sharing compared to sharing users. This happens because as the system gets more loaded, resources are shifted to sharing users because we prioritize exchanges over non-exchanges and so a larger fraction of the upload resources can be given to users that can participate in exchanges. For 40 kbit/s upload capacity the use of two-way exchanges results in download times for sharing users that are less than half of the download times for non-sharing users. The use of higher-order exchanges in addition to two-way (denoted as 2-5-way and 5-2-way in the graph) gives sharing users four times better performance than non-sharing users. When using the exchange mechanism the improvement for sharing users is also significant compared to a system where no exchange mechanisms are introduced ("no exchange" in the graph): downloads are roughly twice as fast when exchanges are used. This observation suggests that sharing peers have a good incentive to deploy the proposed exchange mechanism.

The benefit of seeking and using higher-order exchange

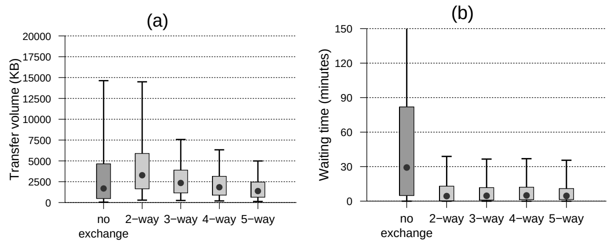

Figure 6. Distribution of transfer volumes and transfer starting times per traffic type

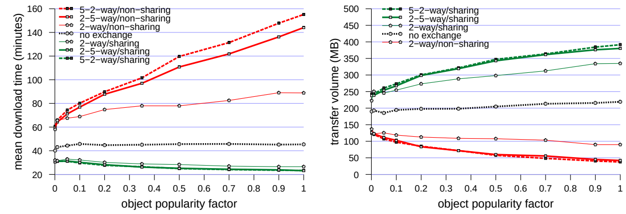

Figure 7. Mean download time and transfer volume vs. object popularity factor

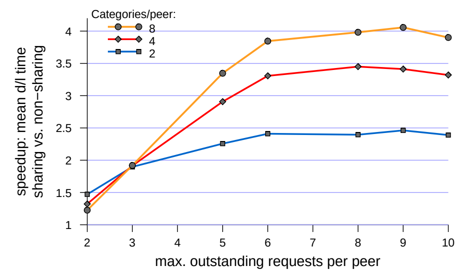

Figure 8. Ratio of mean download times for different maximum pending request sizes and number of categories each peer is interested in

rings as well as the effect of scale (in terms of network size) is shown in Figure 5. We observe that there is a significant difference between $N = 5$ and $N = 2$, suggesting that higher-order exchanges are indeed valuable. However, much larger rings ($N > 5$) do not offer any substantial improvement. We also see that as the network grows in number of nodes, the difference in performance between sharing and non-sharing users also increases.

In Figure 6(a) we present the distribution of the amount of data transfered per session. We see that exchanges have a higher transfer volume, as normal transfer sessions tend to be canceled and replaced by exchanges. We also observe that the transfer volume is much higher for shorter rings than for longer, as there is a higher probability of a peer completing a transfer and therefore dropping the exchange when there are many peers compared to when there are only two peers. This helps explain why higher-order exchanges contribute less than two-way exchanges to the overall improvement. Note that in the current simulation model peers always pick the first feasible exchange in the search process. There may be other feasible exchanges with a longer expected life-cycle. Thus, the system could be modified to determine the best possible exchange, at the expense of increased search time and cost.

In Figure 6(b) we present the distribution of the waiting times for different classes of transfers. The waiting time for a session is the difference in time between the original object request and the start of a transfer. We see that waiting times for non-exchange transfers are substantially worse than for exchange transfers, as the system gives absolute priority to exchanges. This difference is the key reason why exchanges provide significantly better performance to sharing users. The waiting time is only slightly worse for higher-order exchanges compared to two-way exchanges, meaning that this is not the cause for the relatively smaller benefit of higher-order exchanges.

