Peer-to-Peer Information Systems (Proseminar) Exchange-Based Incentives for P2P File Sharing (Kostas G. Anagnostakis, Michael B. Greenwald) Write-Up

J. Marc Roth
jmroth@mpi-sb.mpg.de

Saarland University,
Department of Computer Science

Max-Planck-Institute for Computer Science,
Databases and Information Systems Group

February 1, 2005

Contents

1 Introduction

The performance of peer-to-peer (P2P) file sharing systems largely depends on the level of cooperation of its participants. The paper[1] that I will discuss in this write-up focuses on exchange-based incentives[1].

Exchange-based incentives mean that the members (people/peers) of the network will be provided with priority service if they contribute to the system, i.e. store and share files.

Studies[2] have shown that generally the participants in a P2P network are not cooperative at all. Depending on how the system is built this may not only result in degradation of service but in a complete collapse of the system.

2 Incentive Mechanisms

This problem has started research on the topic of incentive mechanisms. The paper first introduces several existing implementations and quickly discusses their effectiveness.

[1] incentive: (n.) *Something, such as the fear of punishment or the expectation of reward, that induces action or motivates effort.*

2.1 KaZaA

Each peer in the KaZaA file sharing system[3] announces its *participation level*. This participation level is computed out of several local(!) values:

- uptime
- download volume
- upload volume

The higher the announced participation level, the more important the peer is. Since people (usually) have complete control over their own computer, they can trick the client into claiming anything they like, e.g. by using software modifications[4].

2.2 Credit Systems: cash/credit economies

Credit systems have the inherent advantage of providing one resource (e.g. money) that can be exchanged against every other resource, which makes them very flexible and thus powerful. An example is the revised Napster network, where users use money to pay for any type of data they wish to download, in this case mainly MP3s.

2.2.1 Centralized Credit Systems

These systems[2] consist of micropayments issued by a trusted server or payments approved by a central transaction clearing center (e.g. credit card company). They inherit the typical disadvantages of centralized systems, e.g. a single point of failure. In this case there is no clear incentive why a single host should be willing to carry such a workload.

That is why this load may be distributed across an own "payment-P2P-network":

2.2.2 Decentralized Credit Systems

There are some recent proposals for this type of system which are based on the concept of distributed hash tables to lookup account/credit status.[5] However these systems inherit another set of problems:

- Heterogeneous node capabilities that make it difficult to decide what work to give to which peer.
- Reconfiguration performance might be bad since peers connect and disconnect at their own discretion.

In any case, there are possible attacks that may be launched at decentralized credit systems if they become important enough for malicious users to care about and possibly gain benefit from. Furthermore, in a distributed computing environment it is practically impossible for initially unknown remote computing elements to present convincingly distinct identities without the supervision of a central trusted authority.[6]

[2] see section 8.1 for an example

2.3 Exchange Systems: exchange/barter economies

In this paper, a barter economy is used to provide the incentives discussed. In barter economies users directly trade resources between each other.

Exchange transfers are defined as transfers between peers which provide simultaneous and symmetric service in return. Requests from peers that can provide exchange transfers will be given absolute priority to non-exchange (1-way) transfers.

One big disadvantage is that if peer A wants something from peer B, peer B does not necessarily have anything that first peer A needs[3], thus not having a basis for a successful trade. This is why there is the concept of N-way exchanges. Peer B does not necessarily need to have anything peer A wants, but one of peer B's exchange partners might, so trading in circles can begin.

The concept presented here is even more general as there are not only 2-way exchanges going to be supported but N-way exchanges are also possible, examples of which can be found in figure 1. This means rings of any number of peers can be initiated, although it will be evident quickly that performance will no longer significantly increase starting at a certain ring size.[4]

Exchange economies have the advantage of not much overhead occurring because almost no bookkeeping has to be done and the complexities of currency management are avoided.

