

# Review of Incentivization Methods in MAS

Combatting Selfish Peers and Corrupt Resources

*Wouter L. de Vries*

Delft, University of Technology  
Parallel and Distributed Systems

## Abstract

Agents in a file-sharing P2P network need incentives to provide resources and a mechanism to detect trustworthy information about resources. Several incentivization methods will be reviewed for use in the Tribler project. Tribler is based on Bittorrent and uses trust for resource exchange, but it has no mechanism in place yet to detect correct resources. Incentivization techniques based on credits, trust and reputation will be discussed.

## Contents

1	Introduction . . . . .	2
2	Problem Statement . . . . .	3
2.1	P2P File Sharing Networks . . . . .	4
2.1.1	Free-riders . . . . .	4
2.1.2	Corrupt and Fake Files . . . . .	5
2.2	Attacks . . . . .	5
2.2.1	Running Example . . . . .	5
2.2.2	Whitewashing . . . . .	5
2.2.3	Collusion . . . . .	6
2.2.4	Sybil Attack . . . . .	6
2.2.5	Traitors . . . . .	6
2.2.6	Sabotage . . . . .	6
3	Credit-based methods . . . . .	7
3.1	Real money . . . . .	7
3.1.1	Micropayments . . . . .	7
3.1.2	Macropayments . . . . .	9
3.1.3	Second Life . . . . .	9
3.1.4	Fraud . . . . .	10
3.2	Fictional credit . . . . .	10
3.2.1	Non-fungible . . . . .	10
3.2.2	Fungible . . . . .	11
3.2.2.1	Designated bankers . . . . .	11
3.2.2.2	Related bankers . . . . .	12

3.2.2.3	I Owe You . . . . .	12
3.3	Credit transfer . . . . .	13
3.4	Discussion . . . . .	13
4	Trust-based methods . . . . .	14
4.1	Pairwise Resource Exchange . . . . .	15
4.1.1	Scoring . . . . .	15
4.1.2	Tit-for-tat . . . . .	15
4.2	Cyclic Resource Exchange . . . . .	17
4.3	Correlated Opinions . . . . .	17
4.4	Adaptive stranger policies . . . . .	18
4.5	Aspects of Trust . . . . .	18
4.6	Discussion . . . . .	19
5	Reputation based methods . . . . .	20
5.1	Majorities rule . . . . .	21
5.2	Weighted opinions . . . . .	22
5.2.1	EigenTrust . . . . .	22
5.2.2	Maxflow . . . . .	23
5.3	Blacklisting . . . . .	24
5.4	Transitive Trust . . . . .	25
5.5	Ontology of Reputation . . . . .	26
5.5.1	Social dimension . . . . .	26
5.6	Policies . . . . .	27
5.7	Discussion . . . . .	27
6	Discussion and Conclusion . . . . .	27
6.1	Free-riding . . . . .	28
6.2	Fake files . . . . .	28

## 1 Introduction

Due to increased anonymity people act selfishly online, which leads to a specific problem in P2P (peer-to-peer) file-sharing networks. The name file-sharing networks is probably wrong, because most people use them as file-acquisition networks. This is not a problem when only a few people act like that, but when no one uploads no one is able to download either. This behavior is called leeching, free-riding or free-loading. We will examine several methods that give incentives to agents to provide services to other agents.

Another problem in P2P file-sharing networks is the widespread availability of fake files. For various reasons people have been injecting mislabeled files in these networks to confuse regular users. Agents therefore need methods to truthfully communicate metadata of resources with each other.

Section 3 describes several credit based methods, based on real and virtual currency. Section 4 describes several methods based on trust and section 5 elaborates on these trust methods by introducing the reputation concept. But first the problems of free-riding and fake files will be looked at in more detail.

## 2 Problem Statement

In a MAS (multi agent system) agents interact with each other by providing and receiving resources. Each agent has its own goal which it tries to achieve. To reach its goal an agent usually needs to acquire resources from other agents, but providing resources is usually not a goal for agents. Agents request resources from each other. The result of these requests is not always successful, agents will not always receive the resources they requested. There are several reasons why an agent may not receive the resource it expected.

In the following scenarios I will use agents  $A_1$  and  $A_2$ . Agent  $A_1$  wants resource  $a$  and possesses resource  $b$ , Agent  $A_2$  wants resource  $b$  and has resource  $a$  in his possession.

- The goal of agent  $A_2$  is to propagate resource  $b$  as much as possible in the network. By pretending it to be resource  $a$  which is far more desirable by other agents in the network it can lure another agent into requesting resource  $a$ . They will actually receive resource  $b$ , which they do not expect and probably do not need. Another possibility is that agent  $A_1$  has only misidentified resource  $b$  for resource  $a$  and ignorantly propagates a mislabeled resource.
- Agent  $A_1$  has made a deal with agent  $A_2$  to exchange resource  $b$  for resource  $a$ . After agent  $A_2$  has received resource  $b$  it chooses not to deliver resource  $a$ .
- An agent may not be able to deliver a resource it has promised due to external circumstances like a broken network link.

This list is far from exhaustive but I have listed the situations for which I would like agents to be able to estimate in advance whether they will receive the resource they expect. In most situations agent  $A_1$  and agent  $A_2$  are strangers to each other which means agent  $A_1$  has no personal history with agent  $A_2$  to assist the estimation.

The outcome of an interaction with an unknown agent can be modeled by the *prisoner's dilemma*, which can be explained as follows. Two prisoners,  $a_1$  and  $a_2$ , are caught doing a crime and their jail time depends on whether they betray each other or not. When both prisoners remain silent they get the *reward*. However when one betrays the other the betrayer gets the *temptation* and the other is played for a *sucker*. When both betray each other they are both *punished*.

Let  $R_i$  be the event that agent  $a_i$  gets a reward,  $T_i$  the event that agent  $a_i$  receives the temptation,  $S_i$  the event that agent  $a_i$  is played for a sucker and  $P_i$  that agent  $a_i$  is punished.

	$a_1$ is silent	$a_1$ betrays
$a_2$ is silent	$R_2, R_1$	$S_2, T_1$
$a_2$ betrays	$T_2, S_1$	$P_2, P_1$

Tab. 1: Prisoner's dilemma

The jail time they get is based on the outcome of the statements of both prisoners and the duration is ordered as following  $T_i < R_i < P_i < S_i$ . Even though the combined jail time is shortest when they both keep silent, it is most beneficial to betray each other. When  $a_2$  remains silent it is most rewarding for  $a_1$  to betray him ( $T_i < R_i$ ). Even if  $a_2$  betrays  $a_1$  should betray as well ( $P_i < S_i$ ). In both cases a prisoner is better off betraying the other. Things get more interesting when agents interact with each other more and strike a deal to both keep silent because in the end  $2 * R_i < T_i + S_i < 2 * P_i$ .

## 2.1 P2P File Sharing Networks

The problem of dealing with unknown agents is especially present in P2P (peer to peer) file sharing networks. The goal of agents in a P2P network is to acquire files that the owner of that agent requests. In contrast to the client-server model resource exchange is not one way, agents in P2P networks are all equal and each agent can send and receive resources to and from other agents. People log into these P2P file sharing networks to retrieve files. For regular users distribution of their own files is not a goal, which causes the free-rider problem.

### 2.1.1 Free-riders

Many people act altruistic in their relations with other people for various reasons. They may like the reactions they get or they may fear consequences of selfish behavior. When people act selfishly toward other people in person they will most likely not get away with it easily. Other persons will not tolerate this behavior and reciprocate by acting unfriendly as well. They will also most likely inform other people of the rude behavior. When people interact online it is less likely that they will suffer the consequences of selfish behavior and will therefore behave less altruistic than they would otherwise.

Measurements of sharing behavior on the Gnutella network in 2000 [1] of Gnutella users showed that nearly 70% of the users do not share anything at all and that 50% of the responses are returned by the top 1% sharing users. Measurements from 2001 [30] report that 26% of the Gnutella users share no files and 20-40% of the Kazaa users share almost no files. A large amount of users also misrepresented their connection speed to discourage people from downloading. Although these two measurements differ dramatically in numbers it is clear that many people behave in a selfish fashion.

Peers that behave in this fashion are called free-riders. Some state that a peer is free-riding when it provides less service than it receives. Others are more strict and classify a peer as a free-rider when it provides less than it is able to provide. The following definition will be referred to from now on when free-riders are discussed:

**Definition 2.1.** [*Free-rider*] *An agent that does not provide as many resources to other agents as it is reasonably able to.*

Ideally as incentive would turn free-riders into altruistic users. If more people share files downloading files will be faster and more importantly, it will become easier to find rare files.

### 2.1.2 Corrupt and Fake Files

Another problem found on P2P networks is that there exist many files with wrong file names or descriptions. A measurement of the Kazaa network in 2005 indicated that more than 50% of the popular files were corrupt[19]. The existence of corrupt files is attributed to content *pollution* and content *poisoning*[5]. Deliberately providing fake files is called poisoning and accidentally providing corrupt files is called pollution. Each peer has a limited download speed and downloading fake or corrupt files should be prevented by all means. It does therefore not matter whether a file is corrupt due to pollution or poisoning.

