# Collusion in Peer-to-Peer Systems[☆]

Gianluca Ciccarelli[a], Renato Lo Cigno[a,*]

[a] *Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, 14 Via Sommarive, Povo (TN), Italy*

**Abstract**

P2P systems are used to provide distributed storage, file sharing, video streaming, distributed gaming, and other applications based on the collaboration of participating peers and on the observation that sharing resources used sporadically leads to huge savings. The operation of a P2P system as well as its sheer survival, however, is open to many kinds of attacks, which are tough to fight due to both the decentralized nature of P2P applications, and the lack, in some of them, of a central authority, or of a well-defined structure, or both.

Particularly, as P2P systems require the active collaboration of the participants beyond their selfish interest. Many system include methods designed to lure the most resourceful users into broader participation, to provide an overall better service. The methods devised to attract the contribution of users are unfortunately vulnerable to a particular class of attacks: Collusion. Collusion is broadly defined as any malicious coordinated behavior of a group of users aimed at gaining undeserved benefits of at damaging (some) well behaved users.

In this paper, we survey the literature on P2P systems security with specific attention to collusion, to find out how they resist to such attacks and what solutions can be used, e.g, game theory, to further counter this problem and give P2P systems the possibility of developing into full fledged services of the future Internet.

*Keywords:* Peer-to-Peer networks security, Collusion

## 1. Introduction

P2P systems are used in a variety of contexts, from file-sharing applications, to content streaming, to distributed storage. Such systems are based on the collaboration of all participants to provide an acceptable quality of service both to the wealthy and the less resourceful of them.

[☆]This work is supported by the European Commission through the NAPA-WINE Project (Network-Aware P2P-TV Application over Wise Network), ICT Call 1 FP7-ICT-2007-1, 1.5 Networked Media, grant No. 214412: http://www.napa-wine.eu

[*]Corresponding author

*Email addresses:* `ciccarelli@disi.unitn.it` (Gianluca Ciccarelli), `locigno@disi.unitn.it` (Renato Lo Cigno)

Many of these systems try to encourage the cooperation of selfish peers, that tend to participate to the application only to receive benefit. This is usually achieved using incentives, forms of payments, or some other forms of gratification and punishment. Such gratifications, as well as the punishments, are often based on the logging of the level of contribution of a peer. The contribution, however, can sometimes be hidden or be presented differently from how it really happens, as single users or groups of them collaborate to pollute the auditing system. For example, two collusive peers can falsely claim to provide and receive service from each other without actually doing it, accumulating trust, or reputation, and acquiring positions of privilege in the system.

In general, we can expect in P2P systems the same behavior we observe in real life: as soon as there is anonymous coverage that shields users from ethical eviction or marginalization, behaviors tend to deviate, and as soon as there is a large potential gain in cooperative misbehavior, then collusion appears and flourish until the system collapses under its burden.

In this survey we want to emphasize the importance of the problem of collusion, broadly defined as any cooperative action inside the system aiming at any malicious result or undeserved gain, and attract the interest of the scientific community by stressing the security implications of it. Starting from the recent literature, we provide a first classification of the P2P systems according to the method they use to lure every peer to contribute and possibly the wealthiest of them to contribute more than others.

The gratification/punishment system, in itself, is based usually on a phase of contribution and another of benefit. According to the existence or not of an intermediate phase between the contribution provided to the system and the benefit received from it. A *direct mapping* is observed in systems where there is no accumulation of credit, and peers receive an amount of resources exactly equal to the amount they provide from the same peer; *indirect mapping* instead implies an accumulation of an abstract entity (money, reputation, ...) and a later expenditure of it[1]. In some systems, some level of debt and credit among peers are also tolerated, making the system more resilient.

Eventually, we highlight the importance of game theory as a discipline that lies on top of both approaches, and can be used to describe both, and, what is more important, to conjugate the strengths of both philosophies to deliver better solutions.

The survey is organized as follows. Sect. 2 analyzes how incentive systems are in general vulnerable to collaborative attacks. Sect. 3 provides a classification of P2P systems based on their different application domains and analyzing their different vulnerabilities to collusion. Sect. 4 describes micro-payment systems, how they differ from standard incentives and the reasons why they can be useful to counter collusion. Sect. 5 discusses the tools offered by Game Theory in the perspective of a collusive scenario, analyzing their strengths, but also the limits that standard Game Theory enconters in face of collusion. Sect. 6 closes the paper with a discussion of the survey findings and of possible future research on this topic.

[1]We believe that it is not a matter of time, as suggested by Chu, Chang and Zhang [1]; it is rather a matter of existence or not of some form of intermediation between the contribution and the benefit.

## 2. Incentive Systems: Types and Vulnerabilities

Incentive systems are a means to encourage resource holders to spend their spare resources for the benefit of a community of peers. Incentives can be direct, as in a tit-for-tat scheme, where each unit of resource is reciprocated with a unit of resource; or indirect, where the resources given to the community are reciprocated with some currency or credit that can be used and accumulated to acquire other resources or, in some cases, even cashed at a broker entity (micropayment systems).

### *2.1. Attacks against Tit-for-Tat and direct mechanisms*

A direct incentive system is based on the immediate reciprocation of a unit of resource provided with a unit of resource received. They are also called tit-for-tat schemes, and they are historically related to P2P systems: BitTorrent has been described as using such a mechanism, and many other systems take inspiration from its simplicity.

In a tit-for-tat (TFT) scheme, the resource is divided into units: in a file-sharing system, the unit can be a chunk of the file; in a grid application, it can be a time-unit on a processor. When playing TFT, a peer $x$ provides one unit of resource to any peer $y$ encountered for the first time (time $t_0$); at time $t > t_0$, $x$ reciprocates, that is, it provides the resource or denies it, according to what the $y$ did previously (between $t_0$ and $t$). Let's assume that there exist a way, that we call *Authority*, which allows each peer to decide whether the resource unit received is good or bad: going back to the example of file sharing, we are assuming that a peer can decide whether it received a corrupted piece of data (e.g., by checking a digest).

Suppose now that a coalition of peers decides to attack a victim peer $v$. The coalition can:

- Provide lower-quality resources than what they can actually offer (announce less chunks, a slower processor, ...). Providing lower-quality resources forces the victim to not only accept them, but also to reciprocate (maybe honestly). The advantage for colluders is that they get reciprocation with less resource expenditure. A notorious version of this kind of attack is known as Eclipse attack [2].

- Provide bad resources (e.g., corrupted files). This can last only one round, but if colluders mount an Eclipse attack they are actually denying the service to the victim for longer.

- Attack the Authority. This kind of attack can be classified as a Denial of Service, and is more effective when the Authority is centralized.

Unlike pure TFT schemes, the approach used by BitTorrent, however, is better described as an auction, as done by Levin *et al.* [3]. The key point is that the unchoking algorithm is such that the $n$ best contributors to peer $x$, regardless of how much bandwidth they contributed, are reciprocated with $1/n$ of $x$'s bandwidth. For the coalition to be effective, it suffices to offer the minimum such amount of resources in order to get fully reciprocated.

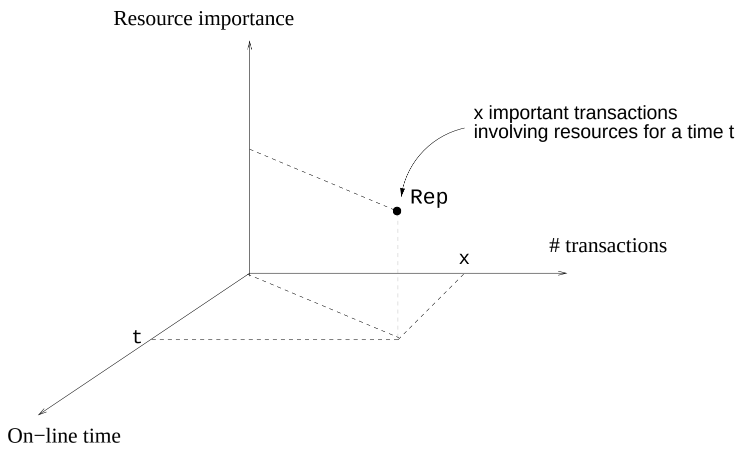


Figure 1: Space used to evaluate the reputation of a peer based on its history.

### *2.2. Attacks against Reputation Systems*

Reputation management represents a different perspective on the concept of incentive. Marti and Garcia-Molina [4], and Despotovic and Aberer [5] provide general overviews about the subject. Reputation systems are especially used against selfish peers, while malicious and Byzantine peers[2] are often neglected.

#### *2.2.1. Design: Requirements and Architecture*

Whatever the adversary, a reputation system should comply with a number of sometimes contrasting requirements that translate then into a system architecture. A basic set of requirements is the following:

**R1** Stability: resilience against churn (peers connecting and disconnecting from the network) exemplifies this problem, because high percentages of churning might disrupt the service and the quality of user's experience;

**R2** Persistent and secure storage of the user behaviour: in fact, one of the main targets of an adversary is the removal of the proofs of his misbehaviour;

**R3** Admission policies to the resources (who can access what);

**R4** Anonymity to preserve privacy, which however contrasts with the previous two requirements.

For a system to manage reputations and resources, a control entity is required, be it central or distributed. Gupta, Judge, and Ammar [6] provide an example of centralized authority; Marti and Garcia-Molina, an example of a decentralized one [7].