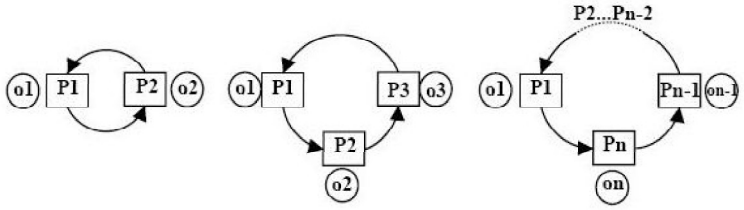


Figure 1: 2-way, 3-way and N-way ring

It is an important property of the presented mechanism that no-exchange (or 1-way) transfers are only possible if there is *no possibility at all* to create one of the above mentioned rings. 1-way transfers are always the last possible resort preventing some peer from not contributing resources to the system at all. It is also important to understand that each peer can be a part of several rings at the same time.

3 Exchange Mechanisms

First we need to define the system at hand. Each peer has a fixed download and upload capacity. Usually these capacities are asymmetric, which means there is more download than upload capacity available[5]. Since the upload link is the main bottleneck in this configuration, it will be managed the following way:

[3] "double coincidence of wants"

[4] see simulation result in section 6.3.2

[5] like consumer DSL, cable or satellite connections

Transfers are carried out with relatively large, fixed-size blocks. These blocks are also called *objects*. The system may concurrently download several parts of the same file from multiple sources.

In this paper object lookup is ignored. It may occur via message flooding as in Gnutella-like networks or via a DHT[6] approach as in Chord etc.

3.1 Incoming Request Queues

There is a data structure in each peer that contains what other peers have requested. It is called an incoming request queue (IRQ). Using that information, the peer then decides if there is some reason for it to serve a request.

3.2 Transfer Decisions

This is done by adhering to the following rules:

1. The local peer must have upload capacity, obviously.
2. One of the following conditions must be true:
   - There is a possibility for an exchange transfer, in which case a priority transfer is initiated. (also see below)
   - There is no possibility for an exchange transfer at all, in which case a 1-way (non-exchange) transfer will be initiated.

It is important to understand that the primary reason for the power of this system is the fact that *non-exchange (1-way)* transfers get preempted[7] as exchange transfers become available. Technically, this means that IRQs are searched regularly and possibilites for exchange transfers that are found are preferred as long as the upload link is not completely saturated with exchange transfers.

Exchanges are performed one fixed-size block at a time. A transfer is terminated if one of the following conditions is met:

- The communication partner disconnects. (Either because it has gone offline, crashed or doesn't want to share anymore.)
- The source object is deleted. (There is nothing to trade for anymore).
- The first transfer has been completed.

This last condition in essence means that if two peers A and B were interested in files f_B and f_A respectively and if f_A is larger than file f_B, the transfer will terminate when f_B has been received by peer A, because there is nothing left at B that A could trade for anymore. Peer B will need to get the remaining part of f_A from somewhere else, possibly by initiating other exchanges. This is possible since partial transfers are supported, as mentioned before.

All of this together means that the probability for a peer to participate in an exchange becomes higher the more it shares, thus rewarding it with a shorter download time for a requested object. These are strong incentives for peers to store objects and to share them.

[6] distributed hash table

[7] preempt: (v. tr.) *to have precedence or predominance over*

4 Exchange Transfers

The proposed approach gives complete priority to exchange transfers. But how can peers find out if and where such exchanges exist?

For 2-way exchanges this is a simple problem and was already mentioned before. Still, the "double coincidence of wants" issue, i.e. peers who requested something from me need not have what I want, remains unaddressed.

The authors of the paper try to solve it by using N-way exchanges. In order to compute N-way exchanges they use graphs with the peers of the network corresponding to the nodes of the graph and requests (not transfers!) in the network corresponding to the graph's edges.

Cycles or *rings* in this graph need to be determined. So a new data structure is needed. It is called a *request tree*.