There are several incentives for people to poison a network with fake files. Lately front offices of organizations like the RIAA and the MPAA<sup>1</sup> have been injecting fake or corrupt files to deter people from downloading copyrighted content. Another source of fake files are people that rename resources willfully or not. Many resources with names that indicate them to be a regular movie are actually porn movies on the Edonkey network. Lastly some people have discovered P2P networks as a distribution network for advertisement.

Giving agents incentives not to share corrupt content is futile. Poisoning is not done deliberately but by accident and punishment may give people incentive to screen their sharing behavior. Pollution on the other hand is a deliberate act based on incentives outside the scope of a P2P network, some people even earn money with it.

## 2.2 Attacks

There are several methods agents can use to exploit other agents. Such an agent hides its devious intentions while stealing resources or providing corrupt resources. Whether an agent can use any of these methods depends on the interaction protocol of the network.

### 2.2.1 Running Example

To be able to explain the interaction pattern of the different types of attacks three sets of agent that will be introduced. Anna and Abby are altruistic peer that provide resources whenever possible. They however favor transactions with peers that provide resources in return. Esther and Eve are exploiting agents that try to use the system only for their own benefit and provide as little resources as possible. Mark and Marie are malicious agent with the sole purpose of harming the network by polluting resources or sabotaging the interaction protocol.

### 2.2.2 Whitewashing

A MAS is susceptible to whitewashing when agents within that network are able to rejoin the network with a new identity. This mean that agents are able to whitewash their interaction history when other agents start to distrust them. I will explain this using the running example.

Eve, Anna and Abby are all a member of the same MAS. Eve interacts with Anna and Abby by consuming their resources and giving sub par resources in return. After some time Anna and Abby become suspicious and will stop trading

---

<sup>1</sup> Recording Industry Association of America and Motion Picture Association of America

resources with Eve. When Eve notices that she is able to obtain less resources because other agents distrust her she changes her name to Esther. Abby and Anna do not know that this new agent Esther is in fact Eve and are willing to trade with her again. Each time Esthers reputation drops too low she changes her name to continue consuming resources.

It would be best to not allow agents to rejoin a network with a new identity, but this may not always be feasible.

### 2.2.3 Collusion

A collusion attack is done by introducing a large amount of agents with the same owner in a network. This swarm of agents works together to boost the reputation or credit rating of one agent in particular. In a credit based system they will all transfer their initial credit balance to one agent. In a reputation based system they will always be best of friends and refer to each other.

Agent Eve introduces several helpers – Esther, Edward, and Elvis – in the MAS. Abby has had bad experience with Eve in a previous transaction. Eve tries to initiate a transaction with Anna. Before starting the transaction Anna will inquire into the reputation of Eve. Abby will give a bad feedback, but Esther Edward and Elvis will all give very positive feedback. By introducing a large amount of helpers Eve will be able to shout down the opinion of agents that have actually had transactions with her.

### 2.2.4 Sybil Attack

The Sybil attack[9] is a special form of the collusion attack. A participant is the entity that controls an agent and normally participants only control one agent in a multi agent system. But when a participant launches a Sybil attack it spawns several agents. The general idea of a Sybil attack is similar to the collusion attack; to have more influence on other agents in the network. What makes it more dangerous than normal collusion is that when a single participant takes control over a multitude of agents the attack can be organized far more effectively.

### 2.2.5 Traitors

Traitors in the real world are friends that turn bad. This is a good analogy for traitors in a MAS. When every successful interaction, big or small, yields the same amount of reputation traitors will be able to exploit this. After a traitor has built up a good reputation with small transactions it can exploit this by cheating on one large transaction. This means they will end up with resources that are worth a lot after providing resources of only limited value.

### 2.2.6 Sabotage

An agent sabotages the network if it's only goal is to disrupt the MAS. An examples of sabotage is: Mark injects incorrectly labeled resources into the network. This example is interesting because it describes the problem of peers injecting fake files into P2P systems. There are other forms of sabotage like DoS<sup>2</sup>, we

---

<sup>2</sup> Denial of Service attack in which the attacker floods individual agents making it extremely hard for them to communicate with others.

are however more interested in those forms of sabotage that exploit weaknesses in the interaction model between agents.

Sybil attacks and whitewashing stem from the same basic problem in P2P systems. Participants are anonymous entities on the internet, the only identification for participants is their ip address. This identification is rather dodgy due to Network Address Translation (NAT) and ip address spoofing. Secondly, with the upcoming IPv6 standard participants are able to obtain huge subnets containing many address. Thirdly, many ip addresses in use are assigned dynamically. Reputation linked to an ip address will become invalid when the owner of that address changes.

### 3 Credit-based methods

The methods in this section are all based on credit exchange between agents. Credit is a currency in a MAS that is used to pay for resources. As stated earlier distributing resources is usually not part of the goal of agents, but when a payment scheme is introduced agents may be more inclined to do so.

Several payment schemes involving currency linked to real word money will be discussed. The credit values in these schemes are administered by a bank, inherently a central authority. After that several payment schemes involving currency without monetary value will be examined. Lastly we will look at the problems of trade disputes in credit systems.

#### 3.1 Real money

In the global economy people pay for goods and services with money, a specific kind of credit. This works reasonably well considering people are committed to full time jobs only to receive money in return.

In P2P systems agents are operated by real people and therefore it would make sense for these agents to pay each other with real money. Resources in these systems are usually not valuable and therefore it makes sense that agents are able to pay each other with extremely small amounts of credits. This is why these small payments are often called *micropayments*. Schemes involving larger payments are sometimes called *macropayments*.

A distinction should be made between payment schemes based on debit and credit. A credit method works similar to the credit-card payment scheme. Vendor believe that they will be payed later via the bank, because clients make a promise substantiated by an authentic credit-card. A debit method works similar to a payment scheme involving PIN-cards. Debit transactions however are handled in direct contact with the bank. Vendors need not believe that they will be payed because the bank will directly inform the parties informed whether the cash transfer succeeded. Coins can be considered debit, because they have intrinsic value as opposed to promises made by a cheque.

##### 3.1.1 Micropayments

Micropayments have a very low value and that is why agents will very probably exchange many of these. Therefore Rivest and Shamir introduced two methods for making micropayments that only put a small burden on the processing

power of agents. Their goal was to minimize the amount of expensive public-key operations and therefore they mainly resort to hash functions.

*PayWord*[27], the first system designed by Rivest and Shamir, is a credit system. Agents need to have an account with a banker to partake in PayWord transactions. Upon registration agents receive a PayWord certificate containing their name, the bank name and other information. This certificate needs to be renewed every so often, which allows the bankers to revoke certificates for agents that do not play fair. A client wishing to buy something from a vendor initiates the payment by signing a commitment to a new user-specific and vendor-specific chain of paywords  $w_1, w_2, \dots, w_n$ . The client creates these paywords in reverse order with the irreversible hash function  $h$ :

$$w_i = h(w_{i+1})$$

The client first sends the commitment containing  $w_0$  to the vendor, which is not a payment in itself. Each succeeding payword is a payment and at the end of the day a vendor can redeem all paywords it received from clients at the bank. Clients can make larger payments by skipping paywords, because the vendor can authenticate these as well using the hash function  $h$ . When a vendor receives  $w_0$  and  $w_2$  it can check the authenticity by calculating whether  $h(h(w_2)) = w_0$ .

Vendors are able to forge paywords using a brute force technique. The cost of forging these paywords should be more costly than the monetary reward to not give malicious vendors incentive to forge paywords. This risk can be reduced further by including the length of the payword chain in the initial promise.

*MicroMint*[27], the other system designed by Rivest and Shamir, is a debit system. In this scheme a central authority issues digital coins to allow payments each worth the same small amount of money. Such a coin is a bit-string with the properties that they are hard to produce and that their authenticity can be easily checked without contacting the bank. These coins consist of hash function collisions for some function  $h$  mapping  $m$ -bit strings  $x$  to  $n$ -bit strings  $y$ . Two strings of type  $x$  collide when:

$$h(x_1) = h(x_2) = y$$

Agents receive these coins from the bank when they withdraw money. These coins can then be spent at vendors who deposit them at the bank at the end of a period. This insecure payment scheme gives rise to two security issues, double-spending and forgery.

Hash collisions are progressively easier to find when more values of  $x$  have been calculated, which makes small scale forgery costly. Large scale forgery can be detected by introducing specific demands to coins in different time periods. These demands make it more difficult for forgers to make a significant amount of coins before the end of the period. At the beginning of a time-period the bank broadcasts these criteria and keeps several criteria secret. Upon deposit of coins the bank can check the authenticity and trace inauthentic coins back to forgers. Just like coin forgery, large scale double-spending can be detected by tracing these coins back to the offender.

PayWord allows payment of larger amounts by skipping several paywords, however the intermediary paywords need to be calculated as well. Another issue is that PayWord is a credit system. The vendor might be willing to take the risk for small payments, but it becomes more dangerous for large amounts. Micromint does not even support payment of larger amounts of money. Therefore methods should be considered that allow larger payments.

### 3.1.2 Macropayments

To allow spending of larger amounts other schemes have been proposed. The main focus in their design is that third parties should not be able to trace spending of an agent. To this end Chaum[4] developed a payment scheme based on blind signatures. The interactions during the creation of a legitimate digital coin entail that the bank can not trace the newly created coin. The following example illustrates the process of an agent withdrawing virtual cash into a digital coin.