[2]The main difference between Byzantine and malicious peers is that the first behave randomly, misbehaving, but not necessarily following a pattern to maximize their benefit or to disrupt the system, while the malicious choose actions according to a goal that is either detrimental to the system or achieves benefits (or both).

Let's now focus on the way to organize the architecture. The main components of a reputation architecture are:

**A1** The way to gather reliable information about peers and their interaction (information gathering sub-system);

**A2** The ranking sub-system, that is, how we use the information gathered;

**A3** The system's response mechanism to a misbehaviour (punishment sub-system).

The information about a peer should link his identity to his behaviour. A peer (the seeker) can gather information about another peer's behavior, starting from peers that he had direct interaction with. Directly interacting peers can be (more) easily recognized as good or bad according to their history (as is the case, for example, in PeerTrust [8]). A general assumption is that, by aggregating more opinions, the accuracy of information about a peer is likely to become higher. The usage of a history creates issues when designing a stranger or newcomer policy, i.e., a way to assign reputation to peers that for the first time join the system and thus have no previous interactions.

The number of interactions each peer needs to aggregate for performing the estimation with tolerable error is in the order of 20-30 (*tolerable* means that the absolute mean error between the prediction and the actual value is around 0.3); higher volumes of feedback provide but slightly better predictions (see Despotovic and Aberer [5]).

The ranking system establishes which behaviour influences the reputation. The reputation is usually built after transactions. Transactions themselves may have different importance: Fig. 1 shows how reputation is built based on transactions and on-line time.

For example, in the SeAL system (see [9]), peers interact through transactions in order to build up reputation. Both peers in a transaction stores a Transaction Receipt (TR) in the form $TR = (client.ID||server.ID, r.ID, timestamp)$, giving a quantitative information about how important a service is (by identifying the resource provided through its ID, $r.ID$).

The reputation can be divided in smaller units describing different aspects of the interaction: while some solutions use a single scalar value to rank a peer, some others, like TRELLIS [10], use arrays of values to separately mark different *sub-reputations* of a peer. As another example, Gupta, Judge, and Ammar [6] differentiate between the behavior of a user forwarding lookup queries or a data stream: the first operation takes fractions of second, while the second requires a longer continuous on-line presence.

Reputation may be assigned using different ranges. This creates the problem of how to judge about a given reputation value: for example, a system designer may choose to use values between 0 and 1 to identify bad to good transactions, and a peer may accumulate a reputation of 500. How can we know if 500 is a good or a bad reputation, if we do not have a comparison threshold? PeerTrust [8] introduces a novel element, the Community Factor, that is the weight assigned to the trust value by the peers in the same community and determined according to internal conventions.

### *2.2.2. Micropayment systems*

Micropayment systems (MPSs) are indirect incentive systems where virtual or real currency provide the level of indirection between the contribution of a service and

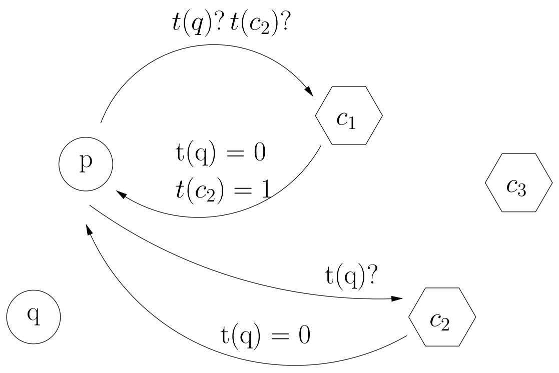


Figure 2: Simple collusion. Peer $p$ gets a wrong trust value for the honest peer $q$. The function $t(x)$ expresses the trust that a peer has in the peer $x$. $c_i$, $i = 1, 2, 3$, are colluders.

the request of a similar contribution from another peer. The architecture of an MPS comprises a *Broker*, which issues the currency and certifies its value.

MPSs are not open to the problem of collusion whenever the single peer can issue its own currency and there is a way to measure the credit diversity of coins coming from other peers, as suggested by Tran, Li, and Subramanian in [11].

### *2.2.3. Collusion in reputation systems*

Collusion can threat any element of a reputation system. We describe the problem, see how it applies to the sub-systems described in Sec. 2.2.1, and then try to present some of the solutions that partially solve the problem.

*Collusion and information gathering system.* The part of the system most vulnerable to collusion is the information gathering sub-system. The basic forms of collusion we can find in this case are slandering and promotion.

A coalition can slander an honest user by always poorly voting the transactions they have with him. Slandering has a natural opposite, malicious promotion, that arises whenever a coalition votes to rise a reputation of an ally of them without him being involved in any (good) transaction.

Slandering and promotion are not the only ways peers can collude and cause harm. Assume the population of the peers is divided in two groups: the honest peers, who always report honestly, and the liars, who misbehave in different ways. Let $p_i$ be the probability for peer $i$ to behave trust-worthily: scenarios analyzed in the literature (for example [5, 12]) include the following:

**Simple collusion:** The liars always misreport about honest peers, and always report 1 (trustworthy behaviour) about peers of their group, as exemplified in Fig. 2.

**Collusive chain:** The liars form an ordered circular set (chain): if there are $n$ liars, then peer $c_i$ always reports 1 for peer $c_{i-1}$ and misreports on all the others, $c_0 \equiv c_n$. Chains are most effective when loops bring gains, which normally is the case, for example, in social networks. The EigenTrust algorithm ([12])

effectively limits the attack, but a set of pre-trusted peers (chosen before the application starts) play a key role in this result, because they help the system keep the reputation of colluders low enough to prevent them from being selected. Without pre-trusted peers the algorithm has no means to combat the attack and the colluders irreversibly take over the system.

**Probabilistic chain:** A variation of the previous scenario occurs when malicious peers misbehave with a probability $f$, but behave collaboratively for the rest of the time; they form a chain as described above. Simulation results in [12] show that in this case colluders have a negative effect on the service of the system as a whole, because they earn trust by providing authentic content. Specifically, the maximum damage is done for $f = 50\%$. The authors argue that this scenario forces the colluders to spend resources in the system to gain their advantage; however, this argument is meaningful only if a malicious peer's goal is not the disruption of the system, but a service better than he deserves.

**Two collusive groups:** The population of liars further splits into two groups, which we call $L_1$ and $L_2$: a peer $i$ belonging to $L_1$ always behaves honestly ($p_i = 1$), but always report 1 for the service provided by any peer from $L_2$; a peer $j$ belonging to $L_2$ always report 1 for the services provided by any peer from $L_1$, but we do not do any hypothesis about the service it provides ($p_j$ is not necessarily 1). This means that peers of group $L_1$ gain high reputation thanks to their honest behaviour, and acquire high credibility to recommend peers of the group $L_2$.

**Infiltrators and Parasites:** A variation of the previous case. Colluders split into two groups: a first group (the Infiltrators) behaves collaboratively with every user and earns high reputation, but always assigns high trust values to a second group of colluders, who never collaborate (the Parasites). The Parasites earn high reputation from the scores assigned by the Infiltrators. With the same effort spent in the probabilistic chain, malicious users have twice the benefit. This kind of threat creates great damage with acceptable effort, and is therefore one of the most effective collusion schemes found so far.

## 3. Applications Suffer Collusion

Of all the possible classifications that can be used, we opt for a division among three dimensions: CPU, memory, and network. Each application shares among the peers one of those resources: file sharing and content delivery networks, for example, fall into the memory-sharing applications, while video streaming is more about sharing the network load, and grid computing is about sharing CPU power.

### *3.1. Memory and Storage*

Memory-intensive applications are systems where the service exchanged among peers is storage, be it semantic-less (pure memory space) or semantically meaningful, as in file-sharing systems.

Kamvar, Schlosser and Garcia-Molina [12] present a decentralized algorithm to build reputation by using the concept of global reputation, which is computed aggregating the local trust information of each node belonging to a subset of the network. Nodes belong to regions of responsibility according to the CAN DHT system described in Ratnasamy *et al.* [13]. The reputation of a target peer is computed by a set of M *score managers*, chosen by hashing the unique ID of the target node through M different hash functions. A score manager has to know the set of peers interacting with the target node, known as *daughter*, either receiving a service or providing it: in particular, the nodes that received service from the daughter give the score manager the reputation values about the daughter, which the manager in turn uses to compute a trust value. The authors show that this trust can be obtained by an iterative computation of the principal eigenvector of the normalized local trust values.

To counter collusion, this algorithm relies on pre-trusted peers. The assumption is that there are a set of peers (e.g., the designers of the system, or a number of mirrors managed by an organization) that can be trusted regardless of previous interactions. The algorithm prescribes the selection of peers at random with a given, parameterized probability to select the pre-trusted peers. Pre-trusted peers are a means of common sense, but their use just testifies that maliciousness and collusion are topics requiring further research.

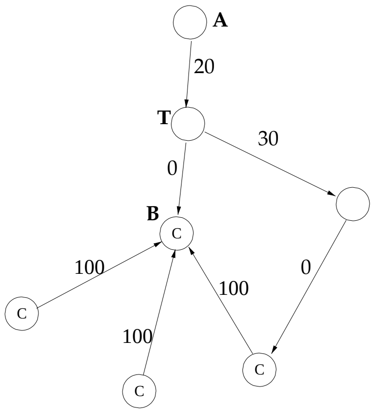


Figure 3: Example of collusion and how the Maxflow algorithm fights it.

