# Replica Placement in P2P Storage: Complexity and Game Theoretic Analyses

Krzysztof Rzadca
School of Computer Engineering
Nanyang Technological University
Singapore
Email: krz@ntu.edu.sg

Anwitaman Datta
School of Computer Engineering
Nanyang Technological University
Singapore
Email: anwitaman@ntu.edu.sg

Sonja Buchegger
School of Computer Science
KTH
Sweden
Email: buc@kth.se

***Abstract*—In peer-to-peer storage systems, peers replicate each others' data in order to increase availability. If the matching is done centrally, the algorithm can optimize data availability in an equitable manner for all participants. However, if matching is decentralized, the peers' selfishness can greatly alter the results, leading to performance inequities that can render the system unreliable and thus ultimately unusable.**

**We analyze the problem using both theoretical approaches (complexity analysis for the centralized system, game theory for the decentralized one) and simulation. We prove that the problem of optimizing availability in a centralized system is NP-hard. In decentralized settings, we show that the rational behavior of selfish peers will be to replicate only with similarly-available peers. Compared to the socially-optimal solution, highly available peers have their data availability increased at the expense of decreased data availability for less available peers. The price of anarchy is high: unbounded in one model, and linear with the number of time slots in the second model.**

**We also propose centralized and decentralized heuristics that, according to our experiments, converge fast in the average case.**

**The high price of anarchy means that a completely decentralized system could be too *hostile* for peers with low availability, who could never achieve satisfying replication parameters. Moreover, we experimentally show that even explicit consideration and exploitation of diurnal patterns of peer availability has a small effect on the data availability—except when the system has truly global scope. Yet a fully centralized system is infeasible, not only because of problems in information gathering, but also the complexity of optimizing availability. The solution to this dilemma is to create system-wide cooperation rules that allow a decentralized algorithm, but also limit the selfishness of the participants.**

***Index Terms*—price of anarchy, equitable optimization, distributed storage**

## I. Introduction

A decentralized system for data storage and replication is an important building block of many peer-to-peer (p2p) applications, such as backup (e.g., wuala.com), or social networks [1] (in which, when a user is off-line, the system ensures that her data is available for her friends). 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. As users in p2p systems are assumed to be independent [3], [4], 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 an user must take 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 (like in wuala.com), or using a fully decentralized algorithm in which users form replication agreements [5], [6].

In this paper, we study the problem of maximization of *data availability* in a decentralized data replication system. In order to obtain worst-case bounds in these complex systems, we model what we consider the crucial characteristics of the problem along two axes: (1) peer availability (deterministic time slots or probabilistic); (2) matching (centralized and enforced or decentralized and autonomous).

In the *probabilistic* model, a peer's availability is the probability of the peer being available (correlated with the peer's expected lifetime, like in [7], [6]). The goal is to maximize data availability given the constraints on the storage size. In contrast, in the *time slot* (deterministic) model peer availability is a function of time, either in a periodic way [8] (also observed for the whole system in [9]), or according to a detailed prediction for the next time period. In this model, availability is a set of time slots in which the peer is available with *certainty*. The goal is to minimize the number of replicas such that the sum of their availability periods covers the whole prediction time.

We analyze both availability models 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.

In centralized systems with global knowledge, we prove that achieving equitable allocation in both probabilistic (Section III-A) and time slot (Section IV-A) models is NP-complete. We also propose heuristics (Section V) to achieve equitable availability in the probabilistic model. According to

The work in this paper has been funded in part by NTU/MoE's AcRF Tier-1 RG 29/09 and A*Star SERC 072 134 0055 grants.

our experimental evaluation (Section VII-A2), this heuristic achieves data availability one to two orders of magnitude better than the random allocation for all classes of peers.

In the decentralized, probabilistic model, we prove the existence of a unique subgame perfect equilibrium (Section III-B), 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 (Section VI-A) that according to our experimental evaluation (Section VII-A3) efficiently converges to the subgame perfect solution.

Finally, in the decentralized, time slot model, we prove that, even if a peer has complete knowledge about others' availabilities and every other peer is willing to replicate, constructing the replication set with minimal size is NP-hard. The price of anarchy is at least linear with the number of time slots (Section IV-B). We also propose a distributed heuristic to form replication cliques (Section VI-B). Using the heuristic, we experimentally show that data availabilities in the time slot model are higher than in the probabilistic model only if disparity of peers' availability slots is high (Section VII-B).

## II. 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, while themselves need to store data of one unit. We also assume that peers replicate data by storing complete copies, in contrast to erasure codes. Although erasure codes are more efficient for some applications, they have other drawbacks like increased maintenance costs and system complexity [11]. This is anyway somewhat orthogonal to the results discussed here.

Peer $i$'s *availability* $av(i,t)$ is the probability of being online at time $t$. Peer's *unavailability* representing the probability that the peer $i$ is not available at time $t$ is given by $nav(i,t) \triangleq 1 - av(i,t)$. We assume that the availability is known and cannot be tampered with by peers, e.g., using [12].

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 $dnav(i,t)$ equal to the probability of the event that all the replicators are offline, $dnav(i,t) = \prod_{j=1\ldots k} nav(i_j,t)$ (assuming that the peers' failures are independent [3], [4]).