We also determined the effect of the object popularity distribution on performance. In Figure 7(left) we show the mean download time for different types of exchange configurations as a function of the object and category popularity factor f. As expected, the difference in performance between sharing and non-sharing users increases as the factor f approaches 1, resembling a zipf-like distribution, although the relative benefit is significant even when object popularity is more evenly distributed. In this experiment, the difference between 5-2-way and 2-5-way exchanges becomes clearer, and it seems that 2-5-way perform slightly better, not because they improve the performance of sharing users but because they reduce performance of non-sharing users. This happens because exchange transfers displace non-exchange transfers and 2-5-way are longer-lived on average than 5-2-way, even if they have similar aggregate transfer volumes, as shown in Figure 7(right). Because they have similar transfer volumes they do not affect the performance of sharing users as much, but, as they are longer-lived, they tend to displace non-exchange transfers for a longer time. These results indicate that giving preference to two-way over higher-order exchanges is a good engineering choice, in addition to being cheaper in terms of search cost.

In Figure 8 we present the ratio of mean download times between sharing and non-sharing users as a function of the maximum number of outstanding requests on each peer as well as the number of categories each peer is interested in. The maximum number of outstanding requests increases system load, but also increases the number of feasible exchanges in the system. Up to a point this results in a better download time ratio for sharing users, as the fraction of the total system capacity devoted to exchanges tends to increase. Beyond this, the improvement levels off and even decreases as the maximum number of outstanding requests increases. This can be explained by the increased competition between sharing users that seems to reduce their relative benefit, although the reduction does not appear to be significant.

We must note that since we do not explicitly model idle peers that have no outstanding requests, this also provides an indirect measure of the effect of idle users on system performance. Idle users do not participate in exchanges and therefore do not discriminate between sharing and non-sharing peers, dampening the effect of exchanges on relative performance.

The effect of the number of categories per peer is also significant, as it generally increases the probability of locating a feasible exchange. If the number of maximum outstanding requests is small, the effect appears to be reversed, with more categories per peer giving a slightly smaller relative benefit to sharing peers.

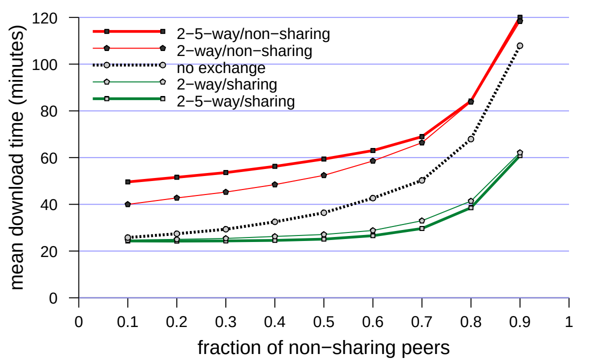

Figure 9. Mean download times vs. fraction of non-sharing peers.

All of the previous simulations assumed a fixed fraction (50%) of peers were good citizens and shared. Figure 9 investigates the effect of frequency of uncooperative behavior on mean download times, to see whether incentives to share continue even if the vast majority of peers do not cooperate (or, contrarily, if almost everyone cooperates). The measurements show that the gap in mean download times persists, regardless of the fraction of non-sharing nodes. The explanation for this is straightforward. We use the "no-exchange" case as a baseline, *i.e.,* mean download time in a system in which every transfer is granted and no preference is given to sharers. When almost everyone is sharing, then sharers get the same performance as no-exchange (sharers rarely get an advantage from sharing), however, the non-sharers get a large penalty. On the other hand, when almost everyone is non-sharing, they rarely compete with a sharer, so the non-sharers receive the same performance as "no-exchange". However, the infrequent sharer gets a big reward, because they are almost always able to preempt other transfers and get immediate service.

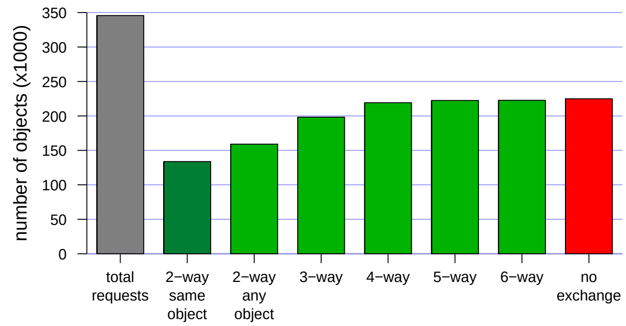

Figure 10. Fraction of requests that can be served in an exchange ring with other nodes in the dataset

4. Measurements

We measured the eMule network in order to determine the applicability of exchange mechanisms in a real system and validate our simulation results (details of our measurement methodology and results can be found in [8]).