To limit collusion, the Reciprocative function [14] uses a *maxflow algorithm* [15, 16] to compute a subjective reputation for the nodes. The maxflow algorithm operates on a portion of a digraph to give a value to the paths that may exist between two peers $A$ and $B$, as exemplified in Fig. 3. The digraph is built in the following way: the vertices are the peers, the edges represent the service a peer requested to another, while the weights are the reports about the service that one peer received from the other. The maxflow algorithm computes the path between the nodes calculating the reputation and the node under examination. The trust of the path is the minimum trust of the bottleneck link; the algorithm searches for the maximum value of all the possible paths from source to target. In Fig. 3, the node $A$ finds a flow of capacity 0 towards the node

$B$ because, even if the colluding nodes (indicated with a $c$ letter) report that $B$ helped them, the node $T$ whom $A$ trusts has never dealt with $B$, which is correctly identified as not trustworthy. The subjective reputation of the peer $B$ perceived by peer $A$ is:

$$min\left\{\frac{maxflow(j,i)}{maxflow(i,j)},1\right\}$$

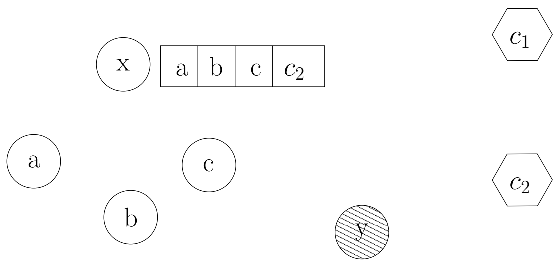
The drawback of Maxflow is its cost in terms of complexity, which is $O(V^3)$; however, a truncated version of the algorithm presents good properties of scalability, though in some cases no flow is found even if it exists.

Clearly if a chain of colluders gain the trust of $A$ the method fails. Indeed, Lian *et al.* [17] argue that the application of the EigenTrust algorithm to Maze (a file-sharing system) creates two problems: false negatives, that happens when a low-reputation colluder has even a one-time interaction with a high-reputation peer, so boosting his reputation; and false positives, that occur when there is a cluster network that downloads high volumes of files but shares them only with the nodes of the cluster. To limit these issues, they manipulate the trust computation algorithm and in particular the process of building the global trust from the local trust matrix. The key observation, that is partially shared by Piatek *et al.* [18], is that most of the peers have one interaction with any one peer, and an increasing number of interactions is less and less probable. The key observation of Piatek *et al.* , instead, is that there is a rough division among peers into two groups: the resourceful peers, that gain much reputation because they can serve large volumes of data, and less-resourceful peers, that can serve less data because of their limitations. Resourceful peers have a very high probability of serving the population, and in particular any two less-resourceful peers likely interact with a common resourceful one. The implication of this observation is that we can use a resourceful peer $R$ as an intermediary between any two peers that have interacted with him. This in turn means that we can use just the one-step reputation matrix to compute a good value for the reputation of any peer, if we know their intermediary.

We stress the fact that this kind of observations are unlikely to be applicable to streaming systems, and in general to any system where there are many more interactions, each one of very brief duration.

Similarly to the one hop reputation model, Marti and Garcia-Molina [7] model the collusion of a group of peers, particularly focusing on *front peers*, that always provide good service, but lie about the reputation of malicious peers. The reputation of a target peer is based on two components: the direct opinion of the peer, if he already had interactions with the target, and the (indirect) opinions that other peers have about the target peer. The two components are weighed according to the trust the peer has in the peers that express an indirect opinion about the target. Fig. 4 pictorially presents the situation, also reporting the trust building equations.

Xiong and Liu [8] suggest a slow-rising/fast-dropping scheme to fight peers accumulating high trust and then starting to misbehave: a large number of successful transactions build a high trust value; at the same time, however, trust decreases fast and few bad-rated transactions are enough to drop it. A proper use of a time window prevents peers from using long up-time periods and past good behaviour to misbehave effectively in the present.



$$
\begin{aligned}
t_x(y) &= Mt_a(y) + mt_{c_1}(y) \\
t_x(c_1) &= Mt_{c_2}(c_1) + mt_y(c_1) \\
& \qquad\qquad M >> m
\end{aligned}
$$

Figure 4: Use of the Friend Cache and problem of the front peer. In the figure, the front peer $c_2$ conquers a position in $x$'s Friend Cache, thus polluting the trust about the colluder $c_1$. $t_u(v)$ is the trust that peer $u$ has about peer $v$.

A way to counter collusion is to have a mechanism to demonstrate that an agent actually provided a service. Specifically, if a peer $p$ declares he received some service from peer $q$, it is desirable to have a proof that is actually happened: this would prevent the collusion between $p$ and $q$. This idea is studied by Reiter, Sekar and Zhang [19], who show preliminary simulation results by applying their system to the Maze P2P file-sharing application [20].

An entity, the *verifier*, wants to verify that a set of peers own a resource, assumed to be a piece of information $I$ in this case: he sends to each peer a puzzle, i.e., a question that can be easily and quickly answered only by the peers who own $I$. The question's answer can be found by hashing $I$; the hash function is universally known by the peers and is modeled as a random oracle[3]. A threshold $\theta$ represents the time by which the peers under trial have to solve the puzzle: if a peer exceeds it, then he becomes suspect of misbehaviour. The threshold is chosen in such a way that peers cannot collaborate: a peer that does have $I$ has just the time to solve his puzzle and send the response.

Under the assumption of random oracle, the article shows that a bound exists to the number of puzzles a set of colluders can collectively solve. This bound has a closed form but is hard to compute. To solve this problem, the authors prove the existence of a tighter but computable bound, not in a closed form. The model of collusion they adopt is based on the work by Lian *et al.* [17].

Barter systems are a flavor of tit-for-tat. A research prototype by Ngan, Wallach and Druschel [21] with the characteristics of a barter system is based on incentives provided in a totally decentralized way (no central authority). Both the benefit and the contribution are in terms of storage space, which represents the service. The attack

[3]A *random oracle* is the abstraction of a function that can produce a truly random output, and gives the same response to the same query.

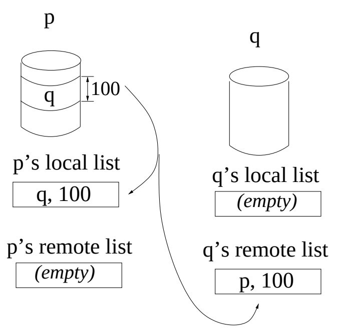


Figure 5: Remote storage application. Peer $p$ stores 100 units of storage for $q$; both report this value in the appropriate list.

scenarios consider the presence of collusion and compare the performance of the system when colluders are present or absent. The authors assume that the colluders are a minority, and can even try to bribe other peers to let them use their space without offering space in exchange. Let's see how the system works to understand its advantages and weaknesses.

Each peer $p$ owns a *usage file*, digitally signed with his private key, containing:

- the advertised capacity provided to store other peers' files,
- a local list with the files stored locally for others,
- a remote list containing information about $p$'s files stored remotely.

The peer can store his files whenever he is *under quota*, that is, his advertised capacity is larger than the storage he is consuming remotely in other peers' storage, that is, he consumes less than he offers.

Let's make an example (see also Fig. 5). Suppose that peer $p$ advertise 250 GB of space available for other peers. Peer $q$ reads the advertisement. He is hosting 100 GB of other peers' data, as reported in his local list, stores 140 GB of his own data remotely, as reported in his remote list, and advertises 300 GB: since $300 > 140$, $q$ is under quota and can ask $p$ for remote storage.

A malicious peer $c$ can lie about his advertised capacity, by claiming he can store locally more than he actually can, or he can lie about files stored remotely, by claiming he stores remotely less files than he actually does. In particular, colluders can form a chain to hide an imbalance between the storage quota offered and used by one misbehaving peer, known as the *cheating anchor*.

Attacks are prevented by using random auditing on top of anonymized communication. If $p$ is storing a file $F$ for peer $c$, he can query $c$ about his remote list. This list must contain an entry for peer $p$. Since the communication system is assumed to be anonymous, $c$ cannot know who is auditing him, and therefore he cannot know which entry he can hide. In fact, if $c$'s remote list maliciously lacks the presence of $F$ ($c$ tries to increase his under-quota situation by claiming he is using less space than he actually does), $p$ can delete him from his local storage, because $c$ is discovered as not paying for the storage he is consuming. The operation of auditing a node by a peer entertaining a service relationship with it is called *normal auditing*.

The chance for a normal user to evict a misbehaving user is a danger, though. A central authority can always verify the right to evict by examining the proof of misbehavior, because the response of the peer being audited is signed by the author, but introduces a single point of failure and a centralized entity in the system.

To discover collusion and chains of cheating, in theory, it is necessary to walk along the chain up to its originating cheating anchor, but this operation does not scale well with the number of peers in the system. For this reason, beyond normal auditing, peers are required to perform a *random auditing* on a randomly chosen peer, which they might have no relationship with. The authors prove that with high probability all the peers in the system are subject to audit, including the cheating anchor.

The punishment is ensured by the usage file being digitally signed, so the misbehaviour of the cheating peer is clearly and easily identified. However, if a set of colluders never tells the truth about fellow colluders, as the colluding party grows larger, the auditing mechanism becomes less effective, since most of the auditors collude with the peer subject to auditing.

