Game-Theoretic Mechanisms to Increase Data Availability in Decentralized Storage Systems

Krzysztof Rzadca, *Member, IEEE,* Anwitaman Datta, *Member???, IEEE,* Sonja Buchegger, *Member, IEEE* and Gunnar Kreitz

Abstract

In a peer-to-peer storage system, peers replicate each other's data in order to increase availability. Compared to organizationally-centralized solutions, such as cloud storage, a peer-to-peer storage system may result in smaller monetary costs and may require less trust in the provider. The data availability provided by the system is a function of the participating peers' availability. However, a straightforward system in which peers' matching is decentralized uses the given peer availability inefficiently. The highly-available peers form cliques replicating data between each other, which makes the system too hostile for the weakly-available newcomers. In contrast, a centralized, equitable matching is not incentive-compatible: it does not reward users for keeping their software running.

We solve this dilemma by a mixed solution: an "adoption" mechanism in which highly-available peers donate some replication space, which in turn is used to help the worst-off peers. We show that the adoption motivates peers to increase their availability (is incentive-compatible); but also that it is sufficient for acceptable data availability for weakly-available peers.

Index Terms

price of anarchy, equitable optimization, p2p, p2p storage

[FIXME: File Version: serenity.tex 238 2012-10-11 09:59:13Z krz]
[FIXME: IEEE TPDS limits: 14 pages]

- *Krzysztof Rzadca is with the Institute of Informatics, University of Warsaw, Poland. E-mail: krzadca@mimuw.edu.pl*
- *Anwitaman Datta is with the School of Computer Engineering, Nanyang Technological University, Singapore.*
- *Sonja Buchegger and Gunnar Kreitz are with the KTH Royal Institute of Technology, Sweden.*

Game-Theoretic Mechanisms to Increase Data Availability in Decentralized Storage Systems

1 Introduction

A decentralized system for data storage and replication is an important building block of many peer-to-peer (p2p) applications, such as backup (in which the data should be durable despite of failures), or social networks [1] (in which, when a user is off-line, the system ensures that the user's friends can still access the data). In such systems, individual users (peers) store other users' data. Data storage uses not only storage space but, more importantly, consumes bandwidth [2]. In return, a user expects that her data will also be stored remotely, increasing availability and resilience. A generic storage system cannot use relations between users that are present in some applications (e.g., "friends" in a social network [3]). Thus, users are assumed to be independent [4], [5], and they seek to maximize their perceived profits (e.g., availability of their data) and to minimize their contribution (e.g., the amount of other users' data they store). Thus, the crucial decision a user must make is to choose other users that will replicate her data (and whose data she will replicate, assuming a reciprocity-based scheme). Depending on the organization of the system, this decision is either done through the agency of a centralized matching system, or using a fully decentralized algorithm in which users form replication agreements autonomously [6], [7].

In this paper, we study the problem of maximization of *data availability* in a decentralized data replication system. We use a stochastic model of peer availability. We study two scenarios of matching, i.e., organizing replication agreements between peers: centralized and enforced; or decentralized and autonomous.

In order to derive worst-case bounds, we assume that peer availability is stochastic and constant in time. A peer's availability is the probability of the peer being available (correlated with the peer's expected lifetime, like in [7], [8]). The goal is to maximize data availability given the constraints on the storage size. In our earlier work [9], we also investigated an alternative model, in which availability was deterministic (on-line/off-line), but varied in time. We omit these results here because of space constraints; the alternative model offered higher data availability only when peers had varied availability patterns (corresponding to a world-spanning system).

We analyze the problem when matching is done either *centrally* or in a *decentralized* manner. A *centralized* system collects information about the peers' availabilities and then derives replication groups so that the expected availability (or resource usage) is optimized in a manner equitable to all the participants. In a *decentralized* system, each peer seeks to find replication partners maximizing her own data availability.

The paper has the following contributions. In centralized systems with global knowledge, we prove that achieving equitable allocation is NP-complete. We propose heuristics to achieve equitable availability. According to our experimental evaluation, for all classes of peers this heuristic achieves data availability two to three orders of magnitude higher than random allocation.

In the decentralized matching model, we prove the existence of a unique subgame perfect equilibrium, in which peers replicate data with other peers that have similar availability (thus, the most available peers replicate data only among themselves). Such an equilibrium may seem inefficient for the system goal, as we show an instance in which it is arbitrarily far from the equitable solution, i.e., the price of anarchy [10] is unbounded. Yet the equilibrium is fair in the sense that the peers who are stable and highly available have their data replicated better than erratic, unstable peers. We propose a distributed heuristic that according to our experimental evaluation efficiently converges to the subgame perfect solution.

We also propose a set of rules for game-theoretic mechanism that reduces the price of anarchy and, at the same time, provides incentives to peers to be highly-available. We present a heuristics that constructs allocations according to the rules for the mechanism. According to the experimental evaluation, the heuristic increases the data availability of the worst-off peers by two to three orders of magnitude.

The paper has the following organization. Section 2 reviews the related work. Section 3 introduces the mathematical model. The centralized matching (NP-hardness, fairness, heuristics) is analyzed in Section 4. The decentralized matching (the Nash equilibrium, the price of anarchy) is analyzed in Section 5. Section 6 proposes a framework for a game-theoretic mechanism that increase the data availability of the weakly-available peers. A distributed algorithm for matching peers is presented in Section 7. Section 8 presents results of simulation of the algorithms. Finally, Section 9 discusses relaxations of some of our assumptions and summarizes the paper.

2 Related Work

Our paper analyzes the problem of replica placement in a p2p storage system. In order to optimize the data availability, we explicitly choose the nodes on which the data is to be replicated (unlike, e.g., DHTs [11]).

Many papers assess loss of efficiency of the system caused by selfishness of individual participants, starting

from the analysis of selfish routing in [10]. [4] studies incentives for system-optimal behavior in p2p systems. [12] considers a game of network creation in which selfish nodes optimize the cost of creating new links that minimize distances to other nodes. [13] considers a similar model in a p2p system. In contrast, in p2p storage systems, the "transaction" (i.e., storing data of another peer) is persistent rather than temporary; unlike in the classic Prisoner's Dilemma, a peer's utility from "cheating" is small, as the cheated party can easily detect cheating and quickly adjust its strategy. Interaction is repeated — similarly to BitTorrent [14] — thus, the typical problems of enforcing agreements can be easily solved using reciprocity and a basic reputation system; which is why in the game theoretic analysis (Section 5), we focus on the process of forming, rather than enforcing, replication agreements.

The most common approach to p2p storage is to place *enough* replicas *randomly* in the system. For instance, [15] analyzes how many replicas are required to achieve a desired level of availability. Existing systems based on such random placement include Pastiche [16] and Total Recall [8].

Approaches that explicitly optimize the placement of replicas include [6], [7], [17], [18]. [7] studies by simulation an algorithm similar to the Pragmatic Queries algorithm (Section 7), with a slightly more complex acceptance ($score(i,j)$) function. Similarly to our theoretic and simulation results, the experiments in [7] show that highly available peers achieve better performance than the peers with lower availability.

[17] and [18] analyzed a problem similar to the decentralized allocation in our model. [17] observes that the "system stabilizes when peers are grouped into clusters, pooling users that have similar profiles", however the proof is left for future work. In our paper, Proposition 7 provides a proof for the analogous behavior in our model; moreover, we compute the price of anarchy (Proposition 10). Additionally, we prove that the centralized optimization of availability is NP-hard (Propositions 1-2), which is hypothesized, but deferred for future work in [18].

Fair optimization of availability (Section 4) is similar to the problem considered in [6], where replicas of individual files are spread over a pool of hosts with given availabilities; however, as [6] has an implicit assumption of a single owner of the system, it does not have to consider reciprocity, nor cliques, in the assignment.

[19] assumes that the replica placement policy is partly determined by locality constraints. The "Buddy" policy is similar to our Clique-Based allocation; however our cliques are optimized, and not partly fixed. By simulation, the authors conclude that locality-aware policies are less efficient than the global, randomized allocation; in contrast, our Equitable heuristic is more efficient than the random allocation.

Our aim is to build a generic storage system; a specialized storage system could use real-life relations between users (e.g., friend-to-friend storage for an online social network [3]).

We propose (Section 6) a truthful mechanism that considers only a single parameter of the system: peers' availability. An alternative scheme is to consider also storage space. With asymmetric agreements [20], weakly-available peers contribute more disk space to their highly-available replication partners to compensate for their availability.

The payoff function in the replication game (Definition 3), in which a player's payoff depends on her availability (type) and availabilities of her replication partners, fulfills the definition of a hedonic game [21]. However, a (general) hedonic game has a complex structure of payoffs: player's preferences are defined over all coalitions she might belong to. In a more restricted model, utilities are symmetric and additively-separated [22]. Our replication game has even more restricted structure of payoffs: each player's contribution to the coalition's payoff is the same — equal to the player's unavailability. Using this restriction, we prove results stronger than those for the generic hedonic game: in particular, the price of anarchy (Proposition 9) and the truthful mechanism to reduce it (Section 6).

3 System Model and Assumptions

A system is composed of n *peers*, representing independent users. The i-th peer is denoted by p_i, or alternatively by i if it is not ambiguous.

For simplicity, we assume that peers are homogeneous in terms of storage needs and available storage resources to share. Thus, all the peers provide a storage space of s units to the system and need to store one unit of data. To make mathematical analysis tractable, we assume that peers replicate data by storing complete copies, in contrast to erasure codes. Erasure codes might offer additional gain in data availability.

We study a snapshot of the system, assuming that peers' availabilities are given and constant (i.e. are parameters of an instance of a problem). Adopting a matching to changing availabilities is out of the scope of the paper (although we give some hints in the discussion section).

Peer i's *availability* a_i is the probability of being online. Peer i's *unavailability* representing the probability that peer i is not available is given by $u_i \triangleq 1 - a_i$. We assume that the availability is known and cannot be tampered with by peers, e.g., using [23].