We also assume that peers form replication *groups* (*cliques*) $\{G_k\}$. Each member $i \in G_k$ of the group replicates data of all other members $j \neq i$: $j \in G_k$. We define group unavailability $nav(G_k,t) \triangleq \prod_{i \in G_k} nav(i,t)$. Note that, $\forall i \in G_k$: $dnav(i,t) = nav(G_k,t)$. Such groups have several advantages compared with the pair-based allocation. Firstly, groups are naturally formed in the subgame perfect solution of some of the decentralized versions of the problem (see Proposition 4 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 $dnav(i)$ by choosing, or forming, a group with $nav(G_k,t)$ 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 $G_k = \{i_2, \ldots, i_s\}$ such that $nav(G_k,t)$ is minimized;
2) given the maximal tolerable unavailability $\epsilon$, minimize the size of the replicating group $\min s = |G_k|$.

Both types of goals can be expressed also from the system's perspective. The system should guarantee that peers are treated fairly [13] by optimizing *all* groups' unavailabilities $nav(G_k,t)$. In this paper, instead of a multi-objective approach, we will aggregate the groups' goals using two aggregating functions: (1) $\sum nav(G_k,t)$ (often used in other domains, but not a fair aggregation in the sense of [13]); (2) $\min\max nav(G_k,t)$ (which can be inefficient for the system, by focusing on the worst-off group).

## III. Complexity and Worst-Case Bounds in the Probabilistic Model

In this section, we assume that peers' availabilities are constant in time. Thus, for each peer $p_i$, $av(i,t) = av(i)$. Such a model is commonly used in the literature (e.g., [6], [7]). First we prove that even for a centralized matching (applicable in systems like wuala.com), equitable optimization of the availability is NP-hard. Then, in a decentralized matching, we show that the subgame perfect equilibrium of a game with selfish peers can be arbitrarily far from the equitably-optimal solution. We propose heuristics to optimize allocation in centralized and decentralized systems in Sections V and VI-A.

### A. Complexity of Centralized Matching

The task of the fair matching system is to divide peers into replicating groups such that: (1) the probability of data being online is as high as possible; (2) the size of each group is bounded. In this section, we firstly 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). Complete details (omitted here due to space constraints) can be found in an extended version available at http://sands.sce.ntu.edu.sg/p2pstorage/.

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' non-availabilities $\{nav(i)\}$ and bound $B$. We ask whether there is grouping $G_1$, $G_2$ such that both groups are non-empty and $nav(G_1) + nav(G_2) \leq B$.*

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

*Proof:* (Sketch) Reduction from Partition with $nav(i) = 2^{-a_i}$, $B = 2 \cdot 2^{-S/2}$. ∎

As the most unrestricted version of the replication problem is already NP-complete, other problems are similarly NP-complete. For instance, creating 3 non-empty groups corresponds to the strongly NP-complete 3-partition problem [14].

Similarly, maximizing availability with constrained resources (Optimum Availability 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.

**Proposition 2.** *The problem of optimizing availability given the maximum number of peers in each group (OACS) is NP-complete.*

Finally, we consider the problem of minimizing the used storage space, given a constraint on minimum availability (Optimum Storage 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' non-availabilities $\{nav(i)\}$, we ask whether it is possible to construct at least $N$ disjoint groups $\{G_1, \ldots, G_N\}$, so that in each group $G_j$ $\prod_{i \in G_j} nav(i) \leq R$.*

**Proposition 3.** *OSCA is NP-complete.*

*Proof:* (Sketch) Reduction from DUAL BIN PACKING with $nav(i) = 2^{-a_i}$ and $R = 2^{-B}$. ∎

### *B. 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 ($\sum_{Gk \in \mathcal{G}} \prod_{i \in G_k} nav(G_k)$) in this equilibrium is unbounded.

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 [15]: 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 the data available in the longer time period, thus the game can be modeled by the infinitely repeated Prisoner's Dilemma with a discount factor $\delta$ 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 a help of a reputation system), which, in turn, can effectively exclude $i$ from the replication system.

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 [15]. 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 $\exists k' \colon r_{k'}(i, 1, j)$), 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., $R_j = \{i \colon (\exists k_1 : r_{k1}(j, 1, i)) \wedge (\nexists k_2 > k_1 \colon r_{k2}(j, 0, i))\}$. The* pay-off *is $u(i) = nav(i) \prod_{j \colon i \in R_j} nav(j)$.*

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,

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* [15] 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 $av(1) > av(2) > av(3) > av(4)$. 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 4.** *In a subgame perfect equilibrium of the Stochastic Replication Game, assuming that peers use tit-for-tat strategies, peers form $\lfloor \frac{n}{s+1} \rfloor$ replication cliques of size $s+1$ and one clique of size $n \mod (s+1)$. The cliques group peers with similar availability. If peers are numbered according to non-increasing availabilities ($av(i) \geq av(i+1)$), the $k$-th clique is formed by peers $\{1+(s+1)(k-1), 2+(s+1)(k-1), \ldots, s+1+(s+1)(k-1)\}$. The equilibrium is unique if and only if the peers' availabilities differ, i.e., $\forall i\colon av(i) > av(i+1)$.*

*Proof:* If the game ends, 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 $i$ ($1 \leq i \leq s+1$) replicates data of at least one peer $j' > s+1$. Thus, by tit-for-tat, at least one peer $j$ from $\{1, \ldots, s+1\}$ does not replicate $i$'s data (instead replicating data of $j'' > s+1$). Thus, $i$ could increase her availability by stopping replication of $j'$ ($r_k(i,0,j')$) and proposing $r_k(i,1,j)$: as $av(i) > av(j'')$ 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 $\{1, \ldots, (s+1)(i-1)\}$ replicate data between themselves, thus none of them replicates with peers $\{1+(s+1)(i-1), \ldots, n\}$. 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 4.