Lian *et al.* [17] study a file sharing system in order to define a set of behaviors identifiable as collusive. In the system they analyze, the reputation is measured by points accumulated as content is uploaded to the other participants. A single central entity assigns users more points per byte for uploads rather than it takes away for downloads: this means that uploading and downloading the same amount of data produces a net gain. This property can be exploited by colluders to earn fake reputation (that is, without actually providing any benefit to the system) and use them to increase their own benefits (in this case, the download speed). The authors build a set of detectors to mark suspect behavior. They are the following:

1. Colluders produce a large amount of traffic with the same content to minimize the number of data uploaded and maximize the number of points gained;
2. Pairs of colluders can upload to each other large amounts of data with respect to the amount of data provided to the rest of the users;
3. Many identities on the same machine might be an attempt of a colluder of gaining reputation by uploading content to itself. This threat exploits the inexpensiveness of identity creation (like in the Sybil attack);
4. Colluders are likely to keep a facade behaviour by uploading small amounts of data to many peers while at the same time directing most of the uploaded data to a single partner. This behaviour is highlighted by a useful indicator named Traffic Concentration (TC).

## *3.2. Computing Resources*

Peer-to-peer networks are used to distribute the computational workload that would be overwhelming when used on a single machine. Distributing the load among machines allows resource-constrained users to exploit the idle cycles of machines whose owners join the system. This type of systems is sometimes called P2P Grid computing, and in some cases (e.g., [22]), it relies on a reputation system to determine the presence of incorrect results at one machine by having a majority of machines confirming the same outcome for the same job (or batch of jobs).

Common process containment techniques, like chroots and virtual machines, can actively limit the damage of a job that tries to abuse of the resources put at its command; we believe, however, that there is a good chance for coordinated adversaries to perform some of the attacks we have discussed in the previous section for storage systems.

To set the picture, we describe a seminal work, the basic system designed by Kim *et al.* [23], that follows. A client wants to have a job executed. He prepares a profile describing the resources required to process the job, and submits it to an *injection node*, which we can think about as a known entry point. The injection node assigns a global ID to the job and routes it to another node, who becomes the *owner*. From the owner node, after a matchmaking phase that consists in searching a node with the required resources, the job is assigned to a *run node*, which in turn starts processing it (after processing all the jobs arrived before). For the protocol to be reliable in case of failure of any of the nodes involved, nodes regularly exchange heartbeat messages.

To get past the limitations of this system, reputation systems can be used here too. Silaghi *et al.* [22] combine a direct and an indirect mechanisms to compute the trust of peers processing jobs, explicitly addressing the collusion problem in a peer-to-peer grid system used for distributed computation. In the original system, volunteer nodes (workers) provide their CPU power to run experiments over a large amount of common data sets. A master node distributes computation tasks for workers to run over the data sets, while data sets themselves are distributed using BitTorrent. Collusion is countered by using replication and consensus, that is, a result is deemed valid when a majority of the workers agree upon it. The original system always uses replication to validate the results, with large computation overhead.

To alleviate this load, the authors propose a weighted voting system to assess the validity of results, using trust values to compute a validity score for results. The setting is the following: we have $n$ workers that are assigned a work replicated $n$ times. The $n$ results are collected by the master, who stores a table containing trust values for each worker. Each result $r_j$ is assigned a score $s_j$ in the form

$$s_j = \frac{\sum_i \phi_{i,j} t_i}{\sum_i t_i}$$

where $\phi_{i,j}$ is 1 if peer $i$ ran the work $j$, 0 otherwise; the value $t_i$ is the trust for peer $i$, stored in the aforementioned table. If we define $s_j^* = max_j s_j$, then the result $r_j^*$ is accepted if $s_j^* > \theta$, where $\theta$ is a threshold properly chosen to guarantee the coherence of results in the presence of low reputation peers, but always greater than 0.5. Moreover, to avoid that low reputation workers (maybe forming a malicious coalition) undermine the correct result provided by a high-reputation worker, the authors require that the lowest reputation peer in a pool delivering the result $r_j$, say, $w_l$ (pivot), has a trust value $t_l > 0.5$.

A further study from the same authors ([24]) deal with the formation of coalitions that explicitly attack the voting system. In particular, they assume that the colluders follow a coordination protocol, and that they only attack when they know all the other members of the coalition are ready to start polluting a vote. The collusion detection system proposed in the paper shows that it is possible, in some settings, to trace the activity of colluders and tell them apart from honest peers; the only assumption, however, that colluders must not be aware of the detection algorithm being in place, weakens the result.

It seems that the awareness of collusion in P2P grid applications is greater than in other contexts. The research line described in [22, 24] is promising and works in some settings. It is desirable that once the detection problem is generally characterized and solved, a major research propulsion would go in the direction of how to limit the damage created by colluders, that is, in the design of prevention mechanism that break the collusion before it starts.

### *3.3. Network and Bandwidth*

Some systems may have peculiar characteristics that make tit-for-tat schemes particularly ineffective: video streaming applications are an example of this anomaly. Chu, Chuang and Zhang [1] propose an alternative incentive scheme based on linear taxation, that we can classify as direct in that it immediately offers service in exchange of contribution. As observed in the paper, most peers use an ADSL connection to access the Internet, and can thus benefit of a download bandwidth far larger than their upload bandwidth. In this case, with a tit-for-tat scheme peers would receive the stream at the speed determined by their upload bandwidth, that is a waste of the larger download bandwidth. This disincentivizes the peer's cooperation, making him leave the system or try to fool it.

Linear taxation is based on the balance of the following equation:

$$f = max(t \times (r - G), 0).$$

The term $t$ is the tax rate: when equal to 1, the scheme is equivalent to tit-for-tat. $r$ is the number of units of bandwidth the peer will receive if he contributes $f$ units of bandwidth; $G$ is a lump sum grant, or *demogrant*, and is basically a measure of the amount of contribution that a peer can voluntarily make without receiving back a benefit. The demogrant is equally distributed among participants. A central entity, the publisher of the content, decides the value of $t$, while $G$ is set by the system to achieve an overall budget balance. The decision is based on the value $f_i$ communicated by the peers according to their utility.

To choose a good value for $t$, the publisher should know the type of the peers in the system. Linear taxation is maximally effective when peers are heterogeneous; however, it makes the most resourceful peers receive a relatively small amount of benefit compared to their contribution, thus making its application to real systems with strategic players[4] appear complicated.

In the distributed algorithm that implements this taxation scheme, each peer $i$ modifies his strategy (the $f_i$ he communicates to the publisher) according to a personal estimation of the demogrant $G$. The estimation is based on a query sent to a subset of the neighbors to know their estimation of $G$. It is clear that a set of colluding neighbors

[4]A player is strategic if he always tries to maximize his outcome in a game.

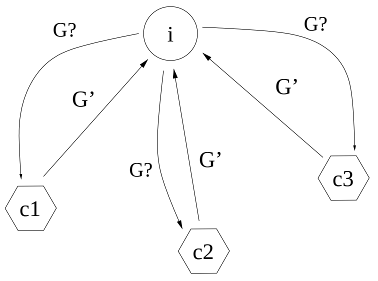


Figure 6: Linear taxation: collusion example. $c_i, i = 1, 2, 3$ are colluders: they report to the victim $i$, that queried them about their estimation of the demogrant $G$, a wrong estimation, $G'$. According to these values, $i$ is induced to believe that $G'$ is the real demogrant, thus computing a wrong (sub-optimal) strategy $f'$, different from the optimal $f_{opt}$.

may alter the value of $G$ they communicate as response to the query, in order to influence the utility that the victim chooses to communicate to the publisher (Fig. 6). This scenario, not explicitly included in [1], suggests further study.

Liu *et al.* [25] describe a system that fights free-riders by providing a method to punish misbehaviors, although not explicitly thought for collusion. In this system, peers proactively request video chunks to their neighbours according to their needs, rather than waiting for spontaneous donations (this architecture is known as *mesh-pull*). Any user $p$ serving chunks maintains a queue for each requesting neighbor, giving priority to peers that have given him more in the past. The incentive coincides with the priority acquired in the server's queue. With this system, free-riders are effectively discouraged because they receive a poor video quality: free-riders have bad positions in the queues of the peers they send requests to, thus receiving less chunks.

This scheme is effective against collusion because the server observes his own history with the client, instead of asking other peers about the contribution they received from him. Of course, this limits the interactions with new peers and poses problems in terms of newcomers policy: a peer joining the system has no contribution and may be discouraged from joining at all if his initial received quality is low.

## 4. Micropayment systems

According to Micali and Rivest [26], a payment system is a set of protocols with three basic actors: a buyer, a seller, and a bank. The actors can be individual entities or collections of entities. Micropayment systems (MPSs) are payment system where the single payment is of a very small amount. In this section we discuss some MPSs, analyze some theoretical collusion attacks, and draw the relevant conclusion that collusion can be fought by such systems. As micropayment systems involve a significant amount of cryptographic verification, they are mostly applied to static content distribution systems, so we analyze them in this context.

### *4.1. Application to CDNs*

Dandelion [27] and PACE [28] are two peer-assisted content distribution network systems based on micropayments. Both approaches describe a centralized bank system, but are open to the possibility of assigning its services to a distributed system.

Aperjis, Freedman, and Johari [28] model P2P CDN applications with a market metaphor of supply and demand, introducing an MPS-like system where the price is associated to a peer, rather than to a file. In this model, each resource that is potentially subject to congestion is assigned a price. This comes as a consequence of a more accurate model of the behavior of the users of a CDN system, based on multilateral exchange rather than on bilateral exchange. The network is modeled as a hierarchy with several theoretical levels (only two levels are used in practice), in order to roughly differentiate between the local link capacity of a peer in a local network, and the capacity of the link between the local and the wide-area network.