In order to simplify the notation of proofs, we assume that peers are numbered according to non-increasing availabilities, $u_i \leq u_{i+1}$. We also assume that n is divisible by $s+1$ (otherwise, at most s "virtual" peers with $a_i = 0$ can be added to the system).

Assume that i's data is replicated at peers $\{i_1, i_2, \ldots, i_k\}$, where $i_1 = i$. i's *data* is thus unavailable with probability d_i equal to the probability of the event that all the replicators are offline (transient failure),

$d_i = \prod_{j=1\ldots k} u_{i_j}$ (assuming that the peers' transient failures are independent [4], [5]).

In several of our models, we assume that peers form replication *groups* (*cliques*) $\{G_k\}$. Each member $i \in G_k$ of the group replicates data of all other members $j \neq i \colon j \in G_k$. We define group unavailability $d(G_k) \triangleq \prod_{i \in G_k} u_i$. Note that, $\forall i \in G_k \colon d_i = d(G_k)$. We denote group size as $\sigma_k = |G_k|$. We do not use groups in the availability-encouraging mechanisms in Section 6. Such groups have several advantages compared with the pair-based allocation. Firstly, groups are naturally formed in the subgame perfect solution of the decentralized versions of the problem (see Proposition 7 that considers the problem without assuming replication groups and proves that such groups will be formed). Secondly, with groups, it is easier to optimize some of the system's parameters not directly considered in this paper, like data dissemination during updates, when a group can form a spanning tree. Thirdly, as groups are based on reciprocity, it is easier for peers to directly control their replicas and react to free-riding.

As peers are assumed to be rational and to derive utility from availability of their data, each peer wants its data to be replicated as well as possible. Thus, i minimizes its d_i by choosing, or forming, a group with $d(G_k)$ as small as possible. Depending on the constraints of the system, this goal can be expressed in two ways:

1) given the storage size s, find s peers $\{i_2, \ldots, i_s\}$ such that $d(G_k) = d(\{i\} \cup \{i_2, \ldots, i_s\})$ is minimized;
2) given the maximal tolerable unavailability ϵ, minimize the size of the replicating group $\min \sigma_k = |G_k|$.

Both types of goals can be expressed also from the system's perspective. The system should guarantee that peers are treated fairly by optimizing unavailabilities of *all* the groups $\{u(G_k)\}$. In this paper, instead of a multi-objective approach, we will aggregate the groups' goals using two aggregating functions: (i) $\sum d(G_k)$ (expressing the average-case behavior); (ii) $\min \max d(G_k)$ (which can be inefficient for the system by focusing on the worst-off group). Section 4.2 provides motivation for aggregating over groups, rather than individual peers.

4 Centralized Model: Complexity, Bounds and Heuristics

This section analyzes the model with stochastic peer availabilities from the perspective of a centralized system that organizes the replication agreements. First, we prove that equitable optimization of the availability is NP-hard. Then, we propose aggregation functions used to characterize fairness of the system; and propose lower bounds which we later use for the price of anarchy estimation. Finally, we show a simple, greedy heuristic that organizes peers into cliques.

4.1 Complexity of Centralized Matching

The task of the fair matching system is to divide peers into replicating groups such that: (i) the probability of data being online is as high as possible; (ii) the size of each group is bounded. In this section, we first define a simple version of this replication problem and prove that it is NP-complete. Then, using this result, we prove that both system-level problems are NP-hard (namely, optimizing availability given constraints on the storage, and optimizing group size given minimal availability).

We define a simplified version of the replication problem as follows.

Definition 1. *An instance of the decision version of a Simple Stochastic Fair Replication Problem (SSFRP) is given by the set of the peers' unavailabilities $\{u_i\}$ and bound B. We ask whether there is grouping G_1, G_2 such that both groups are non-empty and $d(G_1) + d(G_2) \leq B$.*

Note that this problem is "simple", because we consider only two groups, we do not consider limits on the size of each group and we use a simple sum as an aggregation of groups' unavailabilities.

Proposition 1. *The decision version of the Simple Stochastic Fair Replication Problem is NP-complete.*

Proof: SSFRP is trivially in NP: given the grouping G_1 and G_2, it is sufficient to compute the corresponding products, which is in $O(n)$. The proof of hardness is by reduction from PARTITION [24]. In PARTITION, given a set of positive integers $\{b_i\}$, the goal is to decide whether it is possible to construct two disjoint subsets Y_1 and Y_2, such that $\sum_{b_i \in Y_1} b_i = \sum_{b_i \in Y_2} b_i = \frac{S}{2}$, where $S \triangleq \sum b_i$.

We construct an instance of SSFRP from an instance of PARTITION as follows. The i-th peer's unavailability corresponds to i-th integer $u_i = 2^{-b_i}$. Bound $B = 2 \cdot 2^{-S/2}$.

Given a solution Y_1, Y_2 to partition, the corresponding grouping $i \in G_1 \Leftrightarrow b_i \in Y_1$ is a solution to SSFRP. As $\sum_{b_i \in Y_1} b_i = \frac{S}{2}$, $\prod_{i \in G_1} 2^{-b_i} = 2^{-S/2}$. The same holds for G_2, thus $\prod_{i \in G_1} 2^{-b_i} + \prod_{i \in G_2} 2^{-b_i} = 2 \cdot 2^{-S/2} \leq B$.

Given a solution G_1, G_2 to SSFRP, the corresponding subsets are a solution to PARTITION. We denote as $2^{-l_1} = \prod_{i \in G1} 2^{-b_i}$ and $2^{-l_2} = \prod_{i \in G2} 2^{-b_i}$. Thus, $2^{-l_1} + 2^{-l_2} \leq 2 \cdot 2^{-S/2}$. We now show by contradiction that $l_1 = l_2 = S/2$. Assume $l_1 > l_2$. As $2^{-l_1} + 2^{-l_2} \leq 2^{1-S/2}$, $2^{-l_2} < 2^{1-S/2}$ (as $2^{-l_1} > 0$). Thus, $l_2 > S/2 - 1$ which leads to $l_2 \geq S/2$ (as l_2 is an integer). From the assumption of the proof, $l_1 > l_2$, thus $l_1 > S/2$, thus $l_1 + l_2 > S$, which leads to a contradiction, as by definition $S = \sum b_i = l_1 + l_2$. □

As the most unrestricted version of the replication problem is NP-complete, other problems are similarly NP-complete. Maximizing availability with constrained resources (Optimum Availability with Constrained Storage, OACS) translates into a problem of forming groups with a limited number of members. SSFRP can be thus solved by OACS with unlimited groups.

Corollary 1. *The problem of optimizing availability given the maximum number of peers in each group (OACS) is NP-complete.*

Restricted size problems are, generally, harder than

non-restricted ones. For instance, creating groups with exactly 3 members corresponds to the strongly NP-complete 3-partition problem [24].

Finally, we consider the problem of minimizing the used storage space, given a constraint on the minimal availability of a group (Optimum Storage with Constrained Availability, OSCA). OSCA is solved by forming the maximum number of groups such that each group provides at least the required availability level R.

Definition 2. *The Decision version of OSCA is defined as follows. Given the peers' unavailabilities $\{u_i\}$, we ask whether it is possible to construct at least N disjoint groups $\{G_1, \ldots, G_N\}$, so that in each group G_j $\prod_{i \in G_j} u_i \leq R$.*

Proposition 2. *OSCA is NP-complete.*

Proof: The proof is by reduction from DUAL BIN PACKING [25]. In DUAL BIN PACKING, the task is to partition items with sizes $\{b_i\}$ into at least N disjoint sets (bins) $U_1, \ldots U_N$, so that the sum of elements in each set is at least B ($\sum_{b_i \in U_j} b_i \geq B$).

The reduction maps i-th integer b_i to i-th peer's unavailability $u_i = 2^{-b_i}$. The level B is mapped to maximum unavailability of the group $R = 2^{-B}$.

A solution $\{U_j\}$ of DUAL BIN PACKING is a valid solution of OSCA: $G_j = \{i \colon b_i \in U_j\}$, as $\sum_{b_i \in U_j} b_i \geq B$, thus $\prod_{i \in G_j} u_i = 2^{-\sum_{b_i \in U_j} b_i} \leq 2^{-B} \leq R$.

Similarly, a solution $\{G_j\}$ of OSCA is a valid solution of DUAL BIN PACKING ($B_j = \{b_i \colon i \in G_j\}$), as $\prod_{i \in G_j} u_i = 2^{-\sum_i b_i \in U_j b_i} \leq 2^{-B}$, thus $\sum_{b_i \in U_j} b_i \geq B$. □

4.2 Characterization of a Fair Allocation

In multi-agent optimization, a function f_i corresponds to an outcome of an individual agent f_i. In order to optimize performance of all agents, but avoid multi-objective optimization methods, these functions are aggregated (commonly as $\sum f_i$ or $\max f_i$) which transforms the problem to a single-goal optimization ($\min \sum f_i$, $\min \max f_i$). The type of solutions depends on the aggregation function: for instance, $\max f_i$ focuses on the worst-off agent, at the expense of others; on the other hand $\sum f_i$ ignores what was the base on which an improvement in one of f_i was made.

In the subsequent analysis, in order to make our results more general, we will use two "fair" aggregations: $\sum_i d_i$ (replication level of an average peer) and $\max d_i$ (worst replication level). To further simplify the formulation of some proofs, we will aggregate over groups, rather than over individual peers. Trivially, $\max d_i = \max_G d(G)$. Similarly, $\sum_{G_k} d(G_k)$ can be used instead of $\sum_i d_i$: a corollary from Proposition 3 is that $\sum_i d_i = \sigma \sum_{G_k} d(G_k)$.

As the problem of deriving a fair allocation is NP-hard, in this section we show some properties of the optimal fair allocation. These properties will help to derive the price of anarchy in Section 5.

The following two propositions state that it is sufficient to analyze allocations in which all the replication cliques are complete (have the maximum number of peers).