4.1 Request Trees

Each request that ends up in an IRQ of some peer must include its own IRQ. This means that the request tree at some peer A (recursively) includes all the requests up to a certain depth of the peers that have communicated with A. An example is given in figure 2. Real-world experience shows that this depth does not need to be greater than 5.[8]

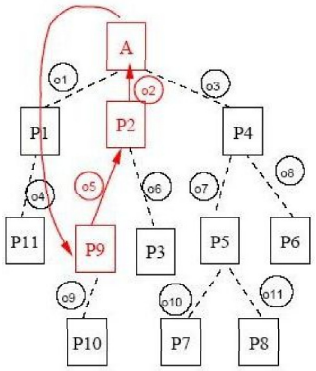


Figure 2: Request tree with exchange ring (cycle in red)

Now if A does an object lookup and determines there is some object that it desires at some peer which is present in its request tree, then it can initiate an (N+1)-way exchange, where N is the depth at which the peer has been located in the request tree. In this example A (at depth 0) located some object at P9 (depth 2) so a ring of length 3 can be initiated, in which each of the participants gets some object it desires.

It is important to understand that:

- Each peer gets served from its predecessor in the graph.
- Each peer serves an object to its successor in the graph.

[8] see section 6.3.2

This is essence means that transfers are in the opposite direction of the directed edges (representing the requests) in the graphs.

The request tree is inspected

1. before a request is sent (in order to determine whether an exchange is feasible), and
2. after a request has been received (in order to determine whether new exchanges have become available).

4.2 Tokens

In practice, the initiator of the ring must circulate a *token* to keep the ring validated. Invalidation occurs if

- a peer has gone offline (announced that it will no longer be there), or
- a peer has crashed (no announcement, but it is still no longer there), or
- other peers in the ring are trying to initiate the same ring (all peers are based on the same heuristics, so race conditions might happen).

The token also negotiates the transfer rate. It is clear that in order to stay fair, the entire ring must operate at the transfer rate of the peer with the smallest bandwidth capacity available. If a peer has more than this capacity available it may very well redistribute its excess capacity to other rings, since they are allowed to be part of several rings and may conduct several exchanges simultaneously.

5 Cheating

Cheating in this case means giving exchange-like priority to non-exchange transfers. This may be accomplished in several ways.

5.1 Serving Junk

Peer A gives real data to B, but B gives A only random data back, which is of no use to it. How can this be fought?

5.1.1 Blacklisting

- *Local Blacklisting* is inefficient because in a large system cheating each peer only once may be enough.
- *Cooperative Blacklisting* needs additional mechanisms which would make the system more complex and which might themselves be vulnerable.

Both cases offer no solution because e.g. on the Internet it is easy for a peer to obtain a new identity. This introduces opportunities to misbehave without paying reputational consequences.[7]

5.1.2 Synchronous Transfers

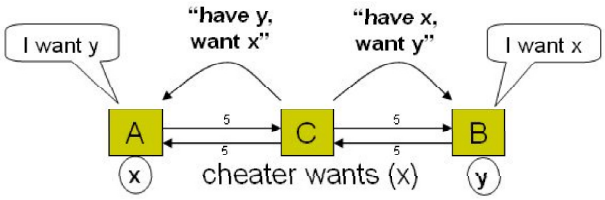
Another solution would be to transmit a new block only if what was received before was valid. Assuming a trustworthy source of checksums for those blocks existed, an exchange could still only happen at a relatively low rate, since the received blocks must be proved valid before transmitting anything new. This would take at least the round-trip time (t_{rtt})[9] of the network as well as the delay needed to validate the received object, which is ignored in this model. The maximum benefit for a cheater would be the block size ($b_{exchange}$).

The transfer rate would thus be limited by $b_{exchange}/t_{rtt}$. Hence, the slot capacity-delay product must be filled. This could be done by using some window protocol with congestion control, such as the Transport Control Protocol [8].