Our measurements show that more than 75% of the peers share at least 7 files, excluding incomplete files from ongoing transfers whose parts are also made available in the network. Assuming that the high percentage of freeloaders reported in other studies also applies to the nodes in our dataset, this suggests that many users refuse to commit upload capacity to the peer-to-peer network, although they do have a number of files stored on disk that can serve requests. It was encouraging to observe that only a very small fraction of peers in our dataset did not have any pending outgoing requests. Peers that do not have outgoing requests do not participate in exchanges, and so they provide their resources equally to both sharing and non-sharing users.

Figure 10 shows the fraction of requests can be partially or fully served in an exchange ring with other nodes in the dataset. We see that almost all requests in the system (excluding those for which there is no corresponding file on any node in our dataset) can be served using an exchange ring. We also observe that small, 3-way rings are sufficient for almost all requests and that there is a noticeable difference between 2-way, same-object exchanges as provided by systems like BitTorrent and 2-way exchanges without the same-object restriction.

5. Discussion

For the purpose of this study we have focused on a rather simplistic simulation scenario, and a specific file-sharing model. We discuss some of the limitations of our analysis and how the mechanisms proposed could be improved further.

In terms of simulation, the basic assumptions (*e.g.*, over-provisioned core network, asymmetric bandwidth, zipf-like popularity) seem to agree with real-world measurements. Many of the other characteristics of the current model tend to err on the conservative side. Firstly, we assume that a peer cannot serve an object unless it has been fully received. In reality, many peer-to-peer systems do serve "chunks" of incomplete objects. If this is incorporated in the model, the opportunity for exchanges is likely to increase further. In fact, this form of exchange is implemented in the BitTorrent system[11].

Second, in our simulation all peers have very similar characteristics, ignoring the heterogeneity observed in real-world systems. For example, the existence of "super-peers" (*e.g.*, peers with substantially better network capacity or storage) is likely to have a positive effect on exchange mechanisms, especially as a way to stimulate the deployment of exchange-capable clients.

Third, we have ignored the complexity issues of communicating request tree information. The cost of communicating the full request tree may be prohibitive for peers with a large number of incoming requests and peers close to them in the request graph. If the request tree is updated incrementally, this is likely to introduce some latency in the search process which is not reflected in the waiting times of exchange transfers in the current simulation model. However, there are ways of reducing this cost. In particular, we can exploit the fact that the majority of exchanges are either 2-way or 3-way. Each peer P that requests an object from A need only send its own request, and include the top level of its own IRQ (*i.e.* the peers at depth 2 on P's IRQ). A then has enough information to completely determine whether any 2- or 3-way exchanges exist. If none exist, then A can send its complete list of pending requests (including the list of peers serving each object) to each of the depth 3 peers in its own IRQ. For a 3rd level peer, P', on A, depths 2 and 3 on P''s IRQ correspond to depth 4 and 5 on A's IRQ. If the IRQ of P' contains any of the peers contained in A's list, then P' immediately knows that a 4- or 5-way exchange through A and P' is possible. Fortunately, this need only be done when no 2- or 3-way exchange exists.

Finally, it is hard to speculate on how the incentives provided by exchanges would affect peer behavior. One possible direction for future work is to determine how the proposed exchange mechanisms interact with replication. In an exchange system, users have an incentive to replicate popular objects that are in demand, as this is likely to increase their chances of participating in exchanges which, being prioritized, would give them a performance improvement. In essence, popular objects take the role of currency in exchange economies as they are easily exchangeable for other goods.

6. Related work

The first known use of payment-based incentive mechanisms in peer-to-peer file sharing was in the now defunct MojoNation network[4]. Each user was given an initial endowment called Mojo which he could spend on purchasing files from other peers. The main limitation of this approach is that all transactions had to be cleared in a centralized system, and users were burdened with managing their Mojo.