Proposition 3. *Any solution with replication groups having less than $\sigma = s + 1$ peers can be transformed to a solution with all groups having exactly σ peers and a smaller or equal value of $\sum_i d(i)$.*

Proof: We analyze groups having less than σ members. The total number of peers across all such groups is divisible by σ. Repeat the following until all groups have exactly σ members. Take the group G with maximal $d(G)$ among these groups and assign its members to other non-full groups (non-full groups have space to accept these peers, otherwise the number of peers in these groups would not be divisible by σ). Observe that, overall, $\sum_i d(i)$ has not been increased, as former members of G are assigned to groups G' having $d(G') \leq d(G)$; and after the re-assignment, $d(G')$ is also not increased (and decreases if the reassigned peer has $u_i < 1$).

□

Proposition 4. *Any fair solution optimizing $\min \max G_k$ can be transformed to a solution with all groups having exactly σ peers (without increasing the value of the goal function).*

Proof: The proof is similar to the proof of Proposition 3. □

The following proposition shows a lower bound for the $\min \max$ aggregation.

Proposition 5. *A lower bound for $\min \max d(G_k)$ is u_G^σ, where u_G is the geometric average of $\{u_i\}$, $u_G = \sqrt[n]{\prod u_i}$.*

Proof: The proof is by contradiction. Assume $\min \max d(G_k) < u_G^\sigma$. In such a solution,

$$\forall_{G_k} d(G_k) < u_G^\sigma,$$

thus

$$\sum_{k=1,\ldots,N} \log d(G_k) < N \log u_G^\sigma.$$

As $N = \frac{n}{\sigma}$ and $\log u_G = \frac{1}{n} \sum \log u_i$, we get

$$\sum_{k=1,\ldots,N} \log d(G_k) < \sum_{i=1,\ldots,n} \log u_i. \tag{1}$$

However, as each peer belongs to exactly one group, in any solution:

$$\sum_{k=1,\ldots,N} \log d(G_k) = \sum_{k=1,\ldots,N} \sum_{i \in G_k} \log u_i = \sum_{i=1,\ldots,n} \log u_i,$$

which contradicts (1).

□

The following proposition shows a lower bound for the other considered aggregation, $\min \sum$.

Proposition 6. *A lower bound for $\min \sum d(G_k)$ is $\frac{n}{\sigma} \left(\min_{i \in \{1,\ldots,n\}} u_i \right)^\sigma$*

Proof: In each group G_k:

$$d(G_k) = \prod_{i \in G_k} u_i \geq \left(\min_{i \in G_k} u_i \right)^\sigma \geq \left(\min_{i \in \{1,\ldots,n\}} u_i \right)^\sigma.$$

Thus,

$$\sum d(G_k) \geq N\Big(\min_{i\in\{1,\ldots,n\}} u_i\Big)^{\sigma},$$

and the proposition follows. □

4.3 A Greedy Heuristic for the Fair Allocation Problem

The following greedy heuristic optimizes the assignment of peers to cliques in OAFS (Section 4.1), assuming global knowledge and coordination of peers. The idea of the algorithm is similar to the First Fit Decreasing approximation algorithm for minimum bin packing [24].

Firstly, the peers are sorted by non-increasing availabilities $av(i)$. Then, $\frac{n}{\sigma}$ most available peers are assigned to separate cliques. Finally, for each of the remaining peers (in the sorted order), the peer is assigned to a non-complete clique G_k that currently has the highest unavailability $G_k = \arg\max_{Gk':|Gk'|<\sigma} \prod_{i\in Gk'} u_i$.

We experimentally compare this algorithm to the random allocation in Section 8.2. The assignment resulting from the above heuristic may be further optimized by a global search meta-heuristic, such as Simulated Annealing (SA). However, in our initial experiments, SA did not significantly improve results returned by the heuristic, probably because of the large number of cliques to consider.

5 Game Theoretic Analysis of the Decentralized Matching

In decentralized matching, we assume that each peer is selfishly interested in maximizing the availability of her data. In this section, we predict replication agreements that will be formed in such systems. We model the resulting game as an extensive game in which peers change their replication agreements. We show that in the unique subgame-perfect equilibrium peers will form replication agreements with peers of similar availability. The drop in the system's efficiency (both for $\sum_{Gk\in\mathcal{G}} d(G_k)$ and $\max_{Gk\in\mathcal{G}} d(G_k)$) in this equilibrium is unbounded.

5.1 Replication System as a Game

As data replication is a long-lasting agreement, two distinct phases can be logically distinguished. In the first, *organizational* phase, each peer forms zero, one or more agreements with other peers in which she commits to storing their data. In the second, *production* phase, the peer can either honor the previous agreements, or break some of them by explicitly dropping other peers' data or by lowering her availability. Naturally, in a real system these two phases will overlap, as peers come and go. In such systems, contracts can be negotiated with a grace period during which they can be broken—the grace period, together with making contracts that start in the future, corresponds to the organizational phase discussed below.

The game in the *production* phase of the system is trivially similar to the repeated Prisoner's Dilemma [26]: a peer prefers not to honor the previous commitments, as storing data consumes peer's resources. However, the goal of the replication system is to make data available over a longer time period, thus the game can be modeled by the infinitely repeated Prisoner's Dilemma with a discount factor δ close to 1. Thus, breaking the agreements is only profitable in a very short term: when j detects that i stopped replicating j's data, j will not only break all her agreements with i, but also notify other peers of a "cheater" (directly, or with the help of a reputation system), which, in turn, can effectively exclude i from the replication system.

5.2 Definition of the Game

The game occurring in the *organizational* phase is much more interesting. We formally define it as an extensive (multi-round) game with infinite horizon and simultaneous moves [26]. Intuitively, in each round of the game, zero, one or more peers propose to replicate other peers' data and/or withdraw previous proposals. The game ends when no peer changes her set of replicating peers.

Definition 3. *The* Stochastic Replication Game *(SRG) is defined as an extensive game with infinite horizon and simultaneous moves, in which:*

- *the set of* players *is equal to the set of peers;*
- *the set of* terminal histories *contains list of sets* $(\{r_k(i, 0 \vee 1, j)\})$*, i.e., sets of replication proposals (denoted by* $r_k(i, 1, j)$*) or withdraws of previous proposals* ($r_k(i, 0, j)$*, possible only when* $\exists k' : r_{k'}(i, 1, j)$*), made by peers* (i) *to other peers* (j) *in each round* k*; in each round, for each peer, the number of active replication proposals does not exceed the peer's storage capacity* s*; all terminal histories end with an empty set* $\emptyset$*;*
- *the* player function $\mathcal{P} = \{p_i\}$*, i.e., after all histories all players can make proposals;*
- *each player minimizes the expected unavailability of her data computed as a product of unavailabilities of players who propose replicating the player's data (and who do not withdraw their proposals in subsequent rounds). We denote by* R_j *the replication set of* j *(after a terminal history), i.e.,* $R_j = \{i : (\exists k_1 : r_{k1}(j, 1, i)) \wedge (\nexists k_2 > k_1 : r_{k2}(j, 0, i))\}$*. The* pay-off *is* $d(i) = u_i \prod_{j:\, i\in R_j} u_j$*.*

The game is defined as an extensive game to model the fact that during the organizational phase peers will react to other peers' decisions and adapt their replication sets accordingly. Similarly, the game is not repeated, as the game models the organizational phase in which the payoff is computed for the production phase rather than for the short-term state after each round. For this theoretical analysis we do not limit the number of rounds in the game.

5.3 The Nash Equilibrium and the Price of Anarchy

We study the outcome of the game assuming that peers' strategies are tit-for-tat based, i.e., if peer i proposes

to replicate j's data in round k ($r_k(i,1,j)$) and peer j does not propose to replicate i's data in the subsequent round at the latest ($\nexists k' \leq k+1\colon r_{k'}(j,1,i)$), peer i will withdraw its proposal in the next round $r_{k+2}(i,0,j)$. This assumption on strategies helps peers to coordinate their actions. At the same time, such strategies are flexible and allow peers to react to actions of other peers.

The following proposition shows the *subgame perfect* [26] equilibrium of the game. Every subgame perfect equilibrium is a Nash equilibrium. In extensive games, the notion of the Nash equilibrium is considered artificial, as it is based on so-called empty threats. In contrast, the subgame perfect equilibrium requires that each player's strategy must be optimal for every history after which the player moves. In order to illustrate the difference, assume that $s = 1$ (each peer can replicate data of only one other peer), and $u_1 < u_2 < u_3 < u_4$. If p_2 and p_3 commit to mutual replication, and p_4 has a tit-for-tat strategy and replicates p_1, in the Nash equilibrium p_1 must replicate p_4 data—after p_1 proposes to replicate p_2, p_2 would not withdraw p_3, even though it is optimal for her to do so.

Proposition 7. *In a subgame perfect equilibrium of the Stochastic Replication Game, assuming that peers use tit-for-tat strategies, peers form $\frac{n}{\sigma}$ replication cliques of size σ. The cliques group peers with similar availability. If peers are numbered according to non-increasing availabilities ($u_i \leq u_{i+1}$), the k-th clique is formed by peers $\{1+\sigma(k-1), 2+\sigma(k-1), \ldots, \sigma+\sigma(k-1)\}$. The equilibrium is unique if and only if the peers' availabilities differ, i.e., $\forall i\colon u_i < u_{i+1}$.*

Proof: The game ends when no peer changed her proposal in the next to the last round (denoted by k). As the outcome is subgame perfect, given the other peers' actions, for each peer it was *optimal* not to change any of her proposals in round k. The proof is by contradiction.