Each peer in the system runs a buy client and a sell client. The application offers a clean interface to a rendezvous service and to a network price service, that can be queried to know which peers own content, and the cost of the links (this feature can be used by network providers to operate the network more efficiently). The (logically) centralized currency service is based on strong identities (based on asymmetric key management) and provides an intermediary between a sell client and a buy client who have a transaction.

The bootstrapping process is suggested to be based on the download of content the new user is not interested in, in order to have them own some content they can offer to other peers to download and trade it with some content they are actually interested in.

In Dandelion [27], the bank is the trusted third party that watches over transactions among pairs of peers. It can manage thousands of peers because its normal operation is a common client-server system, that switches automatically to a peer-assisted scheme whenever the load becomes too heavy to manage efficiently. The cryptographic operations are limited because they are required in at most two steps of the protocol for each chunk of the content, and are fast because are based on symmetric cryptography (keys are shared between the server and each peer in the swarm).

The credit system is based on a small amount of volatile memory, that is written to stable storage after a number of transactions, in order to avoid the overhead of writing to disk for each chunk.

Transactions are rewarded with credit accrued by the uploader and charged to the downloader. The system deployer can use different policies for the selection of the chunk to ask for: the file-sharing-like approach benefits from a local-random rarest selection policy, similar to BitTorrent, while network-bound apps can gain more by stressing the importance of the play-out deadline of each chunk, and thus prioritizing chunks produced earlier from the source.

PPay is a system where cryptographic verification is based on asymmetric key. The proposal designed by Yang and Garcia-Molina [29] is a micropayment system that relies on self-managed currency (*coins*) under the control of a central entity called *broker* issuing the currency. Coin exchange and security issues are managed by the peers themselves, without intervention of the broker, unless punishment is required, or for some security aspects, as explained below.

$PK_b, SK_b$ are the pair of public and secret (private) key of the broker $b$. A peer $p$ gets a coin from the broker paying a sum. The coin sent by the broker is a signed message in the form

$$C = p, sn_{SK_b}$$

where $sn$ is the unique serial number of the coin. Peers receiving the coins directly from the broker are called *owners* of the coin.

From this moment on, the owner is responsible for the maintenance of the coin. The owner uses it to pay another peer $q$ in exchange for a service, thus granting $q$ the right to become the *holder* of the coin through an *assignment* that has the form

$$A_{pq} = q, seq_1, C_{SK_p}$$

The broker has no participation in the assignment phase. The holder can *reassign* the coin to a third peer, notifying the owner and making the older assignment no longer valid. In every moment, the owner is aware of who holds the coin and of the history of exchanges, in order to have a proof of acceptance or relinquishment in case of disputes with any peer that has held or still holds the coin.

When any party but the broker happens to be in a downtime phase, the remaining agent addresses the broker to require the reassignment or the cashing of the coin. Since this creates a load for the broker, he charges both the requester and the owner (when it comes on-line again) to perform the operation: this encourages peers to stay on-line as long as they can.

Leveraging PPay, Wei *et al.* [30] with their *WhoPay* consider the anonymity issue. WhoPay leverages on the system architecture of PPay, but ensures anonymity of peers that perform a transaction by using group signatures as discussed by Chaum and Van Heyst [31]. For the sake of fairness, however, the system requires the presence of a trusted entity, the *judge*, that, in conjunction with the broker, can identify the actors of each transaction. By using group signatures, agents are guaranteed to preserve their anonymity, unless they misbehave: in this case, the judge (and only him) is ensured to have the means to identify the peers involved in the transaction.

Also the solution presented by Catalano and Ruffo [32] is based on PPay. As an improvement of the basic interaction system, delegation of accountability is used to further reduce the involvement of the broker. The *accountability* is the possibility of linking an item, be it an object, an action or a right, to a responsible subject, who thus becomes accountable for it. The authors propose a mechanism to pass the accountability of a coin from one peer to another: the first peer, the grantor, passes to the grantee his right to delegate. To implement this mechanism, a second pair of public, private keys is required in addition to the usual one used to identify peers in front of a certificate authority (the broker). A delegation token is issued from grantor to grantee for each passage, thus it is always possible to reconstruct the chain of exchanges. The responsibility of such a verification is assigned to the grantee.

The micropayment systems considered so far are based on virtual currency, that is cashed cumulatively. A natural alternative is the payment of real money for each transactions.

Nair *et al.* [33] propose a system to incent agents in a BitTorrent-like system to favour the download of content by other agents. Each peer $p$, at his entrance in the system, generates a $\{PK_p, SK_p\}$ pair and contacts a central authority (the broker $b$, managed by a content provider), sending it the $PK$ and the coordinates of a valid credit account, used for the payments. As a second step, $b$ sends to $p$ the contact of a tracker, that in turn provides a list of candidate peers to select from and download content. In the same transaction, the broker $b$ also sends $p$ a pseudo-random sequence of numbers, the *tokens*, that $p$ will use to pay the providers of the pieces in which the content is divided. Specifically, one token will be given in exchange of one piece of the content. After the download, $p$ will send the provider $q$ the token, that in turn can decide to redeem it immediately or after some time by contacting the broker. Upon such request, the broker will take the corresponding money from $p$'s account and transfer it to $q$'s.

This system incents the upload of content by peers who are paid for their service, at the same time exploiting the resources of the network rather than the resources of the content provider.

## *4.2. Micropayment systems can defeat collusion*

Collusive attacks against a MPS systems are theoretically possible. We briefly describe their characteristics here, and then show how the solution found by Tran *et al.* [11] can rule them away.

In PPay [29], colluders could, for example, act as owner ($o$) and holder ($h$) of more coins. Peer $h$ claims it received coins from $o$ and wants to reassign them, but $o$ is offline[5], so he asks the broker to reassign the coins. The goal is to obtain the deposit made by $o$ at the entrance into the system. This strategy is ineffective because $h$ gets the coin reassigned, but $o$ cannot come on-line again, otherwise the broker would charge him with the cost incurred for the reassignment. Even if $h$ gives the sum to $o$, no gain is obtained, because the recovered money is the original deposit of $o$, who already owned it.

In the WhoPay system [30], basic collusive attacks are briefly discussed and proved to be easy to neutralize by the security architecture. An adversary can collude with the coin owner to force the holder to relinquish the coin; however, the holder can challenge the owner to prove the validity of the transaction, and, once proved it is illegal, he can make the owner be punished for his misbehaviour. The authors, however, do not address the attacks based on whitewashing and misbehaviour followed by change of identity. Furthermore, no particular attention is given to systematic collusive attacks, like distributed denial of service.

The study of Catalano and Ruffo [32] analyzes the effects of some collusive threats. As an example, efficiency reasons suggest for each peer to verify only the last step of delegation, thus allowing the chance for collusive peers occupying the last two steps in the delegation chain to provide counterfeit coins. Larger groups of colluders may create longer sub-chains, making it harder (that is, more demanding in terms of computation because more steps have to be verified) to discover the misbehaviour. The forgery can be detected from the broker at the end of the passing process (i.e., when the coin has to be cashed), or by any peer that examines the whole (in the worst case) delegation chain. The authors, however, recognize that collusion is not in the direct scope of the paper and suggest that the topic is an open research field.

Let's consider now the defenses that Floodgate [33] puts against different types of collusion. First, consider a peer $x$ who tries to ask the $N$ pieces of the file to $N$ distinct peers, receives the pieces, and then claims he did not actually receive them. If

[5]The system prescribes $h$ to ping the owner before contacting the broker.

the providers complain against $x$, the broker can decide to punish him. A colluder, however, can help $x$ by giving him a piece of the content without payment and without complaining with $b$, thus giving $x$ a way to fool the system. The authors study this scenario and find a constraint that relates the number of pieces of a file, the number of outstanding tokens (maximum number of tokens accumulated without asking the broker the payment) and the number of peers in the system, and then show that by carefully selecting these parameters it is possible to reduce this type of attack: conceptually, if the number of pieces is far larger than the number of peers that can be contacted simultaneously, then it is hard for $x$ to find enough colluders to provide him a way to gain his advantage.

While colluders can collaborate to attack the broker using DDoS attacks, the authors propose existing methods to alleviate the problem. Colluders can, for example, try and impersonate the broker by intercepting the requests of the peers; this would, however, require to know the private key the broker uses for any transaction in which it has to prove his identity. Still, colluders could simply intercept requests to block them and negating this way the chance for honest peers to get paid for their contribution, making the system's reputation fall down.

Finally, isolating peers (put them in minority) is hard to achieve for colluders. Suppose for instance that a group of colluders decide to complain against the broker about another honest peer. The system design defines the number of complaints that must receive in order to ban an agent from the working system, so the number of colluders must be quite large (assuming the parameters are wisely chosen by the designers) to fool the broker. In any case, the possibility exists.

Collusion can be effectively limited in systems based on virtual currency. This is partly due to the tight control exercised by the cryptographic primitives (reasonably assumed to be unbreakable) over the system. Particularly relevant to this discussion, the work by Tran, Li, and Subramanian [11] shows that collusion can be effectively detected and neutralized by measuring the diversity of contributions that the peers claim they have made and statistically check whether the credit the peer claims to own is as much as expected, given the distribution of credit among other peers. The additional benefit of the system comes from the decentralization of credit production (which is deferred to single peers), that allows a decentralized deployment and a significantly reduced number of checks. A central server is needed only to check for consistency of claimed credit (and effectively thwarts collusion).