A distributed cash-based system for peer-to-peer systems is presented in [24]. The system uses a currency called *karma* which is maintained for each user by a collection of random participants called a *bank-set* that is located using a DHT lookup. Users need to negotiate the price of serving an object through an auction mechanism, and coordinate with the bank-set for transferring karma between accounts. Each user receives an initial amount of karma when signing up with the system, and the system ensures that the rate at which users can create new identities is limited through the use of a cryptographic puzzle. To address inflation or deflation, the system needs to periodically normalize the total amount of currency in the system. The result is a fully-fledged economic system that is in principle more flexible than an exchange system. In an idealized setting, a cash-based scheme may be able to offer a stronger performance advantage to contributing peers, as it is not subject to the *"double coincidence of wants"* constraint that drives exchanges. In practice however, the cash-based approach has two main limitations.

First, it suffers from all the complexities of currency management. If the mechanisms used to negotiate prices, adjust accounts to inflation and deflation and manage a user's budget are not made completely transparent to the user, then such a system is likely to have a high cost in terms of *user attention*[15] which is suggested as a major reason why micropayment schemes are unlikely to get wider acceptance in general[19]. The feasibility of such mechanisms has not been proven to date.

Second, the need to provide start-up funds to new users creates a potential loophole in the economy. Specifically, the cryptographic puzzle used for protecting against the creation of new user identifiers and transfer of credit to existing active users may not be sufficient, as it may not be able to offer a satisfactory trade-off between keeping fake accounts out and allowing legitimate new users in. In essence, it is possible to earn cash in return for CPU cycles, without doing any useful work for the system.

A lightweight, *two-way* credit system is implemented in the eMule system[6]. The goal of the credit system is to reward users contributing to the network by reducing their waiting time in the upload queue. For each request in the upload queue the peer computes the *Queue Rank* based on a scoring function that depends on the current waiting time for the request, as well the upload and download volumes for the peer. The main advantage of this scheme is simplicity: there is no communication overhead and a peer only needs to maintain the upload and download volumes for each peer it has communicated with. The approach is cheat-proof in the sense that peers cannot benefit from tampering with the credit file. However, anecdotal evidence[5] suggests that the approach does not consistently provide a clear performance advantage to sharing users. Although there is no clear evidence in terms of measurements to determine precisely why this is happening, the credit approach appears to have two main limitations.

First, it is hard for a peer to strategize in terms of what peers he wants to earn credit from in order to maximize expected benefits. A large fraction of peers may be temporarily disconnected resulting in delays in rewarding credit; other peers may leave the system permanently, resulting in loss of credit; others may not have any object of interest and some may not share content at all. The use of "waiting time" as a factor in computing the queue rank further complicates this problem. It results in giving weaker performance advantage to users with established credit, as peers that do not have any credit can still use the system if they are patient enough. Tuning the scoring function to reduce the effect of waiting time is possible, but results in never serving users that don't have established credit, even if establishing credit with those peers could be beneficial in the future.

One practical workaround to address this problem[2] is to control the set of shared files in a way that increases credit with peers likely to be useful for a given set of requests in the near future. For instance, if a peer is requesting an object in category C, then it makes sense to limit sharing to only those objects that are already available and belong to category C. Assuming that remote peers sharing the requested object are likely to request objects from the same category, the peer is more likely to earn credit and therefore improve queue rank and reduce waiting time on those peers. In this scenario, the credit system essentially *approximates* exchanges, at the cost of additional effort to get the conditions right for this to happen.

A second problem with the two-way credit system is that there is no clear incentive for individual peers to cooperate in supporting the credit system. There is also no strong individual incentive not to honor credit, but in practice certain variants of the eMule client do not support the credit system, which also means that a fraction of the credit earned essentially gets lost. The mechanism does not directly penalize clients for this type of defection, and building additional protection (*e.g.*, monitoring compliance and maintaining blacklists) adds complexity.

In [10] the authors argue that peer-to-peer "bartering" is an appropriate way to bootstrap peer-to-peer economies, focusing on systems like PlanetLab[20] where peers share basic resources like computing, storage and network capacity. They propose the exchange of signed resource tickets between system participants that can be stored, traded and used for allocating resources. Strictly speaking, this is closer in spirit to credit economies involving personal debt certificates than barter. This approach is well suited for systems with a small set of homogeneous resources like CPU, storage and network capacity. In such systems the exchange mechanisms are unlikely to be useful, as there is little meaning in instant exchanges of same-type resources. In contrast, file sharing systems are content-oriented, providing a high level of specialization in terms of the objects served by each peer. This fits well with the instant exchange model. Exchange-based mechanisms are also discussed in [12] for incentivizing users of peer-to-peer storage systems to contribute resources.