By induction, we show that peers replicate in cliques. For the first clique, assume that peer i ($1 \leq i \leq \sigma$) replicates data of at least one peer $j' > \sigma$. Thus, by tit-for-tat, at least one peer j from $\{1,\ldots,\sigma\}$ does not replicate i's data (instead replicating data of $j'' > \sigma$). Thus, i could increase her availability by stopping replication of j' ($r_k(i,0,j')$) and proposing $r_k(i,1,j)$: as $u_i < u_{j''}$ it is optimal for j to stop replicating j'' ($r_{k+1}(j,0,j'')$) and start replicating i ($r_{k+1}(j,1,i)$) (otherwise, by tit-for-tat, i would withdraw her proposal for j). This contradicts the assumption that it is optimal for i not to change her proposals in round k.

By similar reasoning, if $|R_i| < s$ (i-th storage is not fully utilized), or if $|R_j| < s$, $r_k(i,1,j)$ results in $r_{k+1}(j,1,i)$, thus both i and j gain in availability.

For the i-th clique, observe that, by the induction assumption, all the peers $\{1,\ldots,(\sigma)(i-1)\}$ replicate data between themselves, thus none of them replicates with peers $\{1+(\sigma)(i-1),\ldots,n\}$. Consequently, the same reasoning as for the first clique applies, as peers belonging to i-th clique can either replicate between themselves, or with peers with lower availability. □

The grouping corresponding to the subgame perfect equilibrium is easy to achieve in a system with centralized information in $O(n \log n)$ time. It is sufficient to sort the peers according to non-increasing availabilities and then form replication cliques as in Proposition 7.

In order to compute the price of anarchy, we prove upper bounds for both fairness models (the following two propositions), and then show an instance in which the bounds are tight.

Proposition 8. *In $\max d(G_k)$ aggregation, the price of anarchy is at most $\left(\frac{\max u_i}{u_G}\right)^{\sigma}$.*

Proof: Value of $\max d(G_k)$ in the subgame perfect solution is determined by the group composed of the most unavailable peers that, in turn, is at most $(\max u_i)^{\sigma}$. In an equitable solution, $\max d(G_k) \geq u_G^{\sigma}$ (Proposition 5). □

Proposition 9. *In $\sum d(G_k)$ aggregation, the price of anarchy is at most $\left(\frac{\max u_i}{\min u_i}\right)^{\sigma}$.*

Proof: In any grouping (including subgame-perfect solution),

$$\sum d(G_k) = \sum \prod_{i \in G_k} u_i \leq \sum \Big(\max_{i \in G_k} u_i\Big)^{\sigma} \leq N\Big(\max_{i \in \{1,\ldots,n\}} u_i\Big)^{\sigma}.$$

The optimal grouping has (Proposition 6):

$$\sum d(G_k) \geq N\Big(\min_{i \in \{1,\ldots,n\}} u_i\Big)^{\sigma}.$$

□

The following proposition shows an instance in which the bounds are tight, and thus the subgame perfect equilibrium can be arbitrarily far from the equitable solution.

Proposition 10. *In the Stochastic Replication Game, the price of anarchy is unbounded.*

Proof: Consider an instance with $\sigma = s+1$ highly available peers (with $u_i = u_h \to 0$) and $\sigma \cdot s$ unavailable peers ($u_i = u_l \to 1$).

A equitable solution minimizing both $\sum_k d(G_k)$ and $\max_k d(G_k)$ constructs σ cliques; in each clique there is exactly one highly available peer and s less available peers (indeed, any assignment in which there are more than one highly available peer in the same clique has worse overall availability). Thus, $\sum_k d(G_k) = \sigma u_h u_l^s$; and $\max_k d(G_k) = u_h u_l^s$.

In the subgame perfect solution, highly available peers form a clique, thus leaving the less available peers to form cliques between each other. Thus, $\sum_k d(G_k) = u_h^{\sigma} + s u_l^{\sigma}$; and $\max_k d(G_k) = u_l^{\sigma}$.

The price of anarchy for $\min \sum d(G_k)$ is thus equal to:

$$\frac{u_h^{\sigma} + s u_l^{\sigma}}{\sigma u_h u_l^s} = \frac{u_h^s}{\sigma u_l^s} + \frac{s u_l}{\sigma u_h} \xrightarrow[u_h \to 0]{} \infty.$$

Similarly, for $\min\max d(G_k)$, the price of anarchy is equal to:

$$\frac{u_l^\sigma}{u_h u_l^s} = \frac{u_l}{u_h} \xrightarrow[u_h \to 0]{} \infty.$$

□

We discuss the consequences of such a high price of anarchy in the experimental evaluation (Section 8.2).

6 Availability-encouraging adoption mechanism

The high price of anarchy makes the completely decentralized system almost unusable for the weakly-available peers. On the other hand, a fair, centralized solution (Section 4) does not offer incentives for peers to be highly available. In this section, we propose an alternative to these two solutions: an algorithm that uses some replication slots of the highly-available peers to help the others; but also rewards more available peers with better data availability.

6.1 Characterization of a truthful mechanism

An availability-encouraging mechanism uses some of the replication slots to make the availability distribution more fair. Peers use the remaining slots to form replication agreements as in the Stochastic Replication Game (SRG, Definition 3). The following game focuses on the role of the mechanism, at the same time approximating the subgame-perfect equilibrium of the SRG by approximating replicator's availability by the peer's declared availability.

Definition 4. *The mechanism, given the declared unavailability level u'_i of each player ($0 \le u_i \le u'_i \le 1$), assigns for each player i a set of replicators S_i; and for each replicator $j \in S_i$, sets the exposed unavailability level $u_j(i)$ ($u_j(i) \ge u'_j$). The pay-off of ith player is given by:*

$$d_i(u'_i, u_{-i}) = (u'_i)^{\sigma - |S_i|} \prod_{j \in S_i} u_j(i). \tag{2}$$

Note that both S_i and $u_j(i)$ are functions of declared unavailabilities $[u'_i]$; we omit the vector from the list of arguments in order to simplify the notation.

The notion of the exposed unavailability $u_j(i) \ge u'_j$ permits to artificially diminish the availability of i-th peer data stored at j: this will be crucial to design an availability-encouraging mechanism. Such a limit can be easily implemented: e.g., j can serve only a $u_j(i)/u_j$ fraction of requests for i-th data.

The fundamental problem is how to design the taxation mechanism so that the players are motivated to declare their true availabilities, $u'_i = u_i$. The following definition formally captures that idea.

Definition 5. *A taxation mechanism is* availability-encouraging *(or* truthful*) if and only if for each peer i, given other peers' unavailabilities u_{-i}, the peer obtains higher or equal data availability by declaring its true availability: $d_i(u_i, u_{-i}) \le d_i(u'_i, u_{-i})$.*

Informally, availability-encouraging taxation "rewards" peers who increase their availability by increasing their data availability. A taxation mechanism should be availability-encouraging, because otherwise the peers don't have incentive to be highly available. As being available has some real costs (like electricity), in a system with an availability-*dis*couraging taxation peers would choose low availability levels $u'_i > u_i$; thus the system would collapse.

Corollary 2. *An algorithm that forms cliques according to the subgame-perfect equilibrium (Proposition 7) is availability-encouraging.*

6.2 A truthful mechanism with guaranteed reduction of the price of anarchy

In this section, we propose an algorithm that builds replication agreements between peers that is truthful and has a guarantee on the reduction on the price of anarchy. The algorithm is somewhat "wasteful", in a sense that not all available replication slots are used; yet its regularity allows us to analyze its worst-case behavior. In the subsequent section we present a heuristic that is more efficient in the average case; yet, because of the complex agreements, it is impossible to analyze theoretically.

The mechanism is implemented by a centralized algorithm that assigns to r replication slots of each highly-available peer r distinct weakly-available peers ($1 \le r \le \sigma$), such that each weakly available peer is "adopted by" (assigned to) k distinct peers ($1 \le k \le \sigma$). For instance, if $r = 1, k = 1$, the upper-half of peers adopts the lower-half; if $r = 2, k = 1$, the top $1/3$ adopt the bottom $2/3$. $\tau = u(n\frac{k}{r+k})$ denotes the unavailability of the last highly-available peer (e.g. for $r = 1, k = 1$, τ is the median of $\{u_i\}$; for $r = 2, k = 1$, τ is the first tercile). To simplify the notation, we assume that nk is divisible by $r + k$; otherwise, the last highly-available peer is $\lceil \frac{nk}{r+k} \rceil$, and its r replication slots are used to replicate peers $\{(\lceil \frac{nk}{r+k} \rceil + 1), \ldots, (\lceil \frac{nk}{r+k} \rceil + r + 1)\}$ (which are thus adopted $k + 1$ times).

The algorithm keeps a list L_W of weakly-available peers adopted less than k times (the list is initialized by $(n\frac{k}{r+k} + 1, \ldots, n)$). The list is ordered by increasing unavailabilities u. Highly-available peers are processed in order of increasing u: each highly-available peer i adopts first r peers from L_W (denoted by j), thus $S_i = \arg\min_{j \in L_W} u_j$ and $S_j = S_j \cup \{i\}$.

Next, if $k < r$, for each weakly-available peer, $r - k$ dummy peers with $u_d = 1$ are added to the replicas set S_j. These $r - k$ replications slots are wasted: as we demonstrate later, we need this for the truthfulness of the mechanism; at the same time, these slots do not play a role in the reduction of the price of anarchy. A heuristic proposed in the next section uses these slots more effectively (but has no asymptotic guarantee on the price of anarchy).

Finally, the exposed unavailabilities are set. Adopted peers j expose to their partners the maximal unavail-

ability u_n. This bounds the incentive for highly-available peers to manipulate the threshold. Alternatively, if both r and k is fixed (as in Proposition 11), an adopted peer j can expose to its parent its true unavailability $u_j(i) = u_j$. As the highly-available peers are processed in order of increasing u, this is sufficient to guarantee that the data unavailability increases with peer index.

In contrast, for the weakly-available peers, the mechanism must ensure that the resulting data unavailability increases with peer index; thus $u_i(j)$ are set so that $d_j \geq d_{j-1}$. Approximating d as in Definition 5, $u_i(j)$ are derived from $d_j = \max(d_{j-1}, u_j^{\sigma-r} \prod_{i \in S_j} u_i(j)$.