The following proposition shows that for the system goal, the subgame perfect equilibrium can be arbitrarily far from the equitable solution.

**Proposition 5.** *In the Stochastic Replication Game, the price of anarchy is unbounded.*

*Proof:* Consider an instance with $s+1$ highly available peers (with $nav(i) = n_h \to 0$) and $(s+1) \cdot s$ less available peers ($nav(i) = n_l \to 1$).

A equitable solution minimizing $\sum_{G_i \in \mathcal{G}} \prod_{i \in G_i} nav(i)$ constructs $s+1$ 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). The resulting unavailability is equal to $(s+1)n_h n_l^s$.

In the subgame perfect solution, highly available peers form a clique, thus leaving the less available peers to form cliques between each other. The resulting unavailability is equal to $n_h^{s+1} + s n_l^{s+1}$.

The price of anarchy is thus equal to:

$$\frac{n_h^{s+1} + s n_l^{s+1}}{(s+1) n_h n_l^s} = \frac{n_h^s}{(s+1) n_l^s} + \frac{s n_l}{(s+1) n_h} \xrightarrow[n_h \to 0]{} \infty.$$

∎

We discuss the consequences of such a high price of anarchy in the experimental evaluation (Section VII-A2).

## IV. Complexity and Worst-Case Bounds in the Time Slot Model

In this section, we study the impact of availabilities changing in time. In the previous section, availabilities were constant in time, but continuous in $[0,1]$, corresponding to probabilities. Here, we assume that peers' availabilities change in time, but are crisp. We again study the complexity of centralized matching and for decentralized matching we define the resulting game and prove that the drop in the system performance (the price of anarchy) is at least linear in the number of time slots.[1]

We assume that $av(i,t) \in \{0,1\}$, i.e., for all time moments we can predict with certainty whether $i$ will be off-line or on-line. We also assume that the domain $\mathcal{T}$ of $av(i,t)$ is finite, defined from the current time moment to the period of $av(i,t)$, or the horizon of the prediction. We divide $\mathcal{T}$ into $T$ non-overlapping time slots. For instance, if $\mathcal{T} = 24h$, it can be divided into $T = 24$ one-hour slots. As a result of these two kinds of discreteness, we associate with each peer $i$ a set

[1]Local time for different peers are different, e.g., say in Central Europe versus Japan are different, and are representative of two time slots.

of time slots in which she is available, called $i$'s (discrete) *availability* $A_i \subseteq \mathcal{T} = \{1, \ldots, T\}$.

Consequently, $i$'s data must be replicated only during the time slots not covered by $i$ herself, i.e., $\mathcal{T} - A_i$. As in the previous section, we assume that peers form replication cliques $G_k$, in which every member replicates data of all other members. We denote as $A_{Gk}$ the availability of clique $G_k$, $A_{Gk} = \bigcup_{i \in G_k} A_i$. Clique $G_k$ is *complete*, if availabilities of peers $p_i \in G_k$ cover the whole period $\mathcal{T}$, $A_{Gk} = \mathcal{T}$.

### A. Complexity of Centralized Matching in the Time Slot Model

A centralized matching system should organize peers into complete cliques $\mathcal{G} = \{G_1, \ldots, G_N\}$. As each clique covers $\mathcal{T}$, there is no need for replication between cliques. As a peer replicates the data of other peers from the same clique, the peer's objective is to be in the clique with the smallest number of members. The goal of the system is to provide a *fair* grouping, taking into account the objectives of all the peers.

In a clique $G_k$, each member has the same value of her objective, that is equal to the size of the clique. Thus, a fair aggregation is a function of $(|G_1|, \ldots, |G_N|)$. Among many possible functions, we will use $\min\max(|G_1|, \ldots, |G_N|)$ (for short, denoted as $\min\max|G_i|$), i.e., minimizing the maximum size of the clique.

Therefore, the Fair Peer Replication (FPR) problem is defined as follows.

**Definition 4.** *An instance of the Fair Peer Replication (FPR) problem consists of: the time period $\mathcal{T} = \{1, \ldots, T\}$; the set of peers $\{p_1, \ldots, p_n\}$; and, for each peer $p_i$, its availability $A_i \subseteq \mathcal{T}$. The optimization goal is to assign each peer to a clique, such that all the cliques are complete ($\forall G_k \colon A_{Gk} = \mathcal{T}$) and the size of the largest clique is minimized ($\min\max(|G_1|, \ldots, |G_N|)$).*

As FPR is difficult to analyze theoretically, we will also analyze a related problem—Max Clique Number (MCN), in which the *number* of complete cliques is maximized.

**Definition 5.** *An instance of the Max Clique Number (MCN) consists of the same elements as an instance of FPRP. The optimization goal is to assign each peer to a clique, such that all the cliques are complete ($\forall G_k \colon A_{Gk} = \mathcal{T}$) and the total number of cliques is maximized ($\max|\{G_1, \ldots, G_n\}|$).*

MCN is a restricted version of the Maximum d-Vector Covering problem [14, problem SR3]. $A_i$ correspond to $T$ dimensional vectors, with $t$-th dimension equal to 1 if $t \in A_i$, 0 otherwise. Thus, in MCN the vectors are composed of binary $\{0, 1\}$, and not real $[0, 1]$ numbers.