This means bursts of data[10] can be transmitted and same-size bursts are received back (windowing). Like the TCP Slow Start algorithm (congestion control) this window is increased when the remote peer starts earning trust, e.g. if it sends valid objects back. On receipt of valid packets, the window size is increased. If, at some time, bad objects arrive, the transfer will be terminated. A cheater would need some real objects in order to increase the window. The authors of the paper believe that this level of cooperation may already be enough to be good for the system as a whole.

5.2 Man-in-the-middle attack

How to prevent some peer from obtaining an object without doing any useful work for the system? Figure 3 would be a possible scenario.



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

Figure 3: C is a cheater!

C wants object x. So C is exchanging objects of y, which it knows A wants and which it gets from B, for objects of x from A and vice-versa. C is getting priority service although it does not contribute to the system (it has no object stored). In a sense it trades elevated upload capacity for priority service. Thus, the exchange-based incentives break down.

How can this be dealt with?

5.2.1 Encryption

One option would be to encrypt the transfer in both directions. Hence, C would only see encrypted traffic and it would not be of any use to act as a middleman. A and B are very well off trading between themselves. Technically, this could be accomplished by assigning some trusted peer as a mediator to the transfer,

[9] round trip time: the time it takes to send a packet to a remote host and receive a response
[10] multiple blocks before an acknowledgment/answer needs to occur

which would verify at the end of the transfer that only valid objects have been transferred and then give each of the endpoints the necessary information to decrypt what they have just traded.

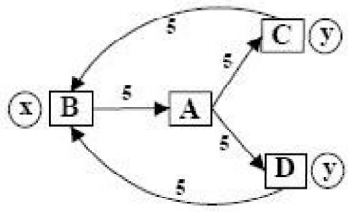
However, the paper assumes that self-interest is put above maliciousness. This essentially means:

1. Peers are *not good*. Since peers usually correspond to persons, and many persons are egoistic, i.e. don't share what they have, only want their own good and don't care about the rest of the world, it is assumed many peers also behave that way.

2. Peers are *not bad* either.[11] Users might be egoistic but usually do not take any active measures to harm the system.

5.2.2 Non-ring topologies

However it can also be argued that there are better alternatives for these cheating peers that offer them more performance at a lower cost, as can be seen in figure 4.



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

Figure 4: non-ring topology

This time A is the cheater. In a usual ring topology A would not be able to participate at all. However because of A's participation B is able to get the object it desires at twice the rate of its own upload. A still trades upload vs. priority traffic but some other peer, namely B, has an advantage because of that. This is a non-ring topology and this paper does not discuss it further.

6 Simulation

In order to find out if this theory holds in a practical application the paper's authors created a simulation, whose parameters are shown in figure 5 on page 10.

6.1 Simulation Setup

A real-world system needs to be simulated, so there is an asymmetric download/upload ratio. It is also assumed that the network is "perfect", e.g. delay and loss are neglected and the only bottleneck is the peers' upload capacity.

[11] Whether egoism is considered a bad character trait is not discussed here.
[12] freeloader/freerider: a peer that does not contribute, i.e. only downloads

number of peers	200
fraction of freeloaders[12]	50%
download capacity upload capacity	800 kbit/s 80 kbit/s
content categories category and object popularity	300 $f_c = f_o = 0.2$
object size IRQ size	20 MB 1000

Figure 5: most important simulation parameters

Each peer has a maximum number of objects that it stores. In the process of simulating, this number may be exceeded. In this case, objects will be deleted, but only if they are currently not being served to somebody else.

6.2 The Object Popularity Model

In order to simulate a peer-to-peer network in the most realistic way, peer behavior needs to be simulated, i.e. the behavior of the people controlling the peers.

The shared files are put into *content categories*. We call C the set of all such categories. Each category is uniquely identified by its *popularity rank* i, i.e. the rank of category c_i is i.