The following proposition shows that the algorithm is truthful if r and k are chosen independently of the distribution of unavailabilities u. For instance, the system might announce r and k, and then gather the peers' declarations of u.

Proposition 11. *Given r, k, the mechanism is truthful.*

Proof: We consider three cases: (i) a highly-available peer ($u_i < \tau$) declaring $\tau \geq u'_i > u_i$; (ii) a highly-available peer declaring $u'_i > \tau$; (iii) a weakly-available peer ($u_j > \tau$) declaring $u'_j > u_j$.

In the first case, $d_i(u'_i, u_{-i}) = (u'_i)^{\sigma-r} \prod_{j \in S'_i} u_j(i) > (u_i)^{\sigma-r} \prod_{j \in S_i} u_j(i)$, as $u'_i > u_i$ and a less-available peer can have less available partners assigned by the mechanism: as highly-available peers are processed by decreasing u_i, some peers from S_i might be not longer available and replaced by weaker peers in S'_i, thus $\prod_{j \in S'_i} u_j(i) \geq \prod_{j \in S_i} u_j(i)$.

In the second case, as $u'_i > \tau$, and the exposed unavailabilities are set so that d'_i is increasing, $d'_i \; d'_i \geq d_i$.

In the third case, by declaring $u'_j > u_j$, the peer can lower its ranking if and only if there exist a peer k with $u_j < u_k < u'_j$. As d is set to be increasing, $d'_j \geq d_k \geq d_j$. □

The following proposition shows that the reduction of the price of anarchy is proportional to the threshold τ^k.

Proposition 12. *The mechanism reduces the price of anarchy by at least $\tau^k / u_n^{\max(r,k)}$ if $r + k \leq \sigma$.*

Proof: The price of anarchy is determined by the data unavailability $d(n)$ of the least available peer n. Without the mechanism, the price of anarchy is bounded by $\left(\frac{u_n}{u_G}\right)^\sigma$ (in $\max d(G_k)$ aggregation, Proposition 8) and by $\left(\frac{u_n}{u_1}\right)^\sigma$ (in $\sum d(G_k)$ aggregation, Proposition 9).

After the mechanism completes, all weakly-available peers have k partners assigned with unavailability of at most τ. Thus, before limiting the exposed unavailabilities (the last step of the algorithm), the weakly-available peers had the data unavailability at most $d_j \leq u_j^{\sigma-\max(r,k)} \tau^k$. The least-available highly-available peer (with index $p = nk/(r+k)$) has its data availability of at least $d_p \leq \tau^{\sigma-r} u_n^r$. If $r + k \leq \sigma$, $\tau^{\sigma-r} u_n^r \leq \tau^k u_n^{\sigma-\max(r,k)}$ Thus, after limiting the exposed unavailability, the least available peer n has still the data unavailability $d_n \leq u_n^{\sigma-\max(r,k)} \tau^k$. □

We treat r as given, as it directly translates into burden for the highly-available peers. In contrast, k should be optimized by the algorithm. Given distribution of peers' unavailabilities u ($u(i) = u_i$), as the reduction in the price of anarchy is proportional to $\tau^k = (u(n\frac{k}{r+k}))^k$, the algorithm should choose an optimal k^* i.e., $k^* = \arg\min_{1 \leq k \leq \sigma-r} u(n\frac{k}{r+k})^k$.

However, as k^* depends on the declared unavailabilities u, some peers might be interested to declare $u'_i > u_i$ in order to manipulate this choice. Smaller values of k result in smaller thresholds τ, thus more peers are treated as weakly-available and have assigned partners. In general, for the same declared unavailability u_i, a peer i prefers to be treated as weakly-available than as highly-available.

We denote by p^* the peer that defines the threshold τ^* for the optimal k^* (assuming that every peer declares its true unavailability), i.e., $p^* = n\frac{k^*}{r+k^*}$; and by p' the peer that defines the threshold τ' for $k' < k^*$, i.e., $p' = n\frac{k'}{r+k'}$. Peers do not have incentive to manipulate the threshold so that $k' > k^*$, as peers that become highly-available $\tau^* \leq u_i \leq \tau'$ can be worse-off than before; and peers that are highly-available for both thresholds ($u_i \leq \tau^*$) do not gain from this change as the unavailability exposed by the weakly-available peers is u_n.

Assume that $p' < i \leq p^*$ (as other peers have no incentive to change the threshold). If k' is chosen, the peer i is treated as weakly-available as $\tau' < u_i$; its $d'_i = (u'_i)^{\sigma-k}(\tau')^k$. If k^* is chosen, the peer i is treated as highly-available; its $d_i = (u_i)^{\sigma-r} u_n^r$. Thus, it is possible that the peer improves its data availability by being treated as a weakly-available: for some k, r, τ', u_i, $d'_i < d_i$.

Given distribution of other peers' unavailabilities u_-, how can a peer i influence the mechanism so that k' rather than k^* is chosen? If i declares $u'_i = u(p^*+1)$, the p^*-th peer in the original distribution u (that determined the optimal threshold τ^*) becomes the (p^*-1)-th peer in the resulting distribution u'. Thus, the threshold is determined by the (p^*+1)-th peer in the original distribution u. (Note that declaring $u'_i > u(p^*+1)$ is not efficient for i, as it does not further influence the threshold, and, once the threshold is set, the peer is better-off declaring u'_i as low as possible by Proposition 11).

If declaring $u'_i = u(p^*+1)$ is sufficient for the mechanism to switch from k^* to k', how does it affect the reduction of the price of anarchy? As the mechanism switched from k^* to k' for $u = (u'_i, u_-)$, $u_{p'}^{k'} \leq u_{p^*+1}^{k^*}$. By dividing this inequality by $u_{p^*}^{k^*}$, we get $u_{p'}^{k'} / u_{p^*}^{k^*} \leq u_{p^*+1}^{k^*} / u_{p^*}^{k^*}$. Thus, the relative loss of the performance of the mechanism $u_{p'}^{k'} / u_{p^*}^{k^*}$ is at most equal to the relative difference of unavailabilities of two consecutive peers $(u_{p^*+1}/u_{p^*})^{k^*}$.

Of course, if $k' > 1$, other peers might have incentive to lower the threshold even further. However, there might be at most k^* such actions. We denote by p_l the peer defining threshold τ_l ($1 \leq l \leq k^*$), the cumulative impact on the mechanism is at most $\prod_{1 \leq l \leq k^*} (u_{pl+1}/u_{pl})^{k^*} \leq (\max_{1 \leq l \leq k^*} (u_{pl+1}/u_{pl})^{k^* \cdot k^*}$

$t_{\min} = 0; t_{\max} = 1$
while $t_{\max} - t_{\min} > \epsilon$ **do**
  $t = (t_{\max} - t_{min})/2$
  for $k = 1$ to r **do**
    $\tau[k] = t^{1/(\sigma-k)}$
  for $i = n$ downto 1 **do**
    $k = 1; S_i = \emptyset$
    while $u_i < \tau[k]$ **do**
      $k{+}{+}$
    $r_i = k - 1$
    if $r_i = 0$ **then**
      $p = i$
  for $j = n$ downto p **do** ▷ p denotes the most-available child
    $d = u_j^\sigma$; failed=false
    while $d > t$ and not failed **do**
      failed=true
      for $i = p - 1$ downto 1 **do**
        if $j \notin S_i$ and $|S_i| < r_i$ and $d \cdot u_i/u_j < t$ **then**
          $S_j := S_j \cup \{i\}; S_i := S_i \cup \{j\}; d := d \cdot u_i/u_j$
          failed=false; break
      if $d > t$ **then**
        for $i = 1$ to p **do**
          if $j \notin S_i$ and $|S_i| < r_i$ **then**
            $S_j := S_j \cup \{i\}; S_i := S_i \cup \{j\}; d := d \cdot u_i/u_j$
            failed=false; break
    if failed **then**
      break
    $p_m = \min\{i \in S_j\}; d = d/u_{pm}$
    for $i = p - 1$ to p_m **do**
      if $j \notin S_i$ and $|S_i| < r$ and $d \cdot u_i < t$ **then**
        $S_j := S_j \cup \{i\} - \{p_m\}; S_i := S_i \cup \{j\}$
        $S_{pm} := S_{pm} - \{j\}$
  if failed **then** $t_{\max} := t$ **else** $t_{\min} := t$

Algorithm 1: A heuristic adoption mechanism.

As the distribution of peers' unavailabilities u is measured, we may assume that it is "dense" in a sense that the relative difference between two consecutive peers is small. Consequently, even if a mechanism optimizing k is not strictly truthful, the possible manipulations are small; and its result does not significantly impact the price of anarchy. Moreover, the game considered in this section is a worst-case approximation of what happens in a real system. In the real system, the unavailability u_i is measured, and not declared by the peers. The impact of the measurement tool can be modelled as a random noise added to the unavailability declared by a peer; thus, the impact of slight, strategical increase of unavailability (as analyzed in this section), can be cancelled by a random noise; and, in general, a significant increase of unavailability is not profitable by Proposition 11.

6.3 A Heuristic Adoption Mechanism

The mechanism proposed in this section does not waste the replication slots compared to the theoretical mechanism presented previously. However, because of the complex graph of replication agreements, we will not derive a closed formula for the reduction of the price of anarchy.