The following proposition establishes an upper bound on MCN.

**Proposition 6.** *A time slot is called a* critical time slot *(and denoted as $t^c$), if it is covered by the least number of peers. The maximum number of cliques is bounded from above by the number of peers that cover $t^c$. This bound is not always equal to the maximum number of cliques that can be constructed.*

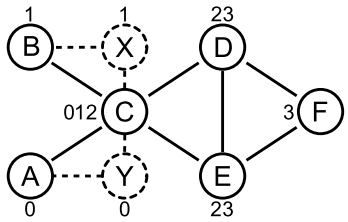

Fig. 1. Reduction of graph coloring to Max Clique Number. Labels assigned by the algorithm are shown next to nodes. Nodes $X$ and $Y$ are added by the reduction. The produced instance of MCN has the following peers' availabilities: $\{\ \{0\}, \{1\}, \{0, 1, 2\}, \{2, 3\}, \{2, 3\}, \{3\}, \{1\}, \{0\}\ \}$

*Proof:* Assume that a critical time slot $t^c$ $(1 \leq t \leq T)$ is covered by $k_c$ peers ($k_c = |\{p_i : t^c \in A_i\}|$). By the definition of $t^c$, for other time slot $t$, $k_t \geq k_c$. The proof is by contradiction. Assume that there are $n > k_c$ complete cliques. But, at least $(n - k_c)$ cliques do not cover $t^c$, thus are not complete.

This bound is not always equal to the maximum number of cliques. Consider an instance with $T = 3$ and three peers having availabilities $\{1, 2\}$, $\{1, 3\}$ and $\{2, 3\}$. All the time slots are covered by exactly two peers, yet only one complete clique can be constructed. ∎

The following proposition states that an optimal solution for MCN can be $O(T)$ times worse for FPRP.

**Proposition 7.** *A solution with the maximum number of complete cliques ($\max|\{G_1, \ldots, G_n\}|$) can be $T/2$ times worse for $\min\max|G_i|$.*

*Proof:* Consider an instance with: $T$ peers $\{1, \ldots, T\}$ that cover the whole period $\mathcal{T}$ $(1 \leq p_i \leq T : A_i = \mathcal{T})$; and also $T$ peers $\{T + 1, \ldots 2T\}$, each covering exactly one, distinct time slot $(T + 1 \leq p_i \leq 2T : A_i = (i - T))$. To maximize the number of cliques, we construct $T$ one-peer cliques, each with one of the peers $1 \leq p_i \leq T$; and one clique with the remaining peers $T + 1 \leq p_i \leq 2T$. The size of the largest clique is $T$. In contrast, to minimize the size of the largest clique, we construct $T$ one-peer cliques as in the previous case. Then, to each clique, we assign one peer from $T + 1 \leq p_i \leq 2T$. All the cliques have thus two peers. ∎

The following proposition shows the complexity of MCN.

**Definition 6.** *The decision version of the Max Clique Number (MCN) problem is the following. Given the set of peers' availabilities $\{A_i\}$, is it possible to construct at least $n$ complete cliques $G_1, \ldots, G_n$, such that each peer belongs to exactly one clique.*

**Proposition 8.** *Max Clique Number is NP-complete.*

*Proof:* (Sketch) Reduction from GRAPH 3 COLORING (Problem GT5, [14]). Given a graph, we construct an instance of MCN, such that, if $n = 3$ complete cliques can be constructed, the cliques match the 3 coloring of the graph. A peer in MCN corresponds to a vertex in the graph; the peer's availability slots are produced by an algorithm (see Figure 1). ∎

### B. Game Theoretic Analysis of the Decentralized Matching

In a decentralized version of the problem, each peer $i$ has to find other peers that will replicate her data during time slots in which $i$ is unavailable. As replicating other peers' data uses $i$'s resources, $i$ will minimize the number of peers she uses as replicators. We start with a proof that a combinatorial (and not game theoretic) version of the problem from a perspective of a single peer is already NP-complete. Then, we define an extensive matching game and show an instance in which the price of anarchy is linear with the number of peers.

We start with an analysis of the replication problem from the perspective of a single peer $i$: assuming that all peers want to replicate data with $i$, which peers should $i$ choose to form a minimal complete clique? We call this problem a Selfish Peer Time Slot Replication (SPTR) problem.

**Definition 7.** *An instance of the Selfish Peer Time Slot Replication (SPTR) problem is composed of a selected peer $i$ and all peers' availabilities $\{A_j\}$. The question is whether there is a complete clique $G$ of size at most $k$.*

**Proposition 9.** *The decision version of Selfish Peer Time Slot Replication is NP-complete.*

*Proof:* (Sketch) Reduction from SET COVER with peers' availabilities corresponding to the subsets. ■

Let us now consider a game theoretic perspective, in which each peer chooses other peers whose data she will replicate. Similar to the analysis of the probabilistic model (Section III-B), we only consider the game in the organizational phase of the system, when peers form replication agreements.