## 5. The role of game theory

Game theory (see [34, 35, 36]) can be used to design incentives, either MPS or generic. Modeling the interactions among users as a game, it is possible to describe equilibria, that is, behaviors that are followed because they are the most convenient choice for each user.

In this sense, game theory can be seen as a meta discipline that lies in the middle between cooperation enforcement and sheer altruism: it lets the user choose his behavior, but creates a system where the best behavior (the one that creates most wealth) is the desired one.

Let's give some definitions useful for classification:

**Definition 1:** A homogeneous setting identifies a system where all the players contribute with the same resources. ■

**Definition 2:** A heterogeneous setting identifies a system where resources are unevenly distributed among users. ■

**Definition 3:** In a protocol $\Pi$ to which a set of peers $\mathcal{P}$ participate in order to maximize a utility function $u_i, i \in \mathcal{P}$, a situation of social dilemma arises whenever users earn more by defecting than by cooperating (that is, $u_d(i) > u_c(i)$, where the user $i$ gets a utility $u_d(i)$ when defecting, $u_c(i)$ when collaborating), but the whole system has a higher social welfare when everybody cooperates ($W_c > W_d$, that is, the social welfare $W$ achieved through collaboration is higher than through defecting). ■

### *5.1. Design*

A design based on game theory defines:

- A utility function that the peer tries to maximize, which in turn is based on a cost function and a benefit function;
- A set of actions peers can perform.

When peers aim at maximizing their utility and can decide actions consequently, they are said to be *rational* and *strategic*. A strategy is a set of actions defined as a response to any situation the peer can be called to respond to during the application session.

### *5.2. Examples*

Let's see some examples of game theory application, trying to understand the problems that may arise in the presence of a malicious coalition, i.e., of collusion. Buragohain, Agrawal, and Suri [37] introduce a design framework, where the strategy of a peer is the level of his contribution, that is, a peer can decide how much to contribute according to the situation.

$D_i$ denotes the level of contribution for peer $p_i$. $D$ can be anything meaningful in the application context A peer $p_i$ incurs a unit cost $c_i$ when he contributes a resource. If a peer contributes $D_i$, the total cost is $c_i D_i$, while the normalized contribution has the form: $d_i = \frac{D_i}{D_0}$, where $D_0$ is a generic contribution normalization.

A similar example of decision function is *Reciprocative*, introduced by Feldman *et al.* [14] and based on a measure of the generosity of peers, defined as the ratio between what they provide and what they consume, properly normalized to avoid that reciprocative peers would become defective to each other.

Now suppose that peer $p_i$ looks for a benefit by joining the system. He requests a resource to another peer $p_j$, who provides it according to a probability distribution, that depends on the contribution provided by $p_i$. By subtracting costs from benefits, $p_i$ decides whether to join the system or not. The utility function has the form:

$$U_i = -c_i D_i + p(d_i) \sum_j B_{ij} D_j$$

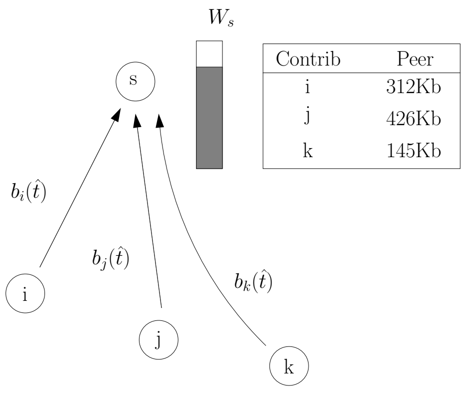
The first term is the contribution cost, and the second the total expected benefit. As we see, this expected utility (for $p_i$) is the sum of the resources provided ($-c_i D_i$) and the product of three elements:

1. $D_j$: what other peers contribute to the system,
2. $B_{i,j}$: how important their contributions are to $p_i$, i.e., their value to $p_i$,
3. $p(d_i)$: the probability that $p_i$ will get his requests satisfied, which is a function of its normalized contribution.

Within this model, the authors analyze the existence of Nash equilibria. The scheme effectively fights selfish free-riders; collusion, however, is neglected. The peers are assumed to be trustworthy and not malicious, and thus to correctly report about their contribution level. The authors admit the need for an audit mechanism to verify the reports from the peers, but no actual implementation rule is described in detail.

In the system designed by Keidar, Melamed, and Orda [38], each node keeps a balance between what it provides and what the neighbour provides him in terms of packets. The balance should never fall below a threshold conceptually similar to the imbalance ratio described in Li *et al.* [39]. If a peer does not own a sufficient number of packets, he asks the source to provide packets on its behalf, and pays a fee in terms of *fine* packets, i.e., dummy packets which do not contribute to its balance, but waste their resource (and the network's). It is clear that it is not in the interest of the peer to ask the help of the source.

The authors prove the existence of a Nash equilibrium if all the nodes choose strongly dominant strategies in the set of protocol-obedient strategies. This proof, however, holds if no peer joins or leaves the overlay. The basic assumptions are that most users obey because they do not have the technical skill to hack the application, or refrain from installing hacked applications. It is also assumed that no out-of-channel communication occurs among users: the results depend on the substantial isolation of users. For this reason, the system cannot be proved to be collusion-resistant: in particular, the absence of any malicious peer is assumed.



| Contrib | Peer |
|---|---|
| i | 312Kb |
| j | 426Kb |
| k | 145Kb |

Figure 7: System designed from Ma *et al.* [36]. The gray area in the rectangle $W_s$ shows the unavailable bandwidth at the source $s$. The source will distribute the available bandwidth according to each user's contribution until time $\hat{t}$.

Ma *et al.* [36] also design a framework with an interesting formal result about the amount of collusion the application can tolerate. The framework consists in a resource allocation mechanism that induces a competition among nodes requesting a service to a peer. The competition is modeled as a game: Peer $i$ expresses his requests through biddings ($b_i(t)$) sent to the source of a service $s$ (see Fig. 7). The source has a given amount $W_s$ of bandwidth to distribute to requesting peers, according both to their biddings and their contribution. The game is proved to have a unique Nash equilibrium, practically implementable with a linear time algorithm by perturbing the theoretical solution by a small positive amount $\varepsilon$. The result is not strictly a Nash equilibrium, but converges to an exact Nash equilibrium as $\varepsilon \to 0$. This makes the mechanism feasible and collusion resistant in the sense described above.

## 5.3. *Approximated equilibria*

Exact Nash equilibria are hard to compute and often fragile (e.g., in presence of churn). An approximated form of Nash equilibrium can sometimes be found to relieve the problem, and this is the idea used in the design of FlightPath [39].

The system is modeled by using the BAR setting, an acronym that describes the three types of behaviors a peer can follow: they can behave Altruistically, that is, always be loyal to the rules of the protocol; or they can be Rational, which means that they are ready to deviate when this is more convenient to them; or finally, they can behave in a Byzantine way, that is, they misbehave randomly, and particularly without pursuing a utility.

In this setting, the approximated NE is characterized by the property that it is not valid in every round, that is, a rational peer may temporarily gain more by deviating. Let's consider the optimal strategy and the utility that comes out of it, that we call $u_o$. We can describe the relative advantage of the optimal cheating strategy over the strategy that obeys the protocol (that we can call $u_e$) as follows:

$$\begin{aligned}\varepsilon &= \frac{u_o - u_e}{u_e} \\ &= \frac{(j_e - j_o)\beta - (w_o - w_e)\kappa}{(1 - j_e)\beta - w_e\kappa} \\ &= \frac{\frac{cj_e}{1-j_e} + (1-b)}{c-1}\end{aligned}$$

where $b$ is a fraction ($b < 1$) of the bandwidth used to run the protocol (that can be lower-bounded), $c$ is the benefit-to-cost ratio, $j$ is the jitter (expected, $j_e$, and actual, $j_o$), $\beta$ measures a benefit, $\kappa$ a cost. In steady state, the user has to upload at least $min_{up} = \lceil \frac{N_{need}}{1+\alpha} \rceil$, with a corresponding cost of $cost = \gamma + min_{up} \times \rho$ ($\alpha$ is the ratio between what should be provided and what is actually provided, or *imbalance ratio*). The parameter $\gamma$ represents the fixed cost of a trade in kbps, while $\rho$ is the increase in cost for each chunk uploaded; finally, $N_{need}$ is the number of chunks a peer needs in each round. To find the equilibrium, we can solve for $c$ with the objective $\varepsilon = 0.1$: we find that this is an equilibrium if the rational peer values the stream at least 3.36 times more than his cost in bits.

The system is proved to be robust against 10% of Byzantine peers, and resilient against selfish behaviour. Long term strategies performed by malicious colluding peers, however, are not considered explicitly and their study is left as a future work.

### 5.4. *Game theory and incentives*

Morselli, Katz, and Bhattacharjee [40] propose a game-theoretic framework to compare trust-inference protocols, that is, protocols where the frequency of interactions between peers depends on the trust they have in each other. The authors define a measure for robustness that can be used to compare different incentives systems.

Given a generic protocol that defines the way trust is computed and assigned to each peer, and a malicious peer $A$ who knows the protocol, can see every message exchanged between any two peers, and can interfere sending messages, a protocol is said *robust* if $A$ maximizes his utility by obeying to the protocol. The authors, however, do not provide an example of a protocol which is robust and at the same time fights collusion, even if the framework allows modeling malicious/selfish coalitions.