The mechanism is still truthful in the sense of Proposition 11, i.e., once the parameters are set, the algorithm guarantees that the data unavailability is increasing with peers' unavailabilities.

function $score_c$(peer i, peer j, $G_k : i \in G_k$, $G_l : j \in G_l$)
  if $|G_k| < \sigma$ **then**
    if $|G_k| + |G_l| \leq \sigma$ **then**
      return $|av_i - av_j|$
    else if $|G_l| < \sigma$ **then**
      return $0.5 \cdot |av_i - av_j|$
    else
      return 0
  else
    $u_k^H = \max_{i' \in G_k} u_{i'}$; $u_k^L = \min_{i' \in G_k} u_{i'}$
    $u_l^H = \max_{j' \in G_l} u_{j'}$; $u_l^L = \min_{j' \in G_l} u_{j'}$
    if $u_l^H < u_k^L$ or ($|G_l| = \sigma$ and $u_l^L > u_k^H$) **then**
      return 0
    else
      return $\max(u_k^H, u_l^H) - \min(u_k^L, u_l^L)$

Algorithm 2: Peer i computes the score of peer j using Explicit Cliques.

Also, similarly to the previous section, the algorithm takes as a parameter the maximum number of slots r it can use at each peer. Unlike the previous algorithm, however, it can use less than r slots.

The algorithm (see Algorithm 1) does a binary search for the optimal target unavailability t^*. For each tested value of t, the algorithm tries to organize the replication agreements so that all the peers have data unavailability at most t, $d(i) \leq t$. If the assignment fails, it means that the tested value was too low; if it succeeds, it tries a lower t.

Given t, the algorithm sets r boundary thresholds τ_k to $\tau_k = t^{1/(\sigma-k)}$ ($k \in \{1, \ldots, r\}$). Peers with $\tau_{k+1} < u_i \leq \tau_k$ contribute exactly k slots to adopt weakly-available peers.

Then, weakly-available peers ($u_j > \tau_1$) are processed in order of increasing availabilities (from the least available to the most available). For each peer j, the algorithm greedily assigns highly-available replication partners to achieve $d_j \leq t$ in the following loop. If there is a single highly-available peer i sufficient to achieve $d_j \leq t$, then the worst-possible i is assigned and the loop breaks. Otherwise, the highest currently available i is assigned to j and the loop continues. If all highly-available peers have all their adoption slots assigned, the subroutine exits with failure (t was too low). After assigning replication partners for j, the algorithm tries to greedily swap j's best replication partner $p_m = \min\{i \in S_j\}$ with a weaker-available peer i. i is chosen so that if p_m is replaced in the replication group by i, j's data unavailability is still below the threshold t, $p' = \max\left\{p : d_j(S_j - \{p_m\} \cup \{p\}) \leq t\right\}$.

If the algorithm succeeds to assign replicators to all weakly-available peers, the exposed unavailabilities $u_i(j)$ are bounded in order to keep d_i increasing with i (in the same way as in Section 6.2; this last step is not shown in Algorithm 1). Similarly to Proposition 11, by the definition of the truthful mechanism 5, this final step makes the mechanism truthful.

7 A Heuristics for the Decentralized Matching

The following algorithm creates an environment similar to the Stochastic Replication Game (Definition 3). The main goal of the algorithm is to reduce the time needed to reach the subgame-perfect equilibrium (Proposition 7) in the context of a real distributed system, that, through limited bandwidth and peers' processing power, limits the number of replication proposals that each peer can make. Among replication candidates (most of whom are unknown due to the distributed nature of the system), the algorithm helps peers find and choose partners that not only maximize data availability, but also are not likely to withdraw replication agreements—thus, the partners from the equilibrium. At the same time, all the decisions imposed by the algorithm are rational (never decrease the peer's data availability), thus the algorithm converges to cliques defined in the subgame perfect equilibrium. Consequently, even if some of the peers choose not to follow the algorithm (but still are rational), the steady state will be the same—the equilibrium—but reached more slowly (or faster, if the aberrants use, e.g., an oracle).

To illustrate this difference, consider a peer with a medium availability. To maximize her data availability, the peer should try to form a replication agreement with a peer with high availability. However, such a highly available peer is likely to already have (or have in the near future) highly available replication partners; thus the replication request from the "mediocre" peer will be either rejected, or withdrawn soon.

Each peer maintains a list of candidates for replicas. This list is refreshed by the T-Man [27] gossiping protocol. Each peer i has two pools of peers: a random pool $rand(i)$—at most s_r peers forming a sample of the population; and a *metric* pool $metric(i)$ with s_m peers that score well according to a local metric. In an iteration of T-Man, each peer updates its random pool by gossiping with a randomly-chosen peer from this pool. During this operation, to form the new random pool, each peer chooses s_r most recently added peers from both random pools. After modifying the random pool, the metric pool is updated as the best s_m peers from the current metric pool and the current random pool. Then, the peer communicates with the best peer from its metric pool: the metric pools are exchanged, merged, scored and then each peer chooses the best s_m peers from the merged pools.

To form replication agreements, peers use heuristics to compare the current replicas with the candidates. We first describe a framework, then several possible heuristics to choose candidates.

In each turn, each peer i scores the peers in its metric pool that are not i's current replicas nor on its taboo list. If i has less replicas than its maximum capacity, it proceeds to querying the peer j^* with the highest score. Otherwise, candidates are compared with i's worst current replica k ($k = \arg\max_{l:\, replica(i,l)} u_k$). i queries the first candidate j^* (in order of non-decreasing score) better than k (for which $u_j < u_k$). If there is no such candidate, i does not switch replicas.

The queried candidate j^* decides whether to accept the mutual replication: if it has less replicas than its maximum capacity, i is accepted. Otherwise, i is accepted only if $u_i < u_{k'}$, where k' is j^* worst replica.

Finally, if j^* accepts i, and i already has as many replicas as its maximum capacity, i drops its current worst replica k.

In order not to repetitively query the same peers, each peer i maintains a taboo list, consisting of former replicas that have been dropped or that dropped i; and of peers that did not accept replication with i.

We used three variants of the above algorithm that differ in the trade-off between short-sighted selfishness and the speed of convergence. In *Optimistic Queries*, candidate j's score is equal to its availability $score_o(i,j) = a_j$. In *Pragmatic Queries*, candidate j's score is equal to the absolute difference between its availability and the availability of the assessing peer i, $score_P(i,j) = 1 - |av_i - av_j|$.

Finally, *Explicit Cliques* maintains cliques composed of one or more peers. Every member of a clique replicates data of all other members. Thus, a representative i of a clique G_k, after choosing peer j^* with the maximum score $score_C(i,j)$, tries to merge its clique with the clique G_l: $j^* \in G_l$ of the chosen candidate. The two cliques exchange members: the "better" clique groups σ peers with the highest availability (or the two cliques combined, if the combined clique has at most σ members); the "worse" clique groups remaining members of both cliques.

The scoring function $score_C(i,j)$ depends on the size of the cliques $i \in G_k$ and $j \in G_l$ (see Algorithm 2).

If $|G_k| < \sigma$, G_k is not complete and the algorithm should increase its size (as it considerably reduces $d(G_k)$). It is best to have G_l such that the two cliques will be merged into one, $|G_k| + |G_l| \leq \sigma$. Thus, $score_C(i,j) = score_P(i,j)$ in this case; otherwise, if $|G_l| < \sigma$, $score_C(i,j) = 0.5 score_P(i,j)$; finally if $|G_l| = \sigma$, $score_C(i,j) = 0$ (as in this case one of the cliques after merging will still be of size $|G_k|$).

If $|G_k| = \sigma$, the goal is to find a clique G_l that after merging will reduce the variance of availabilities of peers in both cliques. During merging of G_k with G_l, the two cliques will exchange members (and thus reduce the variance) if: (i) the intersection of availability ranges is not empty, $[u_k^L, u_k^H] \cap [u_l^L, u_l^H] \neq \emptyset$; or (ii) $|G_l| < \sigma$ and $u_l^H > u_k^L$. In these two cases, the score is equal to the range of unavailability that will be reduced; otherwise $score_C(i,j) = 0$.

We compare these three algorithms on randomly-generated instances in Section 8.4.

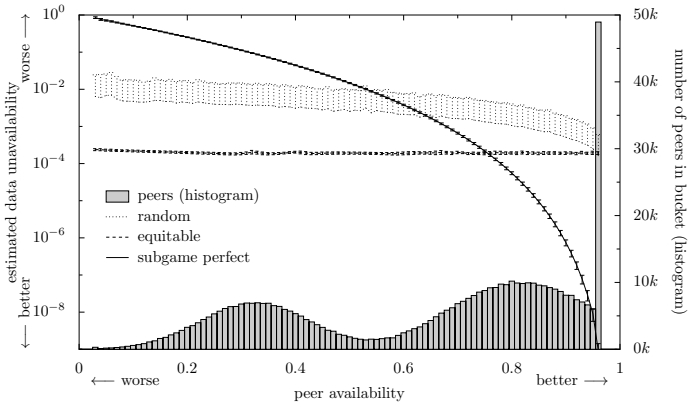

Fig. 1. Peers' expected data unavailability as a function of their availability in random, equitable and subgame perfect assignment. Histogram shows the number of peers in each availability bucket.

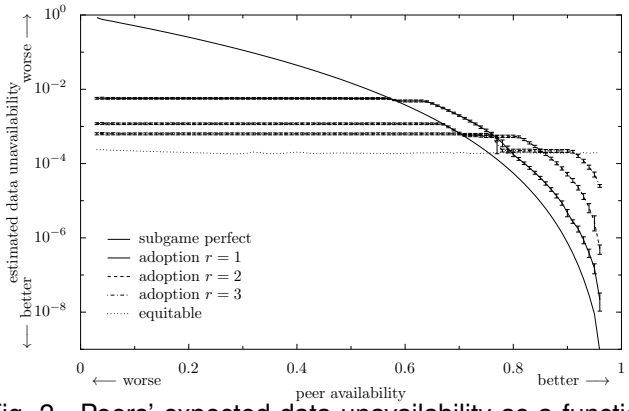

Fig. 2. Peers' expected data unavailability as a function of their availability in adoption-based mechanism for different number of adoption slots r. Subgame-perfect and equitable assignments given as a reference.