Categories are then assigned to each peer in the following way: a subset of size m is selected uniformly out of C and its elements are assigned to the peer.

f_c and f_o are the zipf-parameters for the category and object distributions, respectively. A zipf distribution means that the second most popular item is half as popular as the most popular item. The third most popular item is one third as popular as the most popular item, etc.... For an example of zipf-like distributions with different zipf parameters (which literature mostly calls α) see figure 6 on page 11.

The following terms need to be introduced and explained:

- global category popularity

$$F_{c_i} = \frac{1}{1 + i \cdot f_c}, i \in \{0, \ldots, m-1\}$$

- global object popularity

$$F_{o_i} = \frac{1}{1 + i \cdot f_o}, i \in \{0, \ldots, m-1\}$$

- local interest level

$$w_{c,p}$$

The *category popularity* is a global property. Categories are assigned to a peer using a zipf-like distribution. A zipf-like distribution is used because empirical evidence suggests that this distribution approximates real-world systems

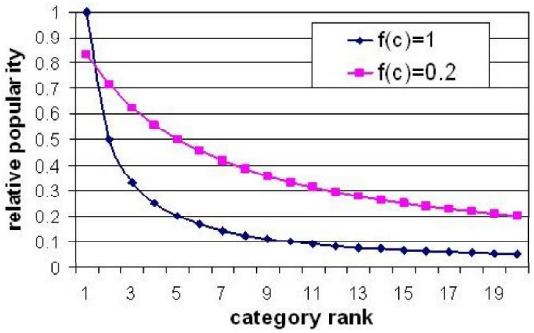


Figure 6: zipf ($f = 1$) and zipf-like ($0 < f < 1$) distribution

best.[9] Zipf-like distributions are also used in the keyword ranking of search engines, for instance.

Hence, the probability P of request for an object in category c (of rank i) is:

$$P_{c_i} = \frac{F_{c_i}}{\sum_{j=0}^{m} F_{c_j}}$$

The *object popularity* also is a global property. Objects are assigned to categories using a zipf-like distribution as well. In general, $f_c \neq f_o$, however in this case the simulation setup assumes $f_c = f_o$ (see figure 5 on page 10).

The *interest level* of a category is a local property. It is independent of the global popularity of the category. It is set for each category and each peer uniformly at random. This essentially means that while for instance adult content may be a popular category (which will probably be most requested), the local peer is most interested in downloading family movies. The local peer only requests objects for categories that it possesses. Categories are not added or removed during the course of the simulation.

6.3 Simulation Results

The key performance indicator in file sharing systems is object download time. In this section we will compare download time to several other variable factors of the system.

Some charts use the notion of 2-5-way transfers and 5-2-way transfers. *2-5-way transfers* mean the initialization of a 2-way ring when a 5-way ring for the same object would also be available, e.g. preference of shorter over longer rings. *5-2-way transfers* mean the initialization of a 5-way ring when also a 2-way ring for the same object has been located, e.g. preference of longer over shorter rings. *No exchange* means a 1-way transfer.

6.3.1 System Load

When upload capacity is reduced, the P2P network gets more loaded. Obviously, downloads then take longer. But the main point here is that the difference (in

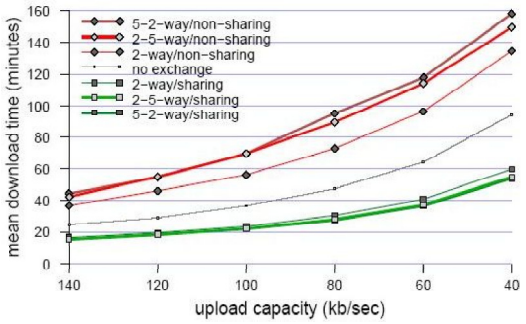


Figure 7: Mean download time vs. system load and exchange policy

this case loss) in performance for non-sharers increases faster than for sharers. This happens because resources are shifted to sharing users when the system gets more loaded because exchanges are given priority. We can see this in figure 7 on page 12.