Agent  $a$  wants to withdraw an amount of money in the form of a digital coin. First agent  $a$  generates  $x$ , a secret random identifier. The coin must be signed by the bank with signing function  $s$  to make it authentic. However, if the bank knows the contents of  $x$  it can identify the payee when the coin turns up later. To this end agent  $a$  has a function  $c$  and its inverse  $c'$  with the property  $c'(s'(c(x))) = s'(x)$ . This enables agent  $a$  to create a legitimate digital coin which the bank can not identify.

Unfortunately digital coins can be copied arbitrarily like every other bit of digital data. This effectively allows agents to double-spend coins. Therefore payees must deposit the coin immediately to check whether it has been double-spent or not. Deposit of and payment with digital coins requires communication with a central authority, the bank. This is logical because people store money in these central places, but requiring communication with a bank upon payment is highly inflexible. It would be much easier when agents are able to pay with digital cash offline and deposit all gathered coins at a later moment in one batch.

Restrictive blind signatures developed by Brands[3] allows just this, offline spending of digital coins. This is not the place to go into detail about the cryptographic solution, but in short it allows traceability of double-spenders after the fact. The bank is not able to trace a coin back to the agent it was issued to, unless an agent payed two or more agents with it. A simple analogy for this technique would be coordinates: the agent that creates a coin is known by  $(x, y)$ . The bank is only able to calculate  $x$  after one transaction and when an agent spends that coin again it can calculate  $y$  as well. Upon double-spending the bank can join this information together to identify the culprit and take appropriate countermeasures.

### 3.1.3 Second Life

Second Life is a virtual world developed by Linden Labs<sup>3</sup>. Although it is not a decentralized system it is interesting because it contains a real economy with its own currency, the Linden dollar. The Linden dollar can be traded at the Linden

---

<sup>3</sup> <http://www.secondlife.com/>

money exchange for real currency. Since its inception several third party money exchanges have started set up business in Second Life as well.

Braddy Forrest[12] looked at the size of the transactions made in Second Life. He noticed that 85% were worth less than 1 US dollar and 57% of them were in amounts of less than 0.07 US dollar. Transactions of this small scale are not cost effective when payed by credit card or Paypal. He concludes by saying that the Linden dollar might be an effective currency to be used for general micropayments on the internet. Second life allows for arbitrary large amounts and no additional macropayment scheme would be necessary. It will be interesting to see how this on-line economy will develop.

#### 3.1.4 Fraud

A system based on real money has to be completely exploit-free and safe. Even though transactions are very cheap the cost of a large amount of transactions can add up to a big amount. The massive amount of potential transactions makes micropayment susceptible to a *salami attack*. In a salami attack a small amount of money of each transaction is sliced off like salami and transferred to the account of the fraud. In January 1993 executives from a rental-car company in Florida were charged for defrauding at least 47.000 customers using a salami technique. From 1998 through 1991 they had charged 5 gallons more than the actual gas tank capacity of their vehicles.

A micropayment system is like a black box, the inner workings and how credits are spent is not completely transparent to the user. People will also not understand why they have to pay money for a service while they use a comparable and free system like Bittorrent. When credit-systems based on real money are undesirable one should look towards virtual economies based on fictional credit.

### 3.2 Fictional credit

Fictional payments can be divided in two classes, fungible and non-fungible payments. Payments are fungible when agent pay each other with a reusable virtual currency. These payments increase the amount of credits the payee owns allowing him to spend it elsewhere. Non-fungible credits are however worthless to any agent but the payee. They are merely a proof that the paying agent is willing to do something in return, shunning free-loaders.

#### 3.2.1 Non-fungible

The idea of non-fungible credits is that agents prove to another agent that they are interested in a resource by performing a simple task. Dwork et al.[10] introduced a POW (*Proof Of Work*) scheme to combat e-mail spam. Normal agents send far fewer e-mails than agents that spam other agents. By requesting e-mail senders to solve a puzzle and submit a proof the rate of sending e-mails is bounded by the amount of available processing power of an agent.

*Clientpuzzles*[15], another non-fungible credit scheme, aims to make internet servers more resilient against DoS attacks. By sending massive amounts of requests for service these servers can be brought down. When a server becomes

congested with requests it will require clients to solve a puzzle and present it as a POW. The difference between client puzzles and the previous method is that clients are only required to deliver a POW when the server is congested. It does not matter whether an internet server has a 30% workload or a 90% workload, what is important is that the server is able to handle all requests.

Lastly, depending on the importance of a request and the difficulty of a POW it becomes interesting for agents to parallelize POWs. Obviously an agent would need a multi-processor computer or favorably a network of zombie agents<sup>4</sup>. To this end non-parallelizable puzzles can be used as POW like LCS35[28]<sup>5</sup>.

According to Dingle et al. [8] a differentiation should be made between cumulative and transient congestion. Transient congestion can be thought of as bandwidth congestion as described before. After a request has been handled the bandwidth becomes available again. Cumulative congestion happens for example in Freehaven[7] where agents store resources for each other. Freehaven is a distributed, anonymous and persistent data storage that allows agents to store sensitive information. To allow anonymity agents store parts of documents for each other for an agreed duration. Every agent has a fixed amount of storage and therefore congestion does not fluctuate as much. Therefore payment should be required for all cumulative services and only in the case of transient congestion. Besides using the POW for resource acquisition it can also be used to make it less advantageous for agents to whitewash their history.

A disadvantage to these non-fungible methods is that it is not completely fair in a heterogeneous network. Network equipment usually has far less computing power than desktop computers and such situations non-fungible methods are undesirable.

### 3.2.2 Fungible

Fungible fictional credit is not linked to real money in any way. In decentralized MAS the biggest hurdle is to keep track of credit balances. Three different storage schemes for credits will be discussed. In the first scheme agents are appointed to keep track of credit balances, in the second scheme agents that are interested in the credit balance of an agent are appointed and in the last scheme peers exchange coins which they store themselves.

**3.2.2.1 Designated bankers** In this method each agent has its own set of designated bankers, agents in the same network that have been assigned to keep track of the credit balance.

Karma[34] is a fictional credit systems with designated bankers. It is based on a DHT<sup>6</sup> network to store credit balances. In this system credits are called karma. Every agent has a set of designated bankers that keeps track of its karma balance. These bankers are a weak spot in the system because they could

---

<sup>4</sup> A zombie agent is an agent compromised by a trojan and under control of another agent.

<sup>5</sup> LCS35 is probably but not provably non-parallelizable.

<sup>6</sup> Distributed Hash Table – Every peer in the network has its own identifier corresponding to a hash value. Using the DHT each peer is able to contact each other peer if it knows the corresponding identifier.

collude to jointly alter the credit balance of a client. Therefore an extensive entry algorithm is in place that prevents agents from choosing their designated bankers and clients. Agents have no incentive to keep track of credit balances of others and therefore they could be encouraged to accurately keep track of this balance by paying them for each credit balance mutation they record.

To stop credit-inflation and deflation a mechanism is proposed to reevaluate the value of a single Karma. Unfortunately this mechanism requires a total of  $\mathcal{O}(N^2)$  messages, but they argue that it only needs to be done every few months. It is however still an incredible burden to the system and it is unconvincing whether inflation and deflation are a bad thing in credit systems.

Also, because agents are able to fabricate credit using collusion or a Sybil attack. It is possible for agents to spend more money than they have, thus having a negative credit balance. This can be exploited by introducing a colluding network of which the agents pay one single agent. A solution would be to introduce a logging, which is exactly what is used in the related bankers system.

**3.2.2.2 Related bankers** In ARA[13] a transaction history is stored for every peer. In general only peers that interact with a peer are interested in its credit balance and therefore only these *interested peers* store the transaction history. Besides this history also the credit balance and a list of interested peers deducted from the transaction history is stored. Time is divided in periods and to prevent the transaction history from growing indefinitely only a certain amount of periods is stored. Not only the history is purged but also the credit accumulated or lost during the purged periods and thus credit is volatile. The effect of this is that peers that idle for long times on end have no incentive to earn these credits, as they will be lost before they become useful.

The transaction history of peers is stored in the form of a collection of proofs. After a transaction between peers a proof is signed by both peers. This proof together with a current credit balance is sent to all interested peers of the peers involved. Interested peers initiate audits with a certain probability. Audits can involve the credit balance, a complete transaction history of a certain period or the lists of interested peers. When during an audit the audited peer appears to be lying it will be blacklisted and is prohibited from any future transactions. This is a very harsh penalty and it is unsure who will decide this and how it can be checked whether some of the interested peers are lying or the audited peer.

To prevent peers from mounting a Sybil attack peers can only interact with other peers when they have a positive credit balance higher than a system wide constant LL (lower limit). The problem of this approach is that a deadlock can occur at the initialization of the system when every peer starts with zero credits. To prevent this deadlock peers with a credit balance less than LL also permit transactions with peers that have the same or a higher balance than themselves.

Collusion is somewhat of an issue because a colluding network can donate credits to a single agent and at the same time manage its credit balance. Therefore the credit balance of interested peers should taken under consideration during audits to check whether their credit balance is higher than LL.

**3.2.2.3 I Owe You** An IOU (I Owe You) is a signed paper that proves that someone owes a favor to someone else. Such an IOU is given in return for a favor and can be redeemed in the future to get a favor in return. Systems in

this class use an electronic version of these favors to pay for resources.