#### 5.4.1. *Mechanism Design*

Mechanism design (along with Algorithmic Mechanism Design, or AMD, and Distributed AMD, or DAMD) [35, 41, 42] has a strict dependence on game theory, and has been used as a modeling tool for the construction of incentive systems. Mechanism design (MD) tries to induce a behaviour onto selfish users by designing the payments and the punishments for good and bad behaviour, respectively. The difference with Game theory is that this studies a behavior in a system, while MD design the system to induce a behavior.

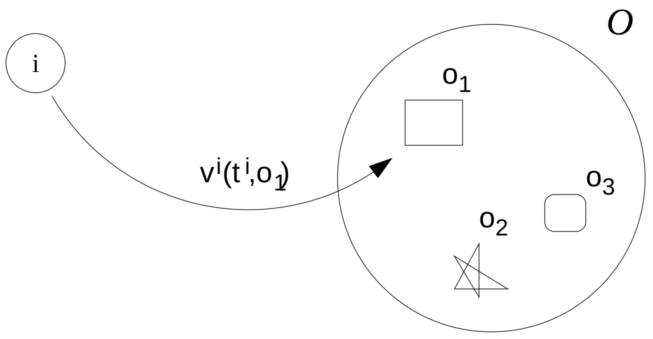


Figure 8: Valuation of the output $o_1 \in \mathcal{O}$ from peer $i$ having type $t^i$.

Formally, in a generic MD problem, we have $n$ users having a *type* each, denoted as $t^i \in T^i$ for user $i$. The type is a privately known input, while the other information is publicly known. A mechanism design problem is composed of an output specification, that maps $t = t^1...t^n \mapsto o \in O$, where $O$ is the set of allowed outputs, and a set of utility functions for the users. According to his type $t^i$, each user gives a value, called *valuation*, to any output, in the form $v^i(t^i, o)$. The utility can be expressed as the sum of the number of currency units assigned by the mechanism to the user ($p^i$), plus his valuation of the output (see Fig. 8):

$$u^i = p^i + v^i(t^i, o).$$

Feigenbaum and Shenker [41] survey the state of the art in the Distributed Algorithmic Mechanism Design (DAMD) field, which can be exemplified as the study and the design of distributed systems where participants, assumed to be self-interested, have to be properly incented to follow the system's algorithm. Specifically, DAMD addresses both incentive compatibility and computational tractability in systems where users and resources are *distributed*. The authors illustrate the open problems inherent to the DAMD problems formulation: first, they show that a measure of the hardness of a problem is still missing; second, they question the value of approximated solutions to hard problems; eventually, an overview of solution concepts alternative to Nash equilibria and dominant strategies are investigated, and indirect mechanisms are presented as possible candidates to direct mechanisms.

The design of a mechanism depends on a model of the problem, and its approximation when the problem is hard. Only basic results exist about approximation: in particular, strategically faithful approximations are problem approximations where the strategic properties of the original model hold, but other properties are approximated. There exist three such approximations:

1. $\varepsilon$-dominance: In this approximation, the players do not play the best strategy, but one approximation that lies within a factor $\varepsilon$ from the optimal strategy (see Li *et al.* [39] for an example of application);
2. Feasibly strategyproof mechanisms: A better strategy exists, but it is infeasible to compute for *all* the users; the infeasibility derives from computational limits;
3. Tolerably manipulable mechanisms: A mechanism is tolerably manipulable if it is not group-strategyproof, but we can characterize the types of malicious groups that can form, and we can demonstrate that their effects on the community are tolerable. Such mechanisms answer the question: How large can the effects of malicious coalitions be? For this reason, they are particularly interesting for the study of collusion.

As we said above, the problems from a mechanism design perspective are usually studied with a special focus on dominant strategies and Nash equilibria, which are known to be hard to compute. However, alternative strategies may be found. In a distributed complex environment like the Internet, it is realistic to assume that the users do not know of the existence of each other, and they observe different payoff for themselves applying the same strategy when network conditions change. In this case, we obtain an interesting simplifying assumption by modeling the system as an iterative game, in which the players do not know about the payoffs of other players, but can iteratively learn them by observing the output of the choices as they appear in the network.

Finally, the authors show that the design of indirect mechanisms offers a trade off between the privacy of the users, who just partially (if at all) reveal their utility functions, and the network complexity that derives from the need to retrieve that information in other ways, which usually involves the explosion of the number of messages to exchange.

## 6. Discussion and Conclusions

In this survey, we have analyzed the literature about P2P systems in order to find out if and how the problem of collusion is taken into account —and eventually fought. Single or non-coordinated free riders and malicious users are normally considered when analyzing security of P2P systems; however colluding users stand to unorganized malicious users and free riders as organized criminality stands to petty robbers, but the complexity of collusion behavior and analysis seams to defeat or scare away most researchers. The study of collusion in P2P systems is of the utmost importance, because of the challenges it creates and the risks it puts on P2P systems that are founded on the notion of collaboration and are based on the sharing of some resources. Moreover, it must be remembered that techniques derived from botnets can be easily imported into P2P systems, making a single ‘real’ user the master of a strategic collusion system, and this rises the level of danger: If the coordinated action of hundreds of different users may be rare, the coordinated control of hundreds of nodes by a single user is much more probable, specially if by doing so the single real user has a large personal benefit, and can get away leaving the controlled, artificial nodes to be blamed for the havoc.

As any survey the goal is taking a snapshot (as accurate as we are able to do) of the state of the art on the survey subject, in order to rise the community awareness on the topic and to be the base for further active research in the field. Not all the papers we have included in the bibliography deals with collusion: many of them, as many others we have not included, simply admit that “problems related to collusion are left for future work and research”, implicitly stating the importance of the topic and, at the same time admitting that research is lagging behind systems’ threats in this field.

The analysis we carried out took different angles to the problem: per application, per methodology, per shared resource. We have distinguished anti-collusion methods based on the presence of incentives, luring users to collaborate, from methods based on ‘enforcement by design’ aiming at evicting misbehaving users from the system.

Regardless of the angle of observation or perspective, we found that there are many open issues in P2P systems in order to embed anti-collusion systems in them. Theoretical research is required to better understand the dynamics of collusion and their impact on the service provided. Experimental research is needed to complement theory, but also to collect data about real systems behavior and workout. Indeed, this seems to be one of the most uncovered areas: the availability of hard facts and data about the impact and effects of collusion in real systems. To conclude the survey we discuss the possibility that a new level of awareness on the problem may spawn a positive loop of research and implementation that may lead to strong results against collusive behaviors paving the road for a new generation of services based on the P2P networking paradigm.

### *6.1. Theoretical directions of research*

MPSs and trust-based systems are existing, but incomplete means to counter collusion. Moreover they seem to be in place more as a side-effect than by purposeful design choice. The role of game theory seems promising, as the motivations that we find for malicious nodes to cooperate are intuitively suitable to game models. Game theory is a complex discipline, however, and we recognize that the efforts made in the research on (Distributed) Algorithmic Mechanism Design, intended to make the computation of the equilibria tractable by machines, are an important part of the process of application of game theory itself to the field of collective attacks.

In particular, the approach that uses approximated equilibria is encouraging since it cuts complexity and offers, in some cases, ways to compute the error with respect to the optimal equilibrium. In particular, Nash equilibria tend to be fragile, in the sense that in a real environment, with errors, churn, and all sort of impairments, they cannot be reached, with the risk of leading to protocol deadlocks or livelocks. An approximated equilibrium instead defines a region of sustainable operation and should lead to more robust and resilient protocols.

Indeed, from a theoretical point of view a formal way to measure the gravity of collusion-based attacks is still missing, as well as recognized methods to define how vulnerable systems are to such attacks. A first step towards this goal can be the definition of metrics and a classification system based on them. Albeit far from a complete definition, this survey provides a first attempt in this direction.

Related to the lack of metrics, the sheer proof that collusion can be prevented in real (not trivial) systems is still missing. The lack of such results, positive or negative, leaves the community in the uncertainty of the scope and limits of collusion effects.

### *6.2. Experimental directions of research*

Some of the theoretical directions of research might be very hard and lead to dead-ends. All the same collusion exists. Under this perspective, traffic measures to provide evidence of the damage of collusion attacks, to detect repetitive patterns, and to confirm the reliability of systems built according to the results of the theory are needed. As theory provides metrics, we can classify the existing systems and show how vulnerable they are against collusion attacks. Experimental research can also focus on the comparison among systems, and find strong points to repeat in future systems to further limit their vulnerability.

Also, testing tools generating representative patterns of collusion, with known and proven coverage, would be extremely useful for the comparison of different protection techniques.

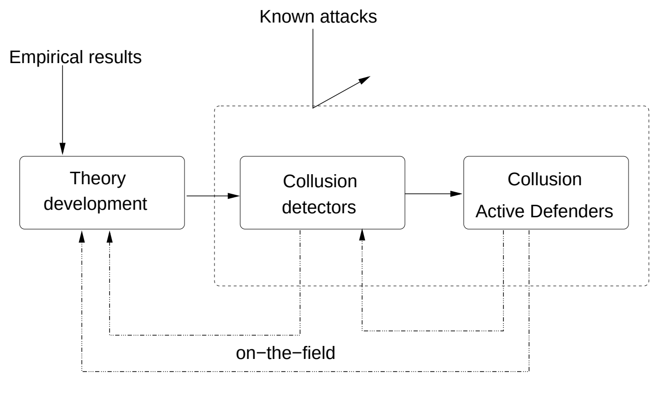


Figure 9: The advancement of research in countering collusion. After each feedback, new systems will incorporate the improvements determined by theory and practice. While known attacks are repelled, new attacks fire the loop and demand further improvements to research.