8 Simulation of the Algorithms

8.1 Simulation Settings

Peers' availabilities were generated in three steps. Firstly, following [7], 10% of the peers have availability 0.95, 25%—0.87, 30%—0.75 and 35%—0.33. Then, we added a Gaussian noise with $\sigma = 0.1$ to each availability. Finally, we capped the resulting value, so that $0.03 \leq a_i \leq 0.97$. Histogram on Figure 1 shows the resulting distribution of peers. We repeated each experiment on 50 instances with peers' availabilities generated as described above; error bars on plots denote standard deviations.

We set the storage size $s = 5$ and the sizes of random and metric pools in T-Man gossiping to 50.

We implemented *decentralized* algorithms in a custom discrete event simulator. In each round of the simulated matching, all the peers are processed sequentially in random order. Each peer performs one iteration of T-Man gossiping, and then one iteration of the decentralized matching algorithm.

8.2 Centralized Allocation: Subgame Perfect vs Equitable Solutions

We started with comparing *random*, *subgame perfect* and *equitable* allocation algorithms according to the resulting data unavailability. We ran these algorithms on instances of 10.000 peers each; then we computed averages over all the random instances and all peers having similar availabilities (with resolution equal to two decimal places, e.g., the score for 0.95 is an average for all peers with $0.95 \leq a_i < 0.96$). Figure 1 summarizes the obtained results.

The equitable algorithm produces cliques that result in similar data availability regardless of the peer's availability. In contrast, the subgame perfect equilibrium results in wide range of data availabilities: while the highly available peers have their data available with expected failure probability of approximately 10^{-9}, the weakest available peers almost do not gain from replication, with data unavailability close to 1.

Such diversification in the subgame perfect solution provides incentives for peers to be highly available. A highly available peer is able to replicate its data with other highly available peers, which exponentially increases peer's data availability. However, the subgame perfect solution might be too "extreme" to the less-available peers. Peers with availabilities less than approximately 0.5 have their data available with probability less than 0.99 (approximately), which might be not sufficient for some applications. This, in turn, can discourage such peers from joining the system, and consequently, prohibit the system from growing to a critical mass.

On the other hand, an equitable solution does not reward highly available peers. In absence of altruistic peers, the system would degenerate.

Also note that the equitable solution clearly Pareto-dominates the random assignment, resulting in higher data availabilities for all classes of peers.

8.3 Centralized Allocation: Increasing Availability by Adoption

We simulated the influence of the availability-encouraging adoption (Section 6) on the data availability. We varied the number of adoption slots $r \in \{1, 2, 3\}$ and, for each r and each distribution of peers' u_i, performed the heuristic algorithm from Section 6.3. The remaining slots were assigned by a subgame-perfect solution. The rest of the settings for the experiment was the same as in the last experiment. Figure 2 shows the results.

The adoption mechanism is efficient in increasing the data availability d of the weakest-available peers. Even with one slot donated by the highly-available peers ($r = 1$), the data availability d of the 20 weakest-available peers is improved by, on the average, two orders of magnitude (128.4 times, std.dev. 7.9). Peers with availabilities smaller than 0.6 have greater data availability than in the selfish solution. All weakly available peers have similar data availability, thus the mechanism does not "reward" peers for increasing availability up to,

approximately, 0.65. The mechanism is not, however, discouraging peers to declare their true availability, as the curve is non-increasing. If one wants to emphasize rewards, the mechanism can be easily modified to decrease the data availability with decreasing availability: the target unavailability t should be decreased with peer index. The impact of the mechanism on highly-available peers is limited: their data availability is decreased by about two orders of magnitude (corresponding to approximately one replication slot "wasted" by the mechanism on these peers). This behavior is caused by the fact that the highest-available peers are assigned the weakest-available partners.

Donating more replication slots results in even higher gains for the weakly-available peers. For $r = 2$, the data availability is improved on the average 618 times (std.dev. 39); for $r = 3$, by more than three orders of magnitude (1159 times, std.dev. 79).

The threshold τ found by the algorithm decreases with increasing r. For $r = 1$, $\tau_1 = 0.35$; $r = 2$, $\tau_1 = 0.26$, $r = 3$, $\tau_1 = 0.23$ (std. dev. less than 0.005; note that τ denotes the *unavailability* of the weakest parent). For $r = 1$, approximately 40% of the population is treated as weakly-available; this share grows up to 57% for $r = 3$.

8.4 Decentralized Allocation: Speed of Convergence

In the next series of experiments, we measure how fast do the decentralized algorithms presented in Section 7 converge to the subgame perfect cliques.

All algorithms use a taboo list not to repeatedly try a replication agreement with the same replication partners. *Optimistic Queries* should not delete a peer from a taboo list, because it is almost certain that the peer that refused a replication agreement will not change its decision. However, *Pragmatic Queries* and *Explicit Cliques* might gain from limiting the number of peers on the taboo list (to e.g. $2s$, as the optimal replication agreements are formed by s among $2s$ neighbors). In the initial experiments, we checked that such a limit does not influence the convergence speed of *Explicit Cliques*; and that *Pragmatic Queries* converges slower when the size of the taboo list is limited. If a list is unlimited, there is a small risk that an optimal replication partner is added to the taboo list; but, according to our experiments, this risk is outweighed by the gain a peer gets from reducing reduntant queries.

Initial experiments revealed that the *Optimistic Queries* version of the algorithm is inefficient. After the first few rounds when the underlying gossiping protocol efficiently fills the metric pools of all peers with the same set of 50 highest available peers, in the subsequent rounds the whole population queries the best peer, the second-best peer, and so on. Thus, replication agreements are formed extremely slowly. We observe that if peers' availabilities are distinct, approximately k/σ cliques are formed after approximately k rounds.

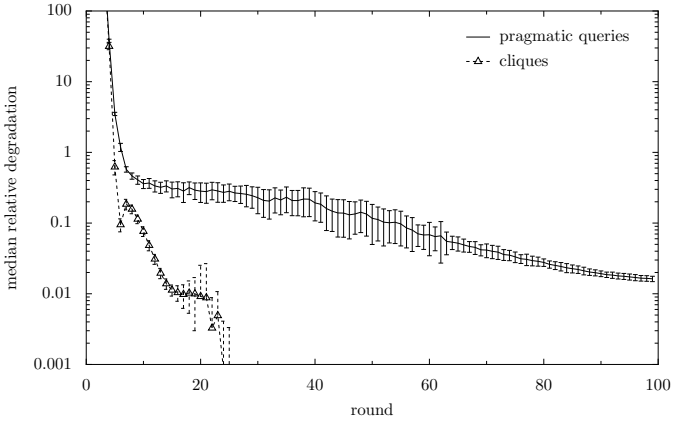

Fig. 3. Convergence speed of decentralized algorithms to subgame perfect cliques. Y axis is the median (computed over only non-zero elements) of relative degradation vs. the subgame perfect solution. $n = 2000$

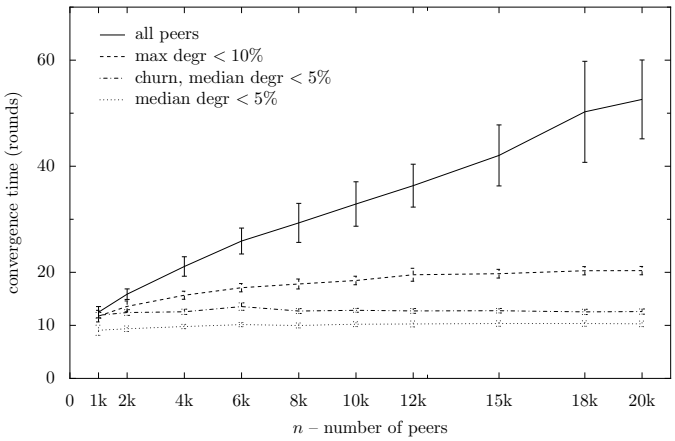

Fig. 4. Convergence speed of the clique-based algorithm to the subgame perfect cliques as a function of the number of peers in the system. The solid line denotes the number of rounds needed to reduce the degradation of all peers to less than 10^{-9}; the dashed—to reduce the maximum relative degradation to less than 10%; the dotted—to reduce the median degradation to less than 5%.

Figure 3 compares the convergence speed of *Pragmatic Queries* to *Explicit Cliques*, measured as the median average degradation vs. the subgame perfect solution (we treated differences less than 10^{-9} as zero). In this experiments, we excluded peers with boundary availabilities (0.97 and 0.03) from the results.

Explicit Cliques converges much faster (in about 25 rounds), by, firstly, quickly building as many full cliques as possible, and then optimizing their contents. Note that the high standard deviation observed in rounds 18-25 is an artifact of computing the median from only few values.

In contrast, *Pragmatic Queries* form two chains of peers (grouping highly available peers in one and weakly available peers in the other). The chains are formed because each peer i replicates with s closest neighbors according to the absolute value of the difference in availabilities: $s/2$ peers with availabilities higher than i and $s/2$ peers with lower availabilities. Only the $s/2$ most-

available peers, lacking even higher available peers, form agreements with worse peers. In subsequent rounds, these worse peers gradually drop their chain neighbors in favor of higher available peers; thus, the dropped neighbors no longer have higher available neighbors, and the phenomenon propagates towards the next peers.

Figure 4 shows the speed of convergence of *Explicit Cliques* as a function of number of peers. The algorithm reduces the median degradation to less than 5% in about 10 rounds; that result does not depend on the size of the population. Even for larger populations, it takes just around 20 to reduce the maximum degradation to less than 10%. In contrast, the number of rounds for the absolute convergence depends on the size of the population: the mean number of rounds rises from 12.5 (for 1,000 peers) to 52.6 (for 20,000 peers). For the largest population, the algorithm converged at most in 71 rounds.