In SeAl[22] peers provide a signed favor in return for a resource. These can be created at that peer or it can be a forwarded favor from another peer. Favors are treated like credit in SeAl, peer  $A_2$  can pay peer  $A_3$  with a favor it has received from peer  $A_1$ . The favor  $A_3$  receives is signed by  $A_1$  and  $A_2$  consecutively proving its origin. These favors can be redeemed at the originating agent  $A_1$  for resources.

When the originating peer refuses to redeem the favor a blacklist report can be submitted including a copy of the favor as proof which is stored in a DHT. Peers check their own record every so often to see whether there any blacklist reports are filed about themselves. These can be canceled out if they have a signed favor proving that they did in fact redeem the favor. If for instance  $A_3$  files a blacklist report that  $A_1$  refused to redeem it's favor  $A_1$  can show the favor signed by consecutively  $A_1$ ,  $A_2$  and  $A_3$  proving that it did in fact redeem the favor.

Unfortunately signing these IOUs does not make this system resistant to tampering. Agents can whitewash their history whenever they are blacklisted, and agents can create many worthless IOUs. Users can create IOUs from several sources by initiating a Sybil attack. These can then be paid to a single agent, which can then use these IOUs as payment without getting into debts itself.

Several possible accounting schemes have been described for fungible fictional credits have been described. Unfortunately all systems described are susceptible to attacks where money is fabricated, but using a transaction history log seems the best way to detect colluding networks.

### 3.3 Credit transfer

The problem with credit systems is that there is no way to punish clients for not paying after a transaction. Syverson[32] describes a solution for fair exchanges using weak encryption. Before starting a transaction the requesting party sends a weakly encrypted reward. Decrypting this reward is considered more costly than providing the service. While the transaction is underway the requesting party sends the same reward but even more weakly encrypted. This could be changed to a strategy where only a weakly encrypted and an unencrypted reward is used and a Syverson transaction is done per block.

The problem with this technique is that the cost of decrypting depends on the machines used by the peers. If the peer providing service has a much quicker machine than the requesting peer it might be more worthwhile to decrypt the message than to provide the requested service. Also if the resource that is sent is not the resource that was promised there is no way to regain the credit spent.

### 3.4 Discussion

Although credit systems can be used to counter free-riding behavior, they are not suited to deal with the fake-file problem. This stems from the fact that most people do not use a P2P file-sharing network for sharing files, but for acquiring them. In decentralized storage networks like Free Haven[7]<sup>7</sup> the focus

<sup>7</sup> Freehaven uses a reputation system instead of a credit system.

is on storage and propagation of files. Agents that want to publish a file need to invest in the system by gaining a high reputation or by paying with credits. Letting agents pay for storage of their resources makes it expensive to publish rubbish. In file-sharing networks where agents need incentive to provide files to others it would be illogical and counterproductive to let uploaders pay.

There are several things that stand in the way of adoption of distributed payment systems based on real money. As of now there is no micropayment system in use although there have been several tries[20]. Szabo[33] argues that the mental costs of micropayments are bigger than the actual costs. People will not exactly know how they spend their microcredits and with the sheer amount of small transactions its hard to keep track of their expenditures. For every download they need to think about the potential costs of downloading, especially if there is a price difference for resources from certain peers. Odlyzko[23] argues that people are willing to pay more for flat rates than metered ones. The fact that telephone companies and internet providers are moving from metered rates toward flat fees seems to substantiate this claim.

A fictional credit balance is a mere number and in some systems some systems credits are not even a currency. In these systems credit balances are calculated scores or diminishing over time. Therefore it might be interesting to look at methods that use values for trust without pretending that they represent an amount of credits.

## 4 Trust-based methods

Another incentive for agents to provide services is trust that another agent will do good in the future. People forge friendships with each other which means that they will help each other out and that they will not deliberately harm each other. For example, you need to remove a large tree from your garden and it is too heavy to lift by yourself. Your friends are probably willing to help you carry it out of your garden, because they trust you and they might call upon you later to fix their car. The idea of trust in a MAS is like friendship. Agents that trust each other provide resources to each other because they trust that some day that other agent will repay his debt. Trust can be a very good incentive for agents to act altruistic, but agents must first learn how to start a trustful relationship with each other. According to Stranders[31] trust can be defined as:

**Definition 4.1.** *[Trust] The expectation that one will not be deceived when relying on an entity one does not control completely.*

When an agent enters a MAS it knows no other agent. A trust-based method that provides incentives should have some sort of guideline to start resource exchange with unknown agents. It should also have an algorithm that determines the trust metric based on an interaction history. Lastly an agent needs a decision function to determines whether the trust metric is high enough to warrant another transaction. In this chapter we will first have a look at trust between two pairs and after that at ways to initiate cycles of agents that trust each other enough to start exchanging resources. At last we will see how friendly agents can boost each other.

## 4.1 Pairwise Resource Exchange

Trust is something that involves a pair of agents and only that pair of agents. When agents use trust as a basis for their interactions with other agents a network of agents that favor altruism should form. Pairwise interaction mechanisms that are built on trust between agents are simple and work quite effectively. Two of the more popular P2P file-sharing networks are built around a trust-based method.

### 4.1.1 Scoring

The first notable P2P file exchange network incorporating fair bandwidth sharing is Edonkey2000. Although the original Edonkey2000 client is not used anymore, the network is still in use with the currently most popular client eMule. It is not a completely decentralized system because agent and resource discovery is done via a central server. A DHT to replace the central server in the Edonkey2000 network is already in use, but it is still in testing phase. The program allows users to search for files in the network and when a user decides it wants to download a certain file the underlying agent sends a download request to all agents of which it knows that possess it. These requests are put in a waiting list in which the ordering is based on the waiting time and the amount of data in debt to that agent. This means that an agent should supply resources to agents it is queued up with in order to boost its position in the waiting list.

By providing resources an agent can advance up to 10 times as fast in a queue as an agent that does not provide resources. The places in a waiting list are determined by a calculated value. Simply put, this value is the waiting time times the ratio between uploaded and downloaded resources. Every agent stores these scores for all agents it has interacted with. This system is robust against whitewashing because every new request from an unknown or a known agent will start at the bottom of the queue. This is also the reason why collusion and Sybil attacks are ineffective, because there is no incentive to have multiple clones waiting in the same queue. The system is however very susceptible to malicious injections of fake files.

Even though there is a resource rating system that users can use to post anonymous comments about files it is not sufficient. Every agent can rename a file introducing many false naming variations in the resource discovery system. It is possible to generate a list of all the different naming variations, but it is not possible to determine the original resource name. To this end websites are created that contain links to resources including the original name and verified reviews, but these websites are prone to failure. P2P networks have always contained a lot of copyright infringing resources and owners of websites with links to these copyrighted files have a very real chance of being taken to court by copyright holders.

### 4.1.2 Tit-for-tat

A fairly recent and very popular P2P file exchange system is Bittorrent[6]. It is purely focused at fair downloading of files and therefore resource and agent discovery is done via central servers. However a DHT has been introduced for agent discovery since the introduction of Bittorrent. A unique feature is that a separate network is created for a set of files and therefore each peer connected

to the network is in some way interested in that set of resources. To enter the network a user must download a *.torrent* file that contains information about the set of files and a link to the central tracker. The central tracker keeps a list of all agents connected to the network and when an agent connects to the network the tracker bootstraps that agent by dealing out a list of agents to connect to. From then on agents communicate with each other to barter for resources.

The interaction between agents is based on the tit-for-tat strategy for the repeated prisoner dilemma. In each round both agents have to decide whether they give a resource to the other. Keeping silent in the prisoner dilemma is the same as providing a resource and betraying the other is the same as not providing a resource. When agents do not know each other nor trust each other the most rewarding strategy is to betray. When every agent does this the trend will become that a lot of agents will also betray and resource exchanges will come to a grinding halt. To promote resource exchange tit-for-tat is introduced. In this scheme agents will always choose to give the resource in the first round. In the second round the tit-for-tat player will do what the opponent did in the round before. When a tit-for-tat agent plays against an agent that plays unfair it will not receive a resource in the first round. From then on it will do as the unfair agent did in the previous round and will thus in total it will lose only one resource. When it plays against a tit-for-tat agent however each agent will provide a resource in the first round and in the second round they do as the opponent did, which means they will cooperate with each other indefinitely. This means that an agent will lose at most one resource to unknown agents, but it will never be betrayed by other agents that employ tit-for-tat.

Agents select other agents in the network at random for resource exchange. To allow newcomers to join the network each agent reserves 20% of its upload bandwidth to establish connections with unknown agents. This also allows agents to discover agents that perform better than agents that it is connected to at the moment. However, there exist a modification of the bittorrent client that exploits this part of the protocol to augment download performance. Bittyrant agents use less than 20% of their upload bandwidth to connect newcomers to the network when it downloads at a reasonable rate already. Bitthief is even more malicious, it abuses this altruism towards newcomers by reannouncing that it is a new peers every round. By doing this it takes advantage of tit-for-tat agents that always provide a resource in the first round.