**Definition 8.** *The* Time Slot Replication Game *(TRG) is defined as an extensive game with infinite horizon and simultaneous moves, in which the set of* players*,* terminal histories *and the* player function *is the same as in game 3.*

*The players'* preferences *are two-fold. Firstly, peer $i$ maximizes the number of time slots during which her data is replicated (i.e., the number of time slots in which peers who have $i$ in their replication set are available, $|\bigcup_{j:\, i\in R_j} A_j|$). Secondly, if the number of time slots is the same, peer $i$ minimizes $|P_i|$, i.e., the number of peers whose data she replicates.*

As it is not realistic to assume that all availability patterns will differ, in TRG the subgame-perfect equilibrium does not necessarily result in forming cliques. For instance, consider an instance with $T = 3$ and peers with availabilities $\big\{\{1\},\{2\},\{3\},\{1\},\{2\},\{3\}\big\}$. A grouping in which all peers form a chain (replicating data in pairs $i \leftrightarrow \big((i+1) \bmod n\big)$ has the same size of the largest replication set (3) as a grouping with two cliques.

Nevertheless, a lower bound on the price of anarchy can be established:

**Proposition 10.** *Assuming that peers use tit-for-tat strategies, the price of anarchy in the Time Slot Replication Game is at least $T/2$.*

*Proof:* Consider the same instance as in Proposition 7, with $T$ peers $p_1,\ldots,p_T$ who cover $\mathcal{T}$, and $T$ peers $p_{T+1},\ldots,p_{2T}$, each covering exactly one, different time slot.

In the subgame perfect equilibrium, each of the peers $p_1,\ldots,p_T$ does not replicate any other peer's data (as each such peer already covers the whole $\mathcal{T}$, so $P_i = \emptyset$ is the preferred action). Peers $p_{T+1},\ldots,p_{2T}$ form one, large clique: if any such peer starts replicating one of $p_1,\ldots,p_T$, it would only increase its $P_i$ size, without increasing the availability, thus it would result in a less preferred outcome; yet if any such peer stops replicating $p_j \in \{p_{T+1},\ldots,p_{2T}\}$, by tit-for-tat, in the next iteration $p_j$ would stop replicating $p_i$, and thus reduce the data availability of $p_i$. Thus, in the subgame perfect equilibrium, the largest group has $T$ members.

The socially-optimal solution ($\min\max|G_i|$) groups peers in pairs, in which each one from $p_1,\ldots,p_T$ is paired with one from $p_{T+1},\ldots,p_{2T}$. The maximum clique size is 2. Consequently, the price of anarchy is $T/2$. ■

## V. HEURISTICS FOR THE CENTRALIZED MATCHING IN THE PROBABILISTIC MODEL

The following greedy heuristic optimizes the assignment of peers to cliques in OAFS (Section III-A), 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 [14].

Firstly, the peers are sorted by non-increasing availabilities $av(i)$. Then, $\lceil \frac{n}{s+1} \rceil$ most available peers are assigned to separate cliques. Finally, for each of the remaining peers (in the sorted order), the peer is assigned to clique $G_k$ that currently has the highest unavailability $G_k = \arg\max \prod_{i\in Gk'} nav(i)$.

We experimentally compare this algorithm to the random allocation in Section VII-A2. The assignment resulting from the above heuristic can 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.

## VI. HEURISTICS FOR THE DECENTRALIZED MATCHING

### A. Decentralized Matching in the Probabilistic Model

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 4) 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 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 [16] 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 $k$ ($k = \arg\max_{l:\ replica(i,l)} nav(k)$). $i$ queries the first candidate $j^*$ (in order of non-decreasing score) better than $k$ (for which $nav(j) < nav(k)$). 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 $nav(i) < nav(k')$, 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 $score_o(i,j) = av(j)$. In *Pragmatic Queries*, candidate $j$'s score is equal to the absolute difference between its availability and the availability of the assessing peer $i$, $score_P(i,j) = |av(i) - av(j)|$.

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 $G_l\colon j^* \in G_l$ of the chosen candidate. The two cliques exchange members: the "better" clique groups $s+1$ peers with the highest availability (or the two cliques combined, if the combined clique has at most $s+1$ members); the "worse" clique groups remaining members of both cliques.

**function** $score_c$(peer $i$, peer $j$, $G_k\colon i \in G_k$, $G_l\colon j \in G_l$)
  **if** $|G_k| < s+1$ **then**
    **if** $|G_k| + |G_l| \leq s+1$ **then**
      return $|av(i) - av(j)|$
    **else if** $|G_l| < s+1$ **then**
      return $0.5 \cdot |av(i) - av(j)|$
    **else**
      return 0
  **else**
    $nav_k^H = \max_{i' \in G_k} nav(i')$
    $nav_k^L = \min_{i' \in G_k} nav(i')$
    $nav_l^H = \max_{j' \in G_l} nav(j')$
    $nav_l^L = \min_{j' \in G_l} nav(j')$
    **if** $nav_l^H < nav_k^L$ or ($|G_l| = s+1$ and $nav_l^L > nav_k^H$) **then**
      return 0
    **else**
      return $\max(nav_k^H, nav_l^H) - \min(nav_k^L, nav_l^L)$

Algorithm 1: Computing $j$-th peer score in Explicit 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 1).

If $|G_k| < s+1$, $G_k$ is not complete and the algorithm should increase its size (as it considerably reduces $nav(G_k)$). It is best to have $G_l$ such that the two cliques will be merged into one, $|G_k| + |G_l| \leq s+1$. Thus, $score_C(i,j) = score_P(i,j)$ in this case; otherwise, if $|G_l| < s+1$, $score_C(i,j) = 0.5 score_P(i,j)$; finally if $|G_l| = s+1$, $score_C(i,j) = 0$ (as in this case one of the cliques after merging will still be of size $|G_k|$).