### *6.3. A path against collusion*

On top of the output provided by theory and practice, the objective is to isolate the results in a modular classification, and sample applications, in order to provide developers with a reliable and tested set of tools that guarantee the resilience of the system against collusion attacks.

We see two goals along this direction: research will first provide tools to *detect* collusion, in as distributed a way as possible; second, tools will break collusion and report about the actions taken: we can call such tools collectively an *anti-collusion active defense*.

Just as security enforcement, also anti-collusion protection is not a destination but a path: As new threats rise, the countermeasures must becomes more sophisticated, spawning a classical cycle of system development as depicted in Fig. 9.

## References

[1] Y. Chu, J. Chuang, H. Zhang, A Case for Taxation in Peer-to-Peer Streaming Broadcast, in: Proc. of the ACM SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04), Portland, Oregon, USA.

[2] A. Singh, T. Ngan, P. Druschel, D. S. Wallach, Eclipse Attacks on Overlay Networks: Threats and Defenses, in: INFOCOM'06, Barcelona, Spain.

[3] D. Levin, K. LaCurts, N. Spring, B. Bhattacharjee, BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives, ACM SIGCOMM Computer Communication Review 38 (2008) 243–254.

[4] S. Marti, H. Garcia-Molina, Taxonomy of trust: Categorizing P2P reputation systems, Computer Networks, Elsevier, 50 (2006) 472–484.

[5] Z. Despotovic, K. Aberer, P2P reputation management: Probabilistic estimation vs. social networks, Computer Networks, Elsevier, 50 (2006) 485–500.

[6] M. Gupta, P. Judge, M. Ammar, A Reputation System for Peer-to-Peer Networks, in: Proc. of the ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03), Monterey, CA, USA.

[7] S. Marti, H. Garcia-Molina, Limited Reputation Sharing in P2P Systems, in: Proc. of the 5th ACM Conference on Electronic Commerce (EC'04), New York, NY, USA.

[8] L. Xiong, L. Liu, PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 843–857.

[9] N. Ntarmos, P. Triantafillou, SeAl: Managing Accesses and Data in Peer-to-Peer Sharing Networks, in: Proc. of the 4th IEEE International Conference on Peer-to-Peer Computing (P2P'04), Zurich, Switzerland.

[10] Y. Gil, V. Ratnakar, Trusting Information Sources One Citizen at a Time, in: Proc. of the 1st International Conference on The Semantic Web (ISWC'02), Sardinia, Italy.

[11] N. Tran, J. Li, L. Subramanian, Collusion Resilient Credit-based Reputations for Peer-to-peer Content Distribution, in: Workshop on Economics of Networks, Systems, and Computation (NetEcon'10), Vancouver, British Columbia, Canada.

[12] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The EigenTrust Algorithm for Reputation Management in P2P Networks, in: Proc. of the 12th ACM International World Wide Web Conference (WWW'03), Budapest, Hungary.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, A Scalable Content-Addressable Network, in: Proc. of the 2001 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01), San Diego, California, USA.

[14] M. Feldman, K. Lai, I. Stoica, J. Chuang, Robust Incentive Techniques for Peer-to-Peer Networks, in: Proc. of the 5th ACM Conference on Electronic Commerce (EC'04), New York, NY, USA.

[15] R. Levien, A. Aiken, Attack-Resistant Trust Metrics for Public Key Certification, in: Proc. of the 7th USENIX Security Symposium, San Antonio, Texas, USA.

[16] M. K. Reiter, S. G. Stubblebine, Authentication Metric Analysis and Design, ACM Transactions on Information System Secururity 2 (1999) 138–158.

[17] Q. Lian, Z. Zhang, M. Yang, B. Zhao, Y. Dai, X. Li, An Empirical Study of Collusion Behavior in the Maze P2P File-Sharing System, in: Proc. of the 27th IEEE International Conference on Distributed Computing System (ICDCS'07), Toronto, Ontario, Canada.

[18] M. Piatek, T. Isdal, A. Krishnamurthy, T. Anderson, One hop Reputations for Peer to Peer File Sharing Workloads, in: Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08), San Francisco, California, USA.

[19] M. K. Reiter, V. Sekar, Z. Zhang, Making Contribution-Aware P2P Systems Robust to Collusion Attacks Using Bandwidth Puzzles, Technical Report CMU-CS-08-156, School of Computer Science, Carnegie Mellon University, 2008.

[20] M. Yang, Y. Dai, X. Li, Bring Reputation System to Social Network in the Maze P2P File-Sharing System, in: Proc. of the IEEE International Symposium on Collaborative Technologies and Systems (CTS'06), Las Vegas, Nevada, USA.

[21] T. J. Ngan, D. S. Wallach, P. Druschel, Enforcing Fair Sharing of Peer-to-Peer Resources, in: Proc. of 2nd International Workshop on Peer-To-Peer Systems (IPTPS'03), Berkeley, California, USA.

[22] G. C. Silaghi, L. M. Silva, P. Domingues, A. E. Arenas, Tackling the Collusion Threat in P2P-enhanced Internet Desktop Grids (2008) 393–402.

[23] J.-S. Kim, B. Nam, M. Marsh, P. Keleher, B. Bhattacharjee, D. Richardson, D. Wellnitz, A. Sussman, Creating a Robust Desktop Grid using Peer-to-Peer Services, in: IEEE International Symposium on Parallel and Distributed Processing (IPDPS'07), Long Beach, California, USA.

[24] G. C. Silaghi, F. Araujo, L. M. Silva, P. Domingues, A. E. Arenas, Defeating Colluding Nodes in Desktop Grid Computing Platforms, Journal of Grid Computing (Springer) 7 (2009) 555–573.

[25] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, Y. Wang, Using Layered Video to Provide Incentives in P2P Live Streaming, in: Proc. of the 2007 ACM SIGCOMM Workshop on Peer-to-Peer Streaming and IP-TV (P2P-TV'07), Kyoto, Japan.

[26] S. Micali, R. L. Rivest, Micropayments Revisited, in: Proc. of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology (CT-RSA'02), San Jose, California, USA.

[27] M. Sirivianos, J. H. Park, X. Yang, S. Jarecki, Dandelion: Cooperative Content Distribution with Robust Incentives, in: Proceedings of the USENIX Annual Technical Conference (ATC'07), Santa Clara, California, USA.

[28] C. Aperjis, M. Freedman, R. Johari, Peer-Assisted Content Distribution with Prices, in: Proc. of the 4th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT'08), Madrid, Spain.

[29] B. Yang, H. Garcia-Molina, PPay: Micropayments for Peer-to-Peer Systems, in: Proc. of the 10th ACM Conference on Computer and Communications Security (CCS'03), Washington, D.C., USA.

[30] K. Wei, A. J. Smith, Y. Chen, B. Vo, WhoPay: A Scalable and Anonymous Payment System for Peer-to-Peer Environments, in: 26th IEEE Intenrational Conference on Distributed Computing Systems (ICDCS'06), Lisboa, Portugal.

[31] D. Chaum, E. Van Heyst, Group signatures, in: IACR Workshop on the Theory and Application of of Cryptographic Techniques (EUROCRYPT'91), Brighton, UK.

[32] D. Catalano, G. Ruffo, A Fair Micro-Payment Scheme for Profit Sharing in P2P Networks, in: 1st IEEE International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P'04), Volendam, The Netherlands.

[33] S. Nair, E. Zentveld, B. Crispo, A. Tanenbaum, Floodgate: A Micropayment Incentivized P2P Content Delivery Network, in: Proc. of the 17th International Conference on Computer Communications and Networks. (ICCCN '08), St. Thomas, U.S. Virgin Islands.

[34] A. E. Roth, The Economist as Engineer: Game Theory, Experimentation, and Computation as Tools for Design Economics, Econometrica, The Econometric Society, 70 (2002) 1341–1378.

[35] N. Nisan, A. Ronen, Algorithmic Mechanism Design, Games and Economic Behavior, Elsevier, 35 (2001) 166–196.

[36] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, D. K. Y. Yau, Incentive and Service Differentiation in P2P Networks: A Game Theoretic Approach, IEEE/ACM Transactions on Networking 14 (2006) 978–991.

[37] C. Buragohain, D. Agrawal, S. Suri, A Game Theoretic Framework for Incentives in P2P Systems, in: Proc. of the 3rd International Conference on Peer-to-Peer Computing (P2P'03), Linköping, Sweden.

[38] I. Keidar, R. Melamed, A. Orda, EquiCast: Scalable Multicast with Selfish Users, in: Proc. of the 25th Annual ACM Symposium on Principles of Distributed Computing (PODC'06), Denver, Colorado, USA.

[39] H. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robinson, L. Alvisi, M. Dahlin, FlightPath: Obedience vs. Choice in Cooperative Services, in: Proc. of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI'08), San Diego, California, USA.

[40] R. Morselli, J. Katz, B. Bhattacharjee, A Game-Theoretic Framework for Analyzing Trust-Inference Protocols, in: 2nd Workshop on the Economics of Peer-to-Peer Systems (P2PECON'04), Cambridge, Massachusetts, USA.

[41] J. Feigenbaum, S. Shenker, Distributed Algorithmic Mechanism Design: Recent Results and Future Directions, in: Proc. of the 6th ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM '02), Atlanta, Georgia, USA.

[42] J. Feigenbaum, R. Sami, S. Shenker, Mechanism Design for Policy Routing, in: Proc. of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC'04), St. John's, Newfoundland, Canada.