Both implementations of pairwise interaction have problems with newcomers. Tit-for-tat is heavily susceptible to whitewashing when peers are able to change identities because agents are designed to be altruistic to newcomers. Agents in Edonkey2000 are not altruistic to newcomers and it is robust against whitewashing, but at a cost. Newcomers need to wait a very long in the queue or they have to boost their scores significantly by uploading data to possible resource providing peers.

Another problem is that in these kind of systems agents may not be able to find a resource exchange partner. For example, agent *A* wants a resource from agent *B* but not the other way around. Agent *B* has no trust incentive to provide services to *A* and therefore agent *A* is dependent on *B* to be altruistic and to have free resources to spare. It is possible to give agents incentives using only trust mechanisms to start exchanges among several agents at a time using one way cyclic resource exchange.

## 4.2 Cyclic Resource Exchange

Pairwise interaction is not flexible, because often only one agent in a transaction is interested in data from the other. This makes the system dependent on the altruism of peers thereby sustaining freeriders. Therefore cyclic resource exchanges to promote the usage of resource exchange over one way resource transfer.

Anagnostakis et al.[2] propose such a method to start a cycle of transactions between peers. Whenever an agent wants a resource from another agent it sends a request. Every agent keeps track of requests it has received in an Incoming Request Queue (IRQ). Each of these requests is coupled with an IRQ of the agent from which it received the request. From this IRQ a tree can be formed with the agent itself as root. The first layer is formed by the requests in its IRQ. The second layer is formed by the requests inside the requests in its IRQs. Each following layer is contains the trees generated by the IRQs in the preceding layer. To prevent extensive flooding the depth of an IRQ is limited.

Whenever an agent  $\mathcal{A}$  request a file from agent  $\mathcal{B}$  and agent  $\mathcal{B}$  is contained in the IRQ held by  $\mathcal{A}$  a circular transaction can be initiate . A cycle of data exchange can be constructed along the path  $\mathcal{B}$  is connected to  $\mathcal{A}$  in the IRQ by activating all requests along the path and initiating the exchange between  $\mathcal{A}$  and  $\mathcal{B}$ . Each agent in this cycle wants data from the next agent in the cycle and this circular data exchange will stay profitable for all parties until one agent leaves.

Testing proved that cycles of length 3 to 5 were most effective in combination with regular pairwise resource exchanges. The explanation to these test results is that larger cycles are more prone to agents bailing out and are therefore shorter lived than short circles.

## 4.3 Correlated Opinions

Walsh et al. propose Credence[35, 36] which allows agents to calculate a value for trust in unknown agents based on the correlation of opinions. Credence is a system designed to order resource search results in the LimeWire Gnutella client by their quality rating, which is computed from votes issued by other agents. Trust between agents is defined as the correlation of their voting behavior, their opinions. Votes are propagated by a gossip protocol supported by query flooding, but a DHT could be used as well.

Votes consist of a reference to a resource and a claim concerning metadata using set operators. For example, the vote  $\langle \mathcal{R} : \text{madonna} \subseteq \text{name}, \text{mp3} = \text{type} \rangle$  claims that *madonna* is a valid name and *mp3* is the type of resource  $\mathcal{R}$ . The claim  $\langle \mathcal{R} : \text{mp3} \notin \text{type} \rangle$  is in contradiction with the preceding vote because it states that  $\mathcal{R}$  is not of type mp3. Resources that are identified as completely unwanted can be described with the vote  $\langle \mathcal{R} : \text{name} = \emptyset \rangle$  which refutes any other claim concerning the name of  $\mathcal{R}$ .

The trust agent  $\mathcal{A}$  places in votes cast by agent  $\mathcal{B}$  is computed using a slightly modified Phi coefficient as follows. Let  $a$  and  $b$  be the fraction of votes where  $\mathcal{A}$  respectively  $\mathcal{B}$  voted positive and  $p$  be the fraction where both agents agree with both vote having positive intentions. Then  $\theta$  is the coefficient of correlation, taking on values in the range  $[-1, 1]$ . When insufficient voting history for an agent is available agent  $\mathcal{A}$  picks an arbitrarily low value of trust.

$$\theta = \frac{p - ab}{\sqrt{a(1-a)b(1-b)}}$$

Agents discover transitive correlations by building a model of the P2P network. Every agent keeps track of a list of correlation coefficients of known agents. These values are propagated through the network using a gossip protocol. By building a model of the P2P network an agent can discover agents to which it is transitively correlated. It will then exchange votes with these transitively correlated agents to see whether a direct correlation can be found.

The problem with this system is that agents do not have a direct relation with each other and can not reciprocate. The trust an agent places in another agent is not based on interaction between the two agents. An agent can easily fabricate votes or copy votes from other agents. This means that an agent can fabricate a high trust level by copying generally held opinions to advertise fabricated votes. This fabricated high level of trust can then be used to advertise their own resources or discredit other resources.

#### 4.4 Adaptive stranger policies

Feldman et al.[11] propose an adaptive stranger policy. It is impossible to determine whether newly joined agents are actually new and therefore they propose that agents dynamically change their behavior towards unknown agents. When an agent encounters many malicious strangers it will act less trustful towards all strangers. If however most unknown agents turn out to play nice the agent will be more inclined to initially trust strangers. A balance between agents that decide to whitewash and agents that do should arise depending on the parameters of this dynamic trust towards strangers.

#### 4.5 Aspects of Trust

Sabater et al. [29] designed the REGRET reputation systems to consider several aspects of trust to evaluate other agents. Consider for example the hierarchy of trust aspects in figure 4.5<sup>8</sup>. All nodes are aspects of trust that travelers place in travel agencies, but not all travelers value the various aspects of their trip equally important. Agents planning for a holiday travel might value TOURS included in the travel arrangement highly and welcome good GUIDES to the local scenery. On the other hand, agents that plan a business travel usually do not have enough time for TOURS and are therefore mostly interested in the HOTEL aspect.

When an agent asks for the opinions of other agents about a travel agency it needs to know how these other agents came to their conclusion. A holiday traveler only weighs the HOTELS aspect with a rank of 0.2, hardly a factor in its own calculation for trust even though he has had extensive experience with that aspect. A business traveler is mostly interested in the HOTELS aspect, but the combined trust value of the holiday traveler is of no use. Not only is it beneficial for agents to keep track of different aspects of an agent for personal use, it is essential when agents share their opinion about other agents with each other.

<sup>8</sup> The example originates from the REGRET[29] paper.

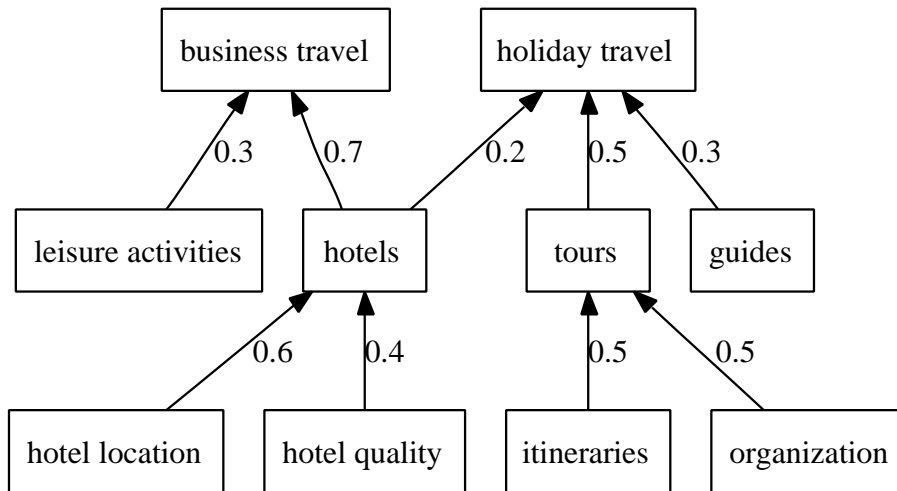


Fig. 1: aspects of trust

#### 4.6 Discussion

Pairwise resource exchange transactions only give good incentives when two peers desire services from each other. Cyclical resource exchange expands the amount possible transactions as well as correlated opinions, but both have a disadvantage. Cyclical resource exchange opportunities are hard to initiate and somewhat more unstable due to amount of peers involved in a single transaction. Correlation of opinions is a rather dubious trust metric, which makes it susceptible to tampering. It could however be improved by adding public/private keys to resources. Votes will then need to be signed by the private key of agents and as a whole by the private key of the resource. This would allow agents to check the authenticity of votes. Malicious peers would need to retrieve a large amount of resources and comment positively on them to promote their own false votes. It should be tested whether the negative effect of these malicious peers is actually larger than their unintended positive effect.

As for the trust systems in use in P2P networks, even though tit-for-tat is limited in the amount of possible transactions it works very well in the current Bittorrent networks. The Edonkey2000 network is still in use, but it is doubtful whether that is due to the pairwise credit system that awards uploading.

It is convenient that these trust based systems only use local data and do not have to worry about untrustworthy global information. There are however negative aspects to solely using local trust information. Agents that have completed all required resources are called seeds in the Bittorrent network. Seeds and other agents that are unable to initiate a resource exchange have no incentive to provide resources to others and the only two options are being idle or to donate resources to which ever agent requests any. When trust information becomes global agents do have incentives to seed resources, because every agent will know about it. Global knowledge about this altruistic behavior will increase the chance that it receives resources in return at a later point from another agent. This does however not work when agents donate resources indis-

criminally and in the next chapter methods will be described that select the right recipient.