If $|G_k| = s+1$, 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: 1) the intersection of availability ranges is not empty, $[nav_k^L, nav_k^H] \cap [nav_l^L, nav_l^H] \neq \emptyset$; or 2) $|G_l| < s+1$ and $nav_l^H > nav_k^L$. 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 VII-A3.

### B. Decentralized Optimization in the Time Slot Model

The decentralized algorithm constructing replication cliques in the time slot model is similar to *Explicit Cliques*. The differences are: 1) scoring of candidates; 2) merging the cliques; and 3) removing redundant peers from cliques in each iteration.

The scoring function $score_T(i,j)$ depends on whether the clique $G_k\colon i \in G_k$ is complete (covers $\mathcal{T}$). If $G_k$ is not complete, the score of $j$ depends on the number of newly

covered time slots (possibly replacing one of the existing members of $G_k$), $score_T(i,j) = |A_j - A_{Gk}|/T$. If $G_k$ is complete, the score is inversely proportional to the difference in the number of members in cliques, $score_T(i,j) = 1 - ||G_k| - |G_l||/(\max(G_k, G_l))$.

When two cliques $G_k$ and $G_l$ are merged, and $|G_k|+|G_l| > s+1$, we use a greedy algorithm to construct the "better" clique. The algorithm starts with choosing the peer who covers the maximum number of time slots. Then, from the remaining peers, the algorithm adds the peer that covers the maximum number of currently uncovered time slots. This step is repeated until there are peers able to cover uncovered time slots (with a limit of maximum clique size $s+1$) or the number of remaining peers is greater than $s+1$ (as the remaining peers will form one clique).

Finally, as peers also minimize the clique size, each clique periodically removes *redundant* members. A peer $j$ is redundant for clique $G_k$ if and only if all the time slots covered by $j$ are covered by other members of the clique, thus $A_{Gk-\{j\}} = A_{Gk}$.

The algorithm is evaluated in Section VII-B.

## VII. Simulation of the Algorithms

### A. Probabilistic Model

*1) Simulation Settings:* Peers' availabilities were generated in three steps. Firstly, according to [6], 10% of the peers have availability $0.95$, 25%—$0.87$, 30%—$0.75$ and 30%—$0.33$. Then, we added a Gaussian noise with $\sigma = 0.1$ to each availability. Finally, we caped the resulting value, so that $0.03 \leq av(i) \leq 0.97$. Histogram on Figure 2 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 (in the first four rounds we perform only gossiping in order to "warm up" the metric pools).

*2) Centralized Algorithms: 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 50 randomly-generated instances of 10000 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 $0.95 \leq av(i) < 0.96$). Figure 2 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 $1$.

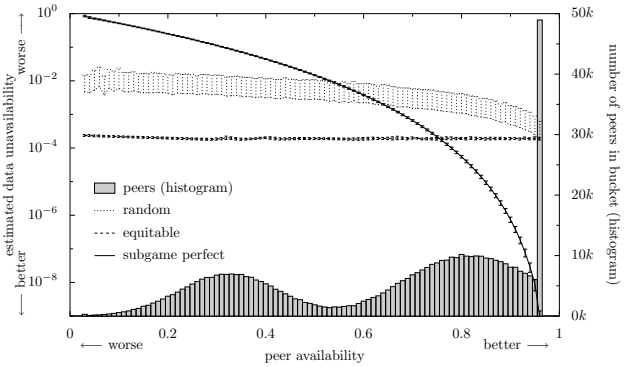


Fig. 2. Peers' expected data unavailability as a function of their availability in random, equitable and subgame perfect assignment. Histogram shows the number of peers in each availability bucket.

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. Thus, the subgame perfect solution is *fair* to participants. 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 to join 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.

Consequently, a robust system might require a hybrid of the selfish and the equitable solution: guaranteeing some minimal level of service to less available peers (but also requiring minimal availability), at the same time rewarding highly available peers with higher data availability.

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

*3) Decentralized Algorithms: Speed of Convergence:* In the next series of experiments, we measure how fast do the decentralized algorithms presented in Section VI-A converge to the subgame perfect cliques.

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 $k/(s+1)$ cliques are formed after approximately $k$ rounds.

Figure 3 compares the convergence speed of *Pragmatic Queries* to *Explicit Cliques*, measured as the median average

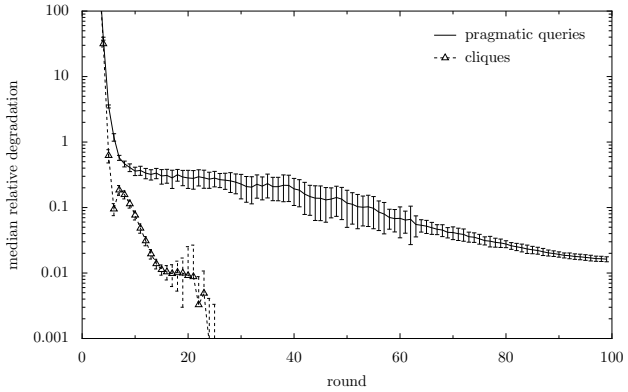


Fig. 3. Convergence speed of decentralized algorithms to subgame perfect cliques. Y axis is the median (computed over only non-zero elements) of relative degradation vs. the subgame perfect solution. $n = 2000$