6.3.2 Ring Size

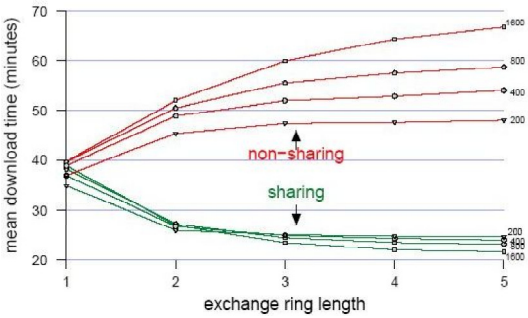


Figure 8: Mean download time vs. ring size and number of peers

We observe that there is a considerable improvement in the mean download time when increasing the ring size from $N = 2$ to $N = 5$. However, for $N > 5$ there is no significant increase in performance anymore, as shown by figure 8 on page 12.

6.3.3 Object Popularity Factor

The difference in performance between sharing and non-sharing users increases as the object popularity distribution f_o approaches 1 (zipf). Sharing users benefit because non-sharing users get penalized a lot, as seen in figure 9 on page 13.

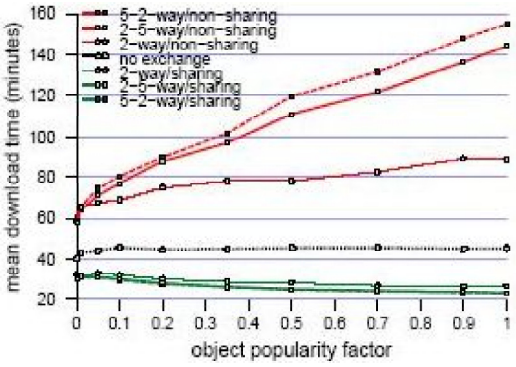


Figure 9: Mean download time vs. object popularity

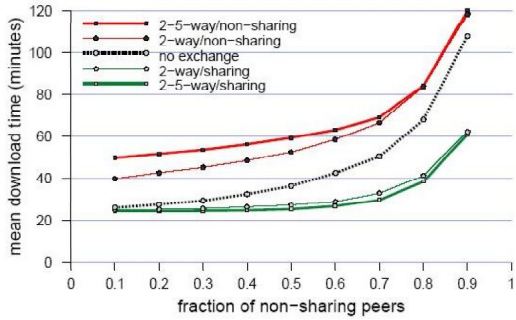


Figure 10: Mean download time vs. fraction of free-riders

6.3.4 Freeriding Ratio

We need to consider the two extreme cases. The incentives to share persist when the network reaches a state in which almost no-one or almost everyone shares. This can be seen in figure 10 on page 13.

Compared to the no-exchange (1-way) case sharers get the same overall performance as 1-way users when almost everyone is sharing (toward the left in the chart) because they rarely have an advantage from sharing. The non-sharer however gets heavily penalized.

In the other case where almost no-one is sharing (toward the right), 1-way members get the same performance as non-sharers since they rarely compete with a sharer.

It is important that in this model even the infrequent sharer gets a big reward because by starting to share he will get immediate service.

7 Limitations and Improvements

Peers are assumed to be equally interested in all of their requests. It would be useful to investigate non-uniform models, as many existing file sharing systems already do. They allow users to express their preference in terms of "low",

"medium" or "high" priority. Peers can leverage their ability to participate in exchanges only when they have pending requests themselves. It would be desirable to provide a stronger incentive for users to actually stay connected even if they cannot immediately participate in some exchange.

There will now be a short discussion about how the proposed solution mainly differs from real-world systems.

7.1 Chunks

Start-of-the-art file sharing systems can serve chunks of incomplete objects. This means that while some download is not yet complete, other peers can already get parts of what has been downloaded so far. This was not taken into account in the simulations, but it is assumed that including this into the model would increase the likelihood for exchanges.