## 5 Reputation based methods

When an agent is active in a P2P network other agents will learn about that agent and decide how much to trust it. These experiences of all these other agents is a distributed track record. These values for trust can be combined into a single value, the reputation of an agent. Reputation is defined by the Oxford dictionary as following:

1. the beliefs or opinions that are generally held about someone or something.
2. a widespread belief that someone or something has a particular characteristic.

Reputation thus not only describes trustworthiness of something or somebody, it also describes particular characteristics and opinions. However, to predict the outcome of transactions with unknown agents only the trustworthiness is of importance. Therefore we will define reputation building upon the definition for trust (4.1) as following:

**Definition 5.1.** [*Reputation*] *Trust that one should place in a dynamic entity according to entities one does not control.*

The problem lies in that an agent can not blindly trust the opinions of other agents; without good incentives agents will lie about other agents for various reasons. For example, an agent can discredit competitors to better its own position or an agent can dishonestly recommend its cheating friends.

There is a thin line between trust and reputation systems and this is why the word dynamic was added to the definition. Consider the following two situations:

- Agent  $\mathcal{A}$  decides whether to trust an unknown resource  $r$  depending on the opinions and trustworthiness of agents  $\mathcal{B}$  and  $\mathcal{C}$ .
- Agent  $\mathcal{A}$  decides whether to trust an unknown agent  $\mathcal{R}$  depending on the opinions and trustworthiness of agents  $\mathcal{B}$  and  $\mathcal{C}$ .

These seemingly similar scenes differ in one important point, the quality of a resource is fixed while the quality of an agent is dynamic. A resource does not change over time and therefore neither the trust of rational agents in it. However, an agent can change their strategy towards other agents over time. While the quality of agents is represented by a trustworthiness notion, the quality of a resource is a fact. Therefore the first scenario is a trust system, while the second scenario is a reputation system.

In this section about reputation a method based on majority votes will be discussed. Methods in the subsection after that build upon it by weighing votes from other agents by their trustworthiness. After that blacklisting is discussed to make life harder for malicious peers as an extension to reputation management. After that the fallacies of the transitive usage of trust will be explained and in

the succeeding subsection a solution will be presented by explaining the different aspects of reputation. And finally interaction policies will be discussed that give further incentives to altruistic agents.

## 5.1 Majorities rule

The idea that the majority must be right is quite appealing, because democracy which is based on the same principle is the most widely used government form. XRep is a system based on this principle, it allows agents to collect the public opinion about other agents and resources through polling. XRep is built as a reputation extension to Gnutella and uses a flooding communication model to collect trust information. The general idea is to let every agent vote and base resource and agent selection on the amount of votes received. Before explaining how XRep works Gnutella will be treated in more detail.

Gnutella is a pure P2P network, meaning that the network of peers operate without the need for a central server. All peers connect to a predefined amount of peers and all messages a peer sends are routed through these neighbors. The Gnutella protocol comprises of only 5 message types: PING and PONG facilitate peer discovery, QUERY and QUERYHIT facilitate resource discovery and PUSH messages are used when normal resource retrieval fails. PING and QUERY messages are flooded through the immediate neighbors spreading out to neighbors of neighbors and further; PONG and QUERYHIT messages are sent directly back to to originating agent to indicate agent and resource discovery.

XRep adds the messagetypes POLL, POLLREPLY, TRUEVOTE and TRUEVOTEREPLY and every agent generates a public/private key generation upon entering the network. Peers can ask the opinion of other agents in the network about a resource or an agent. It should be noted that a majority poll about a resource is a trust mechanism and that a majority poll about an agent is a reputation mechanism. Such a poll is initiated by flooding a POLL message into the network. Let  $\mathcal{A}$  be the agent that wants to poll the network. The POLL message includes the public key of  $\mathcal{A}$  and a list of resources and peers. Every peer that receives the POLL messages sends a POLLREPLY back containing the votes it cast encrypted by the public key of  $\mathcal{A}$ . This encryption prevents tampering with votes and preserves the confidentiality of votes. When enough POLLREPLY messages have been received  $\mathcal{A}$  starts to evaluate the votes. Based on the assumption that colluding agents are usually operated from a small subnet of ip addresses, only one vote is considered from each subnet.

From the resulting set of remaining votes  $\mathcal{A}$  randomly selects peers to check the authenticity of their vote.  $\mathcal{A}$  sends a TRUEVOTE message containing their vote to these randomly selected peers, who in turn send a TRUEVOTEREPLY back. The purpose of this exchange of messages is to check whether the vote received was actually sent by that peer. Agents on the internet are able to forge messages with an arbitrary source ip address. When the authenticity of the audited peers has been noted,  $\mathcal{A}$  can select resources and peers according to majority vote. After having interacted with the selected peers and resources it takes note of their quality to be able to reply with better opinionated POLLREPLY messages to other agent.

Even though majority voting is based on the well proved government form democracy it is inadequate for MAS. Collusion and Sybil detection in XRep is solely based on the conviction that colluding agents operate from a subnet, which discriminates against virtuous companies and university dorms operating from a single subnet. XRep is also vulnerable to whitewashing, agents with a bad reputation are able to rejoin the Gnutella network without any repercussions. The biggest disadvantage of majority voting is that very well opinionated and trustworthy agents have the same voting power as ignorant agents. Several ways of weighing opinions will be looked at in the next subsection.

## 5.2 Weighted opinions

The idea behind the methods in this subsection is that trust is transitive. This means that when an agent  $\mathcal{A}$  completely trusts agent  $\mathcal{B}$ , and agent  $\mathcal{B}$  completely trusts agent  $\mathcal{C}$ , agent  $\mathcal{A}$  can completely trust agent  $\mathcal{C}$ . There are however different methods to determine the reputation of  $\mathcal{C}$  from the perspective of  $\mathcal{A}$ . First a method based on probability theory is discussed where values of trust are multiplied to calculate a value for reputation. After that a method based on the maxflow algorithm is discussed.

### 5.2.1 EigenTrust

EigenTrust[17] is a method designed to determine which peers inject bad files into the network. In EigenTrust agents calculate a value for trust according to the ratio of satisfactory and unsatisfactory transactions with other agents. It can however be used for any other notion of trust and reputation. The value  $c_{ij}$  is the amount of trust agent  $i$  places in agent  $j$ . These values are then normalized to make sure that malicious agents can not assign arbitrarily high or low values to skew the results. These values,  $c_{jk}$ , are then propagated and weighted by value of trust,  $c_{ij}$ , the receiving agent  $i$  places in the agent that provided it, agent  $j$ .

$$t_{ik} = \sum_j c_{ij} c_{jk}$$

The resulting value  $t_{ik}$  is the reputation of agent  $k$  in the perception of agent  $i$  based on the opinion of others weighted by his opinion of those others. This first value is actually the result of a majority vote method scaled by the trustworthiness of the voters. This step can also be described in matrix notation when  $C$  is the matrix  $[c_{ij}]$ .

$$\vec{t}_i = (C^T)^2 \vec{c}_i$$

The vector  $\vec{t}_i$  contains values of trust for all agents based on the weighted opinions of everyone about each agent weighted by the trust agent  $i$  places in them. This aggregation step can be done multiple times producing the following vector for  $n$  steps.

$$\vec{t} = (C^T)^n \vec{c}_i$$

When this step is done often enough the vector  $\vec{t}$  will converge to the same vector for every agent, the left principal eigenvector of  $C$ . This final vector  $\vec{t}$

contains the global values of trust, the reputation of each peer. This method is however susceptible to collusion and therefore pre-trusted agent are introduced. This results in a web of trust around these pre-trusted agents excluding colluding agents, because those agents are not trusted by any agent transitively trusted by pre-trusted agents.

In the actual implementation of EigenTrust each agent is assigned to several agents as a score manager. As a score manager an agent will take care of calculating the reputation for other agents. Score managers are necessary because convergence of reputation values is dependent on where the algorithm is started. When agent  $\mathcal{A}$  is connected through a long string with agent  $\mathcal{B}$  convergence will take many steps. Another reason is that agents are capable to commit fraud when they administer their own reputation. To further prevent score managers from cheating it is crucial that they are not able to pick the agents they audit.

A score manager first requests the local values of trust of the agent it audits, the daughter, and stores these in the vector  $t_d^0$ . It will then annotate which agents have trust information about its daughter in a vector  $B_d$  and which agents the daughter has trust information about in  $A_d$ . Every agent knows or receives the vector  $p_d$  which contains trust values pertaining to the pre-trusted agents in the network and a scalar  $a$  that indicates the influence of the pre-trusted agents.