The work most closely related to ours is BitTorrent, a system for large-scale content distribution where peers exchange blocks of the same file in an effort to expedite the distribution of large files [11]. The approach is more limited in that it only supports two-way exchanges on the same file, and appears to be vulnerable to freeriding middlemen (as discussed in Section 2.2). To the best of our knowledge, our study is the first to examine the effect of exchange mechanisms on peer performance and their value as an incentive mechanism in a file-sharing system.

7. Summary and concluding remarks

We have presented an exchange-based approach that provides incentives to cooperate in peer-to-peer file-sharing networks. Our approach is decentralized, and is considerably simpler than systems that provide system-wide forms of credit or cash. The basic idea is that peers give higher service priority to requests from a set of peers that can (transitively) provide a simultaneous, symmetric service in return. We describe methods for discovering sets of feasible n-way exchanges, and the methods for regulating transfers to provide incentives to share resources. We have also discussed how to guard these mechanisms against attacks by users wishing to exploit them to increase their own performance.

We have used simulation to analyze the mechanisms and determine their effect on performance. Our results show

2 This has been suggested on message boards as a strategy that has worked in practice.

that exchange mechanisms offer a significant performance advantage to cooperating users, in terms of object download times. The performance advantage is more pronounced when the system gets more loaded, and when object popularity leans more towards a zipf-like distribution. Our results also show that higher-order exchanges offer a noticeable improvement (with improvements significantly diminishing with $n > 5$), if used together with two-way exchanges. Thus, the proposed approach provides a strong incentive for users to share resources.

Acknowledgements

We thank Sotiris Ioannidis, the anonymous members of the EMule community and the ICDCS'04 reviewers for useful comments and suggestions on this work.

References

[1] Hack kazaa participation level - the easy answer. *http://www.davesplanet.net/kazaa.*

[2] Kazaa Hack 2.1. *http://kazaahack.250x.com/*.

[3] Kazaa Media Desktop. *http://www.kazaa.com/*.

[4] MojonNation. *http://www.mojonation.net/Mojonation.html.*

[5] Official eMule-Board. *http://www.emule-project.net/board/*.

[6] The EMule Project. *http://www.emule-project.net/*.

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

[8] K. G. Anagnostakis and M. B. Greenwald. Exchange-based mechanisms for peer-to-peer file sharing. Technical Report MS-CIS-03-27, Department of Computer and Information Science, The University of Pennsylvania, August 2003. (revised, December 2003).

[9] J. Chu, K. Labonte, and B. N. Levine. Availability and locality measurements of peer-to-peer file systems. In *Proceedings of ITCom: Scalability and Traffic Control in IP Networks II Conferences. Proceedings of SPIE Vol. No. 4868*, July 2002.

[10] B. Chun, Y. Fu, and A. Vahdat. Bootstrapping a distributed computational economy with peer-to-peer bartering. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[11] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[12] L. P. Cox and B. D. Noble. Samsara: honor among thieves in peer-to-peer storage. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 120–132. ACM Press, 2003.

[13] J. R. Douceur. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, March 2002.

[14] E. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[15] M. H. Goldhaber. The attention economy and the net. *First Monday*, 2(4), April 1997.

[16] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 314–329. ACM Press, 2003.

[17] J. Ioannidis, S. Ioannidis, A. Keromytis, and V. Prevelakis. Fileteller: Paying and Getting Paid for File Storage. In *Proceedings of the Sixth International Conference on Financial Cryptography*, March 2002.

[18] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. EigenRep: Reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference (WWW 2003)*, May 2003.

[19] A. Odlyzko. The case against micropayments. In *Financial Cryptography 2003, J. Camp and R. Wright, eds, Lecture Notes in Computer Science, Springer*, 2003.

[20] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of the 1st ACM Workshop on Hot Topics in Networks (HotNets-I)*, October 2002.

[21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM 2001*, August 2001.

[22] M. T. Schlosser, T. E. Condie, and S. D. Kamvar. Simulating a P2P file-sharing network. In *First Workshop on Semantics in P2P and Grid Computing*, December 2002.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[24] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A secure economic framework for P2P resource sharing. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[25] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, April 2001.