7.2 Heterogeneity

Real-world systems are not homogeneous. There are slow computers and network connections and there are fast ones. In the real world, peer characteristics are different for each peer.

The existence of some number of peers with superior power is believed to have a positive effect on the system.

7.3 Complexity

For peers with large numbers of incoming requests, transmitting their request trees when they issue requests themselves, may be prohibitive to their performance. This may end up in latency in the search process, which this simulation did not take into account.

It is however mentioned that there are ways to reduce this cost by doing incremental request tree transfers. This means since 2-way or 3-way transfers are most common[13], an attempt will first be made to initiate such cycles by sending only a request tree pruned to depth 2 with each request and only if this is not possible, then the lookup depth is increased.

7.4 Peer behavior

There can be no general strategy for peers on how to behave when confronted with the given incentive mechanisms. One could argue that the proposed mechanisms would stimulate peers into the replication of popular but infrequently held objects, which would of course increase the possibilities for exchanges to happen. This may be a possibility for future work to find out.

8 Related Work

8.1 MojoNation

MojoNation is a centralized payment-based file-sharing network. The currency used is called a *Mojo*. The disadvantages of such centralized systems have been

[13] as has been empirically determined in section 6.3.2

mentioned before.[14]

8.2 KARMA

Karma in fact is not the name of the system but the overall standing of each participant in the system. It is represented by a single scalar value, called their karma. So KARMA is in essence the currency of the system. It is a cash-based distributed system within a secure economic framework for peer-to-peer resource sharing.[5] Random participants of the distributed system form what would otherwise be the centralized clearing house, which is called *bank-set* here. There even is an auction mechanism in place which lets users negotiate the price for a given resource. One interesting approach of this system is the limitation of new identities. Peer-to-peer systems have the problem that identities cannot easily be related to entities one-by-one.[6] In this system a cryptographic puzzle is used to limit the rate at which new identities can be created. In essence it simulates a fully-fledged economic system, including inflation, deflation, ...

Since these systems generally are powerful there are two main disadvantages:

- It suffers from the complexities of currency management and as such is unlikely to get wider acceptance in general.
- The cryptographic puzzle is a nice idea, but this also means that CPU cycles can be traded against a new identity.

8.3 eMule

eMule is a light-weight two-way credit system. Each peer remembers certain values about peers it has seen:

- waiting time
- download volume
- upload volume

These are then used to compute the so-called *Queue Rank*. This is a very simple approach and cannot easily be cheated. Peers have no reason to tamper with the credit file since it doesn't contain any information about themselves, in contrast to KaZaA[15]. Looking at the discussion in several online forums it seems however that there is no clear advantage to sharing users here. Anyway, this system has got two disadvantages:

- The little information in the credit file is of no real use to the peer to build up a strategy on how to maximize its expected benefits.
- There is no clear incentive for peers to cooperate or not to cooperate in supporting the credit system.

[14] see section 2.2.1
[15] see section 2.1

8.4 PlanetLab

Until now, we focused on sharing and distributing information over P2P networks. This can however be generalized into sharing any kind of resources like CPU cycles, storage, network capacity, ... The PlanetLab system [12] does exactly that. This approach works best in systems that are heterogeneous as in that case there is a strong incentive to trade different types of resources.

8.5 BitTorrent

The authors of this paper see the BitTorrent system as the closest approach to what they have designed.[11] BitTorrent however is only a 2-way same-object exchange. So the authors of this paper have essentially generalized the BitTorrent system into N-way any-object exchanges.

9 Summary

The presented paper provides an exchange-based approach with incentives to collaborate in a P2P file-sharing network. The approach is decentralized. It is considered simpler than other forms of credit or cash since it is an exchange or barter economy, i.e. resources are traded 1:1.

The power of this approach can be summed up in two main points:

- Peers give absolute priority to exchanges, e.g. other peers that will provide a symmetric and simultaneous transfer in return.
- A non-exchange transfer gets preempted if the possibility for an exchange arises.