The algorithm works in rounds indicated by  $k$ . In each round every score manager of agent  $i$  forwards  $c_{id}t_i^{k-1}$  to the score managers of the agents in  $B_i$  and thus each score manager for agent  $d$  will receive  $c_{id}t_i^{k-1}$  from the score managers of the agents in  $A_d$ .

$$t_d^{(k+1)} = (1 - a)(c_{1d}t_1^{(k)} + c_{2d}t_2^{(k)} + \dots + c_{nd}t_n^{(k)}) + ap_d$$

The resulting vector  $t_d^{(k+1)}$  is then used in the next round until the result has converged far enough.

$$\left| t_d^{(k+1)} - t_d^{(k)} \right| < \epsilon$$

Malfunctioning agents will never get a high reputation value, because already in the first round these agents will not receive any reputation from other agents. The pre-trusted agents make sure that all agents are connected to each other through a trust path containing these pre-trusted agents. Colluding agents do not provide resources and are assigned very low trust values by agents not in the colluding swarm. In the end their reputation will converge to 0. EigenTrust is however vulnerable to agents that provide good service and collude. This creates a trust path from well performing agents to agents in a colluding network, boosting the ratings of all the agents within it. Also, EigenTrust does not provide protection against agents that whitewash their history and return as a new user.

### 5.2.2 Maxflow

Feldman et al[11] introduced a reciprocative decision function to calculate values of trust. This reciprocative decision function depends on some definitions. The *generosity* of agent  $j$  is calculated according to what  $j$  has provided ( $p_j$ ) divided by what it has consumed ( $c_j$ ). Generosity can therefore not be applied in systems where trust is not based on resource consumption.

$$g(j) = p_j/c_j$$

The outcome of this generosity function is then used to calculate the *normalized generosity* of agents. The generosity of each agent is divided to the generosity of the agent itself.

$$ng_j(i) = g(i)/g(j)$$

When the normalized generosity for an agent is higher than 1 that agent is more altruistic and has been more generous. This means that peers will consider interaction with lowly reputed agents when it is itself lowly reputed. Policies for server selection will be looked at in more detail in 5.6. To make this method more scalable these values for generosity can be made available to other agents by storing them in a DHT. Without precautions shared history is however vulnerable to collusion and other malpractice.

Therefore the idea of normalizing generosity is applied to a distributed and collusion proof method based on the maxflow algorithm. Agents keep track of the amount of successful services provided by other agents. It is assumed that an agent can trust the opinions of agent that have provided many services in the past and this trust is transitive. The maxflow algorithm finds the path with the largest capacity in a flow network. Agents are nodes in the flow network and the capacity of edges are formed by the amount of service agents have received from other agents. Unfortunately the maxflow algorithm has  $\mathcal{O}(V^3)$  complexity and therefore a heuristic is used that runs in constant time at the cost of not always finding a flow when it does exist.

$$c_{ij} = \min\left(\frac{\text{maxflow}(j \text{ to } i)}{\text{maxflow}(i \text{ to } j)}, 1\right)$$

When a reputation value is 1 more resources have been provided towards the inquiring agent than the other way around. Colluding networks that do not provide services are therefore never chosen for the maxflow path because they choose not to provide services. The usage of maxflow has however the same weakness as EigenTrust: agents that perform good, but collude as well.

### 5.3 Blacklisting

Most reputation methods described earlier are primarily focused on finding correct agents and resources. While this hampers the performance of malicious agents in attaining their goal it might be more effective to directly punishing them. Peers that act malicious because they perform better that way, blacklisting them may give them more incentive to behave correctly.

Papaioannou et al. propose a blacklist extension[24] to the EigenTrust system. During the time an agent is blacklisted it will be denied transactions with other agents. In EigenTrust the reputation of agents is administered by several review agents and after each transaction both agents submit a report to the review agents of both. When these review agents receive a negative report from either agent both agents are blacklisted, which seems strange but is necessary

nonetheless. When one agent reports that the other misbehaved neither can be trusted and thus both must be punished. Agents that behave correctly will be punished two times by being blacklisted after a failed transaction and these altruistic agents thus need incentives to step up.

To give altruistic peers an incentive to report bad peers a dynamic blacklist duration is introduced. Each agent is assigned a non-credibility value upon entering the network,  $ncr_0$ . The duration an agent is blacklisted is  $b^{ncr}$  with a base  $b > 1$ , which means the lower  $ncr$  the shorter the blacklist duration. This value  $ncr$  is decreased by  $y$  when an agent gets a positive review and increased by  $x$  every time an agent gets a negative review, where  $0 < y < x$  and  $0 < ncr$ . Agents that perform bad are able to whitewash their non-credibility value by rejoining the network, but good performing agents will always have a lower non-credibility value than  $ncr_0$ .

A punishment scheme for review agents is also introduced, because otherwise review agents would form a weak point in the system. First, when review agents disagree the opinion of the majority will be chosen. Second, agents audit their review agents and report misbehaving agents to their respective review agents. When reviewing agents are caught lying they will themselves be punished by blacklisting.

The system is however very susceptible to whitewashing. Malicious build up a high non-credibility value and a low reputation rather quickly. This makes it tempting for a selfish agent to rejoin the network each time it is blacklisted. A possible solution would be to introduce a compulsory blacklisting upon signing into the network.

## 5.4 Transitive Trust

Agents that perform well usually have good opinions about other agents and advise other agents truthfully. Therefore the trust agents place according to resource sharing behavior is correlated to the trust it should place in its opinions. Correlation does however not imply that these aspects of trust are the same. The problem of transitive usage of trust is that trusted agents do not necessarily tell the truth about everything, which can open the door to collusion networks. All methods treated in this subsection base the trust agents place in other agents on the freeriding behavior, which means this value of trust is not descriptive for their trustworthiness concerning referrals to other agents. Trust in freeriding behavior can thus not be used transitively. Jøsang et al.[14] explain that the length of the formulation of trust grows proportionally with the length of transitive trust path. The definition in the last leg in a trust path is the trust that the last node is trustworthy about something. All the preceding legs are defined by the trust that the succeeding node is a trustworthy referrer included by the definition of the succeeding leg. All legs but the last one are thus definable by a recursive structure and therefore Jøsang proposes to simplify things by only differentiating between direct and indirect trust.

In practice mixing two aspects of trusts makes P2P file sharing networks vulnerable to collusion attacks with spies. A colluding swarm of agents contains badly performing agents and well performing agents. Well performing agents, spies, try to build up reputation by perform as good as they can. In the mean time they favor colluding agents over normal agents in their reputation reports. These colluding agents in turn give misinformation and when trust in them has

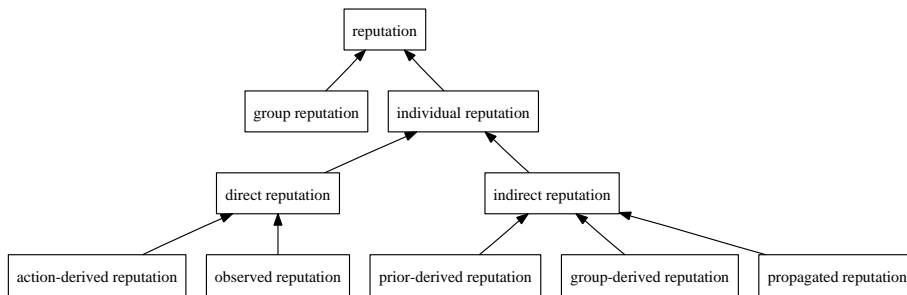


Fig. 2: ontology of reputation

dropped too low they rejoin the network, thereby whitewashing their identity.

## 5.5 Ontology of Reputation

Mui et al.[21] propose an ontology reputation describing various sources of reputation information. This ontology is shown in figure 5.5. Trust methods described in chapter 4 fall under the action-derived reputation aspect. Reputation systems described in this chapter are based on the propagated reputation aspect. Prior-derived reputation is a preconception agents have about other agents based on features, the adaptive stranger method falls in this category.

This leaves observed reputation, group-derived reputation and group reputation. Observed reputation information is intrinsically hard to obtain in a P2P network. It is almost impossible for agents to observe transactions between other peers and determine which agent is at fault in case of a failed transaction. A possible strategy could be a digital escrow service which has not been looked into. Group reputation and group derived reputation extend the trust horizon of agents and can be deduced from social ties.

### 5.5.1 Social dimension

As discussed previously agent can infer reputation information from group relations between agents. An agent can obtain reputation information from its group members and an agent can generalize about an opponent because of the group it is in. Agent are judged according to the group they are in and their judgment is biased by their group members. Agents need methods to determine which group to join.

Ravichandran and Yoon[26] propose EigenGroupTrust, a method that lets agents join in groups led by a leader. This leader observes all transactions undertaken by its members and uses this information to keep track of the trustworthiness of members. When members behave bad they are kicked out. This makes sure the group reputation remains high. Another perk of joining such a group is that the leader helps the members by keeping track of reputation of other groups and agents. As always the greatest point of weakness is the central authority. When the leader is corrupt it is difficult for group-members to notice, because the leader might employ

## 5.6 Policies

It is tempting for agents to contact the most reputable agent offering a specific resource. This will however hurt network performance for these most reputable agents, as they are flooded by other agents requesting resources. Agents that act trustworthy are thus essentially punished for performing good.

Papaioannou et al.[25] have looked at several policies for *provider selection* and *contention resolution*. When the same resource is provided by several agents an agent uses provider resolution to pick the agent from which it will request the resource. When multiple agents are in a service request queue for an agent contention resolution will be used to figure out in which order agents are served.

For provider selection three policies are introduced. The policy *highest reputation* selects the agent that has the highest reputation. The policy *comparable reputation* selects agents that have a comparable reputation in a predefined range. And finally the *blacklisting* policy expels all badly performing agents from selection.