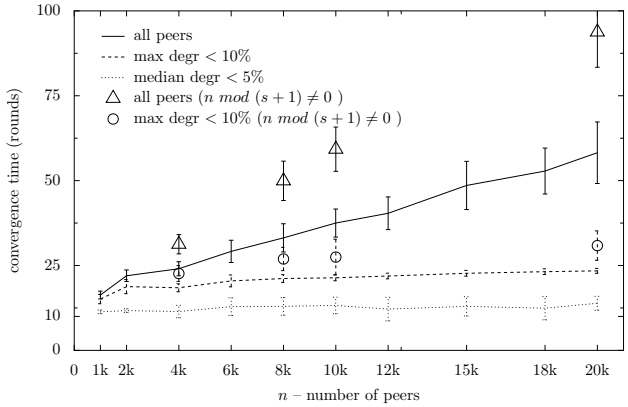


Fig. 4. Convergence speed of the clique-based algorithm to the subgame perfect cliques as a function of the number of peers in the system. The solid line denotes the number of rounds needed to reduce the degradation of all peers to less than $10^{-9}$; the dashed—to reduce the maximum relative degradation to less than $10\%$; the dotted—to reduce the median degradation to less than $5\%$. Outliers (denoted by $\triangle$ and $\circ$) denote the performance for instances with the total number of peers not divisible by the clique size $(s+1)$.

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 high standard deviation observed in rounds 18-25 is an artifact of computing the median from only few values.

In contrast, *Pragmatic Queries* form 2 chains of peers (grouping highly available peers in one and less in the other). The chains are formed because each peer $i$ replicates with $s$ closest neighbours 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 neighbours in favor of higher available peers; thus, the dropped neighbours do not 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. *Explicit Cliques* manages to converge in less than 100 rounds in all instances except 46% of instances with 20,000 peers (however, it converges for all instances with 19,998 peers). The algorithm reduces the median degradation to less than 5% in about 10 rounds. For absolute convergence (solid line) and the approximate maximum degradation (dashed line), the algorithm converges faster if the total number of peers is divisible by 6, the size of the clique. The phenomenon is caused by boundary effects of the one incomplete clique.

### B. Time Slot Model

In the previous series of experiments, we demonstrated that the subgame perfect solution, while efficient for highly available peers, results in low data availability for less available peers. The goal of this series of experiments is to see whether the performance for such peers can be improved by explicitly considering their availability patterns.

*1) Simulation Settings:* In order to simulate only less available peers, we reduced availabilities generated as in the previous experiments (Section VII-A1) by multiplying them by 1/3. A peer is available during $\lceil T \cdot av(i) \rceil$ consecutive hours (from 1 up to 8 hours, see the histogram in Figure 5).

The first hour in which the peer is available is its *time zone* $t_0(i)$. As we could not find data about the distribution of time zones of users in large-scale systems, we generated time zones as follows. Firstly, we do a random permutation of $(1, \ldots, T)$ (we do it only once for each repetition of the simulation). Then, for each peer, we pick the time zone from the permuted list taking the index from the Pareto distribution with controlled shape parameter $\alpha \in \{0.1, 0.5, 1.0, 1.5, 2.0\}$. The greater is $\alpha$, the more peers are in the most popular time zones: for $\alpha = 1$, around $33\%$ of the peers are in the most popular time zone; whereas for $\alpha = 2$ it is $55\%$. The initial permutation permits to obtain time zone distribution in which popularity is not correlated with time zone index. We used Pareto (and not, e.g., uniform) distribution because in some online social networking websites we observed that the distribution of users is highly uneven: rather than being evenly spread over the globe, these sites have vast majority of their users *either* in the US, or in Europe, or in India, or in China.

We used the distributed algorithm described in Section VI-B with the same settings as in the previous section, except for the warm-up period (2 rounds) and the total time of the algorithm (10 rounds).

*2) Influence of Peer Availability:* Figure 5 shows the estimated data unavailability of peers as a function of their availability. A negative correlation between these two values is still clearly visible: a peer available during 1 hour has its data available during roughly 7 hours; while a peer available during 8 hours—roughly 21 hours. These results are similar to the results of the probabilistic model, in which peers with availability of $0.33$ had data availability of approximately $0.9$. Large standard deviations are caused by uneven distribution of peers in time zones: a peer covering unpopular time slots will

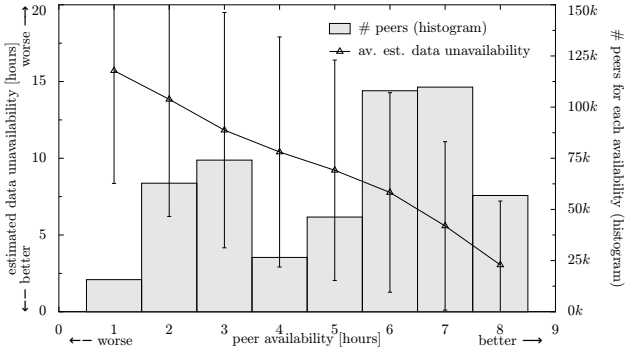


Fig. 5. Estimated unavailability as a function of peers' availability for the time slot algorithm. 250 instances, $n = 2000$. The histogram shows the number of peers for each availability bucket.