Cash or credit economies have the disadvantage of being complex but they do have one resource that can be traded for all others (e.g. money). To overcome the deficiency in this approach, an algorithm and the corresponding data structures to find N-way exchanges is introduced. There are methods that allow to cheat this mechanism and ways to prevent it.

Then, a simulation environment has been set up, taking care of its applicability to real-world systems by using probability distributions that approximate a realistic behavior. The results of this simulation have been analyzed, with the mean download time being the key metric.

The simulation results have shown that always either one of these two scenarios occurs:

- Sharing users have *absolute* advantage over non-sharing users, because they can preempt other non-priority transfers.
- Sharing users have *relative* advantage over non-sharing users, because non-sharing users get penalized in this model.

The results proved what was said earlier, namely that exchanges with $N > 5$ need not be computed since the increase in performance will no longer be significant.

10 Conclusion

In my opinion, the simulation results cannot be directly applied to a real-world network.

The idea of at least punishing users which do not contribute is alright, although similar solutions already exist in current implementations of file-sharing systems.

The simulation makes a lot of assumptions about the network and the peers which do not hold in a network like the Internet.

The authors were somewhat reluctant and the paper was inconcise in several points, which will now follow.

Object lookup (in networks like Gnutella which use broadcasts) may very well consume very much of the available bandwidth, so allowing *any* kind of object lookup may be a very bad idea.

The exchange mechanisms (section 3) are essentially the bandwidth management of this approach. They have not been defined in very much detail.

It is not clear how exactly encryption helps preventing man-in-the-middle attacks and how this would affect the performance of the system in means of generated overhead.

The authors "believe" that super-peers may have a good effect on their system. But what about peers with less performance than the average peer. Are they bad for the system? I think they simply get less because they can give less.

Simulation setup suggests that delay and loss are neglected. This is a foolish assumption since this could have fatal effects on e.g. modem users. Simulation setup also assumes that files are never deleted when transfers are still happening. I personally delete whatever data that I don't need, e.g. already put on CD, etc..., in order to make room, independent of who might still be connected. There has been no clear statement or simulation about what will happen when connections are interrupted, peers disappear, reappear etc....

The authors provide references to a paper of their own (mostly identical to the one discussed here) which does not contain the expected information, especially about validation of their results by using a dataset gathered from the eMule network, a section which I did not include in this summary because these results cannot be backed up.[10]

The authors seem to have a bad opinion about the general possibility of trading CPU cycles against other resources, like generating new identities in KARMA-based networks. I would like to provide an example from a different area. There are proposals which use (inexpensive) electronic stamps for e-mails in order to fight spam. Let's say the regular user needs to pay a small amount of money (e.g. 0.1 cent per mail). This means for the spammer who sends millions of mails that he needs to pay thousands of Euros. What I meant to say is that this already is a good idea in the sense that it probably won't scale for cheaters. Staying within certain limits costs almost nothing but if I exceed this limit and start to harm the system it will become very expensive very quickly.

What is also important to note is that delay/round-trip time were not simulated. Starting downloads from a modem user may put the modem user into serious trouble.

References

[1] Kostas G. Anagnostakis, Michael B. Greenwald. *Exchange-Based Incentives for P2P File Sharing.* 24th International Conference on Distributed Computing Systems (ICDCS'04), Tokyo, Japan, Mar. 2004.

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

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

[4] K-Hack 2.6. *http://www.khack.com/.*

[5] 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.

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

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

[8] M. Allman, NASA Glenn/Sterling Software, V. Paxson, ACIRI/ICSI, W. Stevens Consultant. *TCP Congestion Control.* http://www.networksorcery.com/enp/rfc/rfc2581.txt, April 1999.

[9] 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 (SOSP03), pages 314-329. ACM Press, 2003.

[10] 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).

[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. 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.