For contention resolution two policies are introduced. The policy *highest reputation* again selects the requesting agent with the highest reputation. And the policy *probabilistically fair* selects requesting agents with a chance based on their reputation. Let  $R_i$  be the reputation of agent  $i$  and  $j$  are the agents that request a service, then  $R_i / \sum_j R_j$  is the chance that agent  $i$  will be selected.

All combinations of these policies have been tested. The focal point of the research was the success rate of requested services for altruistic peers. Both provider selection and contention resolution have a large impact on this success rate. Comparable reputation provider selection policy performed best when many altruistic peers are available. The reason for this is that success rate among started services is most important, selecting the most reputable agents will overloaded those agents thereby failing many requests. When however only a small number of peers are altruistic the highest reputation policy performs better. When altruistic agents are scarce finding a resource is most important regardless of the reputation of the provider.

## 5.7 Discussion

It is very hard to determine the exact qualities a reputation systems in a P2P file-sharing network should have without testing these models. It is clear however that the reputation systems described in this section are not good enough; they all suffer from the same transitive trust pitfall. Therefore it is clear that a newly designed reputation systems should employ multiple aspects of trust to determine reputation. To enlarge the set of entities an agent can form an opinion about a multitude of sources should be consulted. A promising new source of reputation information is social grouping. Depending on the effectiveness of a reputation system the top reputed agents may become overloaded in which case a comparable reputation policy should be used for provider resolution.

## 6 Discussion and Conclusion

The problems that need to be solved are free-riding and fake file injection in Tribler. These two problems need different approaches and therefore they will be discussed separately.

## 6.1 Free-riding

Free-riding is the behavior where agents choose to upload as little as possible. Due to inherent anonymity on the internet it is difficult to effectively punish wrongdoers. Therefore incentives are needed to give agents a reason to donate resources to other agents.

In Tribler the trust system tit-for-tat is employed for resource exchanges, data and torrents. Normal bittorrent swarms are focused on a specific set of resources, but Tribler may be going in the direction where agents remain active in many bittorrent swarms. Tit-for-tat only works well when agents exchange resources with each at the same time. When agents serve a multitude of resources chances diminish that agents need resources from each other. This scarcity of possible resource exchanges will lead to more seeding and therefore Tribler should move to an incentive method other than trust based.

As a precursor to a full-fledged credit or reputation system the Bartercast protocol has been introduced in the latest version of Tribler. Each agent accounts how much it has uploaded to and downloaded from other agents. This transaction history is exchanged with other agents via an epidemic protocol. Agents will then be able to infer a credit rating or reputation value for other agents based on this shared transaction history. At the time of writing there are however no checks in place to resist cheating, but the main purpose is to retrieve a data set for research.

Using a reputation system to give agents incentive to upload is quite fuzzy. The properties of a high reputed agent can be chosen as a combination of the ratio of uploaded and downloaded resources, the scarcity of hosted resources and the amount of time spent online. Using reputation as a measurement is possible, but a credit based system seems more logical.

Far more precise than a reputation system is a market based on fungible virtual credits. An advantage of such an approach is that the value of services can be determined automatically by the market when agents are allowed to barter over the price of resources. Also, credits can be used to trade various items without the need for adding an extra aspect of trust or reputation. A credit system is however very vulnerable to malicious creation of credit by use of collusion networks. Lian et al.[18] looked at collusion behavior in a Chinese P2P network and proved that collusion was actively used.

## 6.2 Fake files

Another nuisance in P2P networks is the abundance of fake files. Agents that introduce fake files are driven by goals outside of the scope of the network, which makes it impossible to create incentives for agents not to introduce fake files. Therefore incentives need to be introduced for agents to provide other agents with good information about resources. Tribler has no moderation system in place yet and therefore all incentive methods can be taken under consideration.

Trust systems allow agents to differentiate agents with good opinions from agents with bad opinions. The availability of dependable resource opinions for an agent depends on three factors: the number of resources an agent has knowledge about, the number of agents that gave their opinion about these resources and the number of agents within the horizon of the observing agent. Due to the

enormous amount of resources on file-sharing networks it may become very hard to find trustworthy and knowledgeable agents. Therefore reputation should be used, because it allows agents to take far more opinions under consideration.

When a credit system is in place as well further incentive can be given by a side-payment scheme[16]. Agents that provide good feedback are payed with credits which for example can be used to purchase resources. Other combinations of incentivitation schemes can be made to encourage other wanted behavior from agents. The advantage of linking these incentives is that goodwill obtained by agents can be spent on more services. This flexibility gives agents more incentive to act on their best behavior in every way.

## References

- [1] E. Adar and B. Huberman. Free riding on gnutella. Technical report, Xerox PARC, 2000.
- [2] K. Anagnostakis and M. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *ICDCS '04*, 2004.
- [3] Stefan Brands. Untraceable off-line cash in wallet with observers. In *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 302–318, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
- [4] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer-Verlag, 82.
- [5] Nicolas Christin, Andreas S. Weigend, and John Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*, pages 68–77, New York, NY, USA, 2005. ACM Press.
- [6] Bram Cohen. Incentives build robustness in bittorrent, 2003.
- [7] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: distributed anonymous storage service. In *International workshop on Designing privacy enhancing technologies*, pages 67–95, New York, NY, USA, 2001. Springer-Verlag New York, Inc.
- [8] Roger Dingledine, Michael J. Freedman, and David Molnar. *Peer-To-Peer: Harnessing the power of disruptive technologies*, chapter 16. Accountability, pages 171–213. O'Reilly, 2001.
- [9] John R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.
- [10] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147, London, UK, 1993. Springer-Verlag.

- 
- [11] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 2004. ACM Press.
- [12] Braddy Forrest. Lindens as micropayments. [http://radar.oreilly.com/archives/2006/05/lindens\\_as\\_micropayments.html](http://radar.oreilly.com/archives/2006/05/lindens_as_micropayments.html).
- [13] MyungJoo Ham and Gul Agha. Ara: A robust audit to prevent free-riding in p2p networks. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 125–132, Washington, DC, USA, 2005. IEEE Computer Society.
- [14] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing Topologies of Transitive Trust. In Theo Dimitrakos and Fabio Martinelli, editors, *Proceedings of the First International Workshop on Formal Aspects in Security & Trust (FAST2003)*, pages 9–22, Pisa, Italy, Sep 2003.
- [15] Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Networks and Distributed Security Systems*, pages 151–165, 1999.
- [16] Radu Jurca and Boi Faltings. Using chi-scores to reward honest feedback from repeated interactions. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1233–1240, New York, NY, USA, may 2006. ACM Press.
- [17] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM Press.
- [18] Qiao Lian, Zheng Zhang, Mao Yang, Ben Y. Zhao, Yafei Dai, and Xiaoming Li. An empirical study of collusion behavior in the maze p2p file-sharing system. Technical report, Microsoft Research, 2 2006.
- [19] Jian Liang, Rakesh Kumar, Yongjian Xi, and Keith W. Ross. Pollution in p2p file sharing systems. In *IEEE Infocom, Miami, FL, USA, March 2005*, 2005.
- [20] Jim McCoy and Doug Barnes. Mojo nation. <http://mnetproject.org/>.
- [21] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 280–287, New York, NY, USA, 2002. ACM Press.
- [22] Nikos Ntarmos and Peter Triantafillou. Seal: Managing accesses and data in peer-to-peer sharing networks. In *P2P '04*, pages 116–123, Washington, DC, USA, 2004. IEEE Computer Society.
- [23] A. Odlyzko. The case against micropayments. In *Financial Cryptography: 7th International Conference, FC 2003*, 2003.

- 
- [24] Thanasis G. Papaioannou and George D. Stamoulis. An incentives' mechanism promoting truthful feedback in peer-to-peer systems. In *CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 1*, pages 275–283, Washington, DC, USA, 2005. IEEE Computer Society.
- [25] Thanasis G. Papaioannou and George D. Stamoulis. Reputation-based policies that provide the right incentives in peer-to-peer environments. *Comput. Networks*, 50(4):563–578, 2005.
- [26] Ajay Ravichandran and Jongpil Yoon. Trust management with delegation in grouped peer-to-peer communities. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 71–80, New York, NY, USA, 2006. ACM Press.
- [27] Ronald L. Rivest and Adi Shamir. Payword and micromint: Two simple micropayment schemes. In *Proceedings of the International Workshop on Security Protocols*, pages 69–87, London, UK, 1997. Springer-Verlag.
- [28] Ronald L. Rivest, Adi Shamir, and David Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, 1996.
- [29] Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 194–195, New York, NY, USA, 2001. ACM Press.
- [30] S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking*, 2002.
- [31] Ruben Stranders. Trust models in multi-agent systems. Master's thesis, TU Delft, 2004.
- [32] Syverson. Weakly secret bit commitment: Applications to lotteries and fair exchange. In *PCSFW: Proceedings of The 11th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1998.
- [33] N. Szabo. Micropayments and mental transaction costs. In *2nd Berlin Internet Economics Workshop*, may 1999.
- [34] Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gün Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.
- [35] Kevin Walsh and Emin Gün Sirer. Fighting peer-to-peer spam and decoys with object reputation. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 138–143, New York, NY, USA, 2005. ACM Press.
- [36] Kevin Walsh and Emin Gün Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*, pages 1–1, Berkeley, CA, USA, 2006. USENIX Association.