We also simulated the impact of temporal failures when the algorithm is running. To simulate the worst-case behavior, we assumed that once a peer leaves, it does not return to the system; and there are no new peers joining. In order to model the probabilistic process of leaves, we needed to characterize the duration of the distributed algorithm. Each peer during one round of the algorithm contacts a single peer from its random pool and a single peer from its metric pool; as the amount of changed information is negligible, we may bound the length of that phase by two maximal round trip times (RTTs) in the network. We upper-bounded the duration of a round to 1 second. The process of peers' leaves depends directly on expected sessions' durations. A P2P storage system would typically have long sessions; to simulate the worst-case behavior, we assumed that the mean session length is just 1,000 seconds (roughly 17 minutes). This leads to a probabilistic model in which in each round each peer has a probability of $1/1000$ of leaving the system.

We measured the number of rounds needed to reduce the median degradation to less than 5%. Certainly, measuring the absolute convergence does not make sense, as if a peer leaves, its replication partners with higher availability would have their data availability degraded (as they need to choose worse replication partners). The results are summarized by a dash-dotted line on Figure 4. Compared to a no-churn system, it takes 2-3 rounds more for the system to converge. The algorithm is thus robust in the presence of high churn.

9 Discussion and Conclusions

We studied the problem of replica placement in a p2p storage system in order to optimize availability and/or the number of replicas. We argued that replication should be based on cliques of peers replicating each others' data, rather than on a directory or bilateral assignments. We analyzed an idealistic model of peer availability that focuses on uncertainty of peer's online status. We proved that it is NP-hard to optimize availability for the *socially-equitable* scheme (in which the data availability of all peers is similar). We also analyzed a game theoretic version of the problem in which peers form bilateral replication agreements. We demonstrated that the loss in the global efficiency compared to the socially-optimal solution (the *price of anarchy*) is arbitrarily large.

In order to reduce the price of anarchy, we proposed a semi-centralized "adoption" mechanism: highly-available peers donate some of their replication slots. The algorithm uses these slots to replicate data of weakly-available peers. We formulated rules that guarantee that this algorithm is a truthful mechanism: no peer would obtain higher data availability by lowering its declared availability.

We proposed heuristics for centralized and decentralized matching that perform well in simulation experiments. In particular, the "adoption" heuristics increases the data availability of the weaker peers by two to three orders of magnitude, which should satisfy basic storage services.

Our results have quite a few more practical consequences for p2p storage or replication systems. Most importantly, in such systems, if allowed to choose partners by themselves, highly available peers will tend to replicate data among each other, and to exclude peers with low availability. This could result in unacceptable performance for peers with lower availabilities (in our experiments, less than about 30%). While this phenomenon provides an incentive for peers to be highly available, it can be also discouraging for the newcomers to join, and thus—hard for the system to gain momentum and large scale.

The system thus needs a method of availability redistribution — for instance, the proposed adoption algorithm. However, such methods have a problem analogous to any taxation and social security system. While we have proved that the mechanism is truthful, we did it on a model assuming that all peers would participate. Obviously, for highly-available peers, not donating slots for adoption (not paying taxes) dominates any adoption (paying taxes, or any redistribution mechanism). In real-world societies, this problem is solved by tax law enforcement; in p2p storage, a combination of a reputation system and common sense must be sufficient.

In this paper, we deliberately focused on a single issue, replica placement, which allowed us to derive mathematical, as well as simulation results. We left unaddressed other problems like maintenance [28], [29], redundancy schemes [30], or heterogeneity of peers' storage needs and capabilities.

Storage heterogeneity would only slightly alter the resulting equilibrium. A peer with higher storage space would be able to form more "unit" replication agreements (and thus increase her data availability), although with peers having lower availabilities (or with peers with higher storage space and higher availabilities). A peer having more data to store would need to surrender

storing complete copies on all the replicas which would result in lower data availability.

In the theoretical analysis, we considered only availabilities constant in time. Our earlier work [9] proposed and analyzed also another model, in which availabilities change in time (which models e.g. daily usage patterns of a computer). Our initial results suggested that time-based analysis improves availability only when a system has a truly global scope: downtimes of peers from one continent (e.g. Europe) can be complemented by others (e.g. Asia). In our future work, we plan to extend this analysis with a more formal approach.

Our model had constant population of peers (although we assessed the impact of temporary failures on our distributed algorithm). However, it can be easily extended to realistic systems with new peers joining and existing peers permanently leaving the system. Storage contracts should be signed for a pre-determined time period (e.g., 1 week). In the background, to replace the expiring contracts and to handle changes in partners' availability estimates, peers should run the distributed matching algorithm (that will determine the "next" assignment). Of course, a contract would be replaced only if the gain in the availability balances the bandwidth needed to transfer the data. Such a system naturally responds to peers permanently leaving the system, as their availability would gradually diminish. Newcomers should enter the system with zero availability (otherwise, a Sybil attack would be possible). The newcomers will be adopted by highly-available peers: the newcomers will gradually replace peers that no longer need to be adopted; also, some of the adoption slots can be reserved, so that a newcomer is immediately assigned to a highly-available peer.

Currently, we are working on an implementation of a peer-to-peer storage system with reciprocal storage contracts. The principal design goal is to provide a storage layer for a distributed on-line social networking software, such as PeerSoN [1].

REFERENCES

[1] S. Buchegger, D. Schiöberg, L. Vu, and A. Datta, "PeerSoN: P2P social networking: early experiences and insights," in *ACM SNS, Proc.*, 2009.

[2] M. Feldman, K. Lai, J. Chuang, and I. Stoica, "Quantifying disincentives in peer-to-peer networks," in *P2P Econ, Proc.*, 2003.

[3] R. Sharma, A. Datta, M. DeH'Amico, and P. Michiardi, "An empirical study of availability in friend-to-friend storage systems," in *P2P, Proc.* IEEE, 2011, pp. 348–351.

[4] M. Babaioff, J. Chuang, and M. Feldman, *Algorithmic Game Theory*. Cambridge, 2007, ch. Incentives in Peer-to-Peer Systems, pp. 593–611.

[5] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," in *IPTPS, Proc.*, ser. LNCS, vol. 2735. Springer, 2003, pp. 256–267.

[6] J. Douceur and R. Wattenhofer, "Competitive hill-climbing strategies for replica placement in a distributed file system," in *DISC, Proc.*, ser. LNCS, vol. 2180. Springer, 2001, pp. 48–62.

[7] S. Bernard and F. Le Fessant, "Optimizing peer-to-peer backup using lifetime estimations," in *Damap Proc.*, 2009.

[8] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, "Total recall: System support for automated availability management," in *NSDI, Proc.*, 2004.

[9] K. Rzadca, A. Datta, and S. Buchegger, "Replica placement in p2p storage: Complexity and game theoretic analyses," in *ICDCS 2010, The 30th International Conference on Distributed Computing Systems, Proceedings*. IEEE Computer Society, 2010, pp. 599–609.

[10] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *STACS, Proc.*, ser. LNCS, no. 1563. Springer, 1999, pp. 404–413.

[11] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware Proc.*, ser. LNCS, vol. 2218. Springer, 2001, pp. 329–350.

[12] A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, and S. Shenker, "On a network creation game," in *PODC, Proc.* ACM, 2003, pp. 347–351.

[13] T. Moscibroda, S. Schmid, and R. Wattenhofer, "On the topologies formed by selfish peers," in *PODCS, Proc.* ACM, 2006, pp. 133–142.

[14] R. Rahman, T. Vinkó, D. Hales, J. Pouwelse, and H. Sips, "Design space analysis for modeling incentives in distributed systems," in *ACM SIGCOMM, Proceedings*, 2011.

[15] R. Bhagwan, S. Savage, and G. Voelker, "Replication strategies for highly available peer-to-peer storage systems," *Fu-DiCo, Proc.*, 2002.

[16] L. Cox, C. Murray, and B. Noble, "Pastiche: Making backup cheap and easy," *ACM Operating Systems Rev.*, vol. 36, pp. 285–298, 2002.

[17] P. Michiardi and L. Toka, "Selfish neighbor selection in peer-to-peer backup and storage applications," in *Euro-Par, Proc.*, ser. LNCS, vol. 5704, 2009.

[18] L. Toka and P. Michiardi, "Analysis of user-driven peer selection in peer-to-peer backup andstorage systems," *Telecommunication Systems*, vol. 47, pp. 49–63, 2011.

[19] F. Giroire, J. Monteiro, and S. Pérennes, "P2P storage systems: How much locality can they tolerate?" in *LCN, Proc.* IEEE, 2009, pp. 320–323.

[20] L. Pamies-Juarez, P. Garcia-Lopez, and M. Sanchez-Artigas, "Enforcing fairness in p2p storage systems using asymmetric reciprocal exchanges," in *P2P, Proc.* IEEE, 2011, pp. 122–131.

[21] J. Drèze and J. Greenberg, "Hedonic coalitions: Optimality and stability," *Econometrica: Journal of the Econometric Society*, pp. 987–1003, 1980.

[22] A. Bogomolnaia and M. Jackson, "The stability of hedonic coalition structures," *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[23] F. Le Fessant, C. Sengul, and A. Kermarrec, "Pace-maker: Tracking peer availability in large networks," INRIA, Tech. Rep. RR-6594, 2008.

[24] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and approximation*. Springer, 1999.

[25] S. Assmann, D. Johnson, D. Kleitman, and J. Leung, "On a dual version of the one-dimensional bin packing problem," *Journal of algorithms*, vol. 5, no. 4, pp. 502–525, 1984.

[26] M. Osborne, *An introduction to game theory*. Oxford University Press, 2004.

[27] M. Jelasity and O. Babaoglu, "T-man: Gossip-based overlay topology management," in *ESOA, Proc.*, ser. LNCS, vol. 3910. Springer, 2006.

[28] B. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," in *NSDI, Proc.*, vol. 6, 2006.

[29] Z. Yang, J. Tian, B. Zhao, W. Chen, and Y. Dai, "Protector: A probabilistic failure detector for cost-effective peer-to-peer storage," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 9, pp. 1514–1527, 2011.

[30] R. Rodrigues and B. Liskov, "High availability in DHTs: Erasure coding vs. replication," in *IPTPS, Proc.*, ser. LNCS, vol. 3640. Springer, 2005.