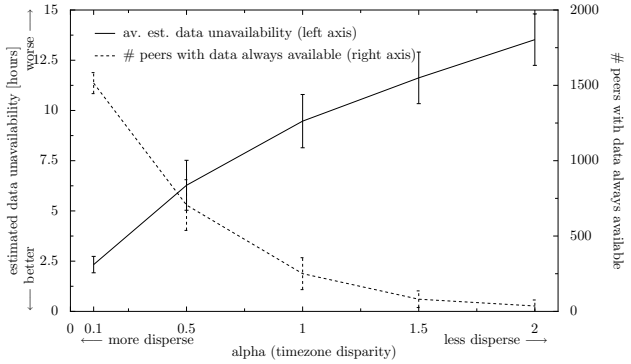


Fig. 6. Estimated average unavailability and number of peers in complete cliques as a function of disparity of peers in time zones. $n = 2000$.

be eagerly accepted to join a clique, even if its availability period is short; on the other hand, a peer covering popular time slots (from a popular time zone) might not find partners to cover unpopular time slots, even if its availability is high.

*3) Influence of Time Zone Disparity:* Figure 6 shows the data unavailability of peers as a function of the time zone disparity $\alpha$. The figure clearly shows that data availabilities are higher if the time zones are more diverse (smaller $\alpha$). In such systems, peers cover very different time slots, and thus are able to find other matching peers to form complete cliques.

For more diverse time zone distribution, the resulting average unavailability is smaller than in the probabilistic model. In the probabilistic model, the average data unavailability averaged over peers with availabilities less than $0.33$ is $0.37$. In the time slot model, for $\alpha = 0.1$, the average data unavailability is $2.32$ hours, which averaged over 24 hours corresponds to average unavailability of $0.10$. Similarly, for $\alpha = 0.5$, we obtain $0.26$. Starting from $\alpha = 1.0$, the data is *less* available in the time slot model than in the probabilistic model. This is caused by the fact that, in the time slot model, adding replication peers does not increase data availability, unless peers have matching availability patterns.

## VIII. Related Work

Our paper analyzed 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 [17]).

There are many papers assessing the loss of efficiency of the system caused by selfishness of individual participants, starting from the analysis of selfish routing in [10]. [3] studies incentives for system-optimal behavior in p2p systems. [18] considers a game of network creation in which selfish nodes optimize the cost of creating new links that minimize distances to other nodes. [19] considers 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. Thus, the typical problems of enforcing agreements can be easily solved using reciprocity and a basic reputation system; that is why in the game theoretic analysis, we could 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, [20] analyzes how many replicas are required to achieve a desired level of availability. Existing systems based on such random placement include Pastiche [21] and Total Recall [7].

Approaches that explicitly optimize the placement of replicas include [5], [6], [22], [23]. [6] studies by simulation an algorithm similar to our Pragmatic Queries, with a slightly more complex acceptance ($score(i, j)$) function. Similarly to our theoretic and simulation results, the experiments in [6] show that highly available peers achieve better performance than the peers with lower availability.

Very recently, [22] and [23] analyzed a problem similar to the decentralized version of the probabilistic model. [22] 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 4 provides a proof for the analogous behavior in our model; moreover, we compute the price of anarchy (Proposition 5). Additionally, we prove that the centralized optimization of availability is NP-hard (Propositions 1-3), which is hypothesised, but deferred for future work in [23].

Equitable optimization of availability in the probabilistic model is similar to the problem considered in [5], where replicas of individual files are spread over a pool of hosts with given availabilities; however, as [5] has an implicit assumption of a single owner of the system, it does not have to consider reciprocity, nor cliques, in the assignment.

[24] 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.

## IX. 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 two idealistic models of peer availability that capture the two important phenomena in p2p systems: uncertainty in the probabilistic model; and diurnal patterns in the time slot model. For both models, 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 high: unbounded in the time-independent model and at least linear in the time slot model. For both decentralized models and the centralized probabilistic model, we proposed heuristics that perform well in simulation experiments.

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 performance for less available peers can be improved by considering diurnal patterns of peer availability, rather than just a single number. However, exploiting diurnal patterns has a measurable impact only when the system has truly global scope, gathering participants from different time zones. Another possibility is to centralize, at least partially, the matching between peers; however, the exact solution is NP-hard to get. Consequently, the solution to this dilemma is most probably somewhere in the middle: a system allowing peers to optimize their replication agreements; but forcing each group to accept, e.g., at least one worse-off peer.

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 [25], redundancy schemes [11], 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 our future work, we plan to conduct more realistic experiments by combining the two models, and simulating permanent and temporary churn of peers during the process of matching. In the theoretical analysis, we left open the complexity of the centralized, equitable optimization in the time slot model. We also plan to design approximation algorithms for both models.

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

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

[5] 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.

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

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

[8] J. Douceur, "Is remote host availability governed by a universal law?" *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 3, pp. 25–29, 2003.

[9] M. Steiner, T. En-Najjary, and E. Biersack, "Long term study of peer behavior in the kad dht," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, 2009.

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

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

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

[13] M. Kostreva, W. Ogryczak, and A. Wierzbicki, "Equitable aggregations and multiple criteria analysis," *European Journal of Operational Research*, vol. 158, no. 2, pp. 362–377, 2004.

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

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

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

[17] 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.

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

[19] T. Moscibroda, S. Schmid, and R. Wattenhofer, "On the topologies formed by selfish peers," in *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*. ACM New York, NY, USA, 2006, pp. 133–142.

[20] R. Bhagwan, S. Savage, and G. Voelker, "Replication strategies for highly available peer-to-peer storage systems," *Proceedings of Fu-DiCo: Future directions in Distributed Computing*, 2002.

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

[22] 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.

[23] L. Toka and P. Michiardi, "Analysis of user-driven peer selection in peer-to-peer backup and storage systems," *Telecommunication Systems J. (to be published)*, 2009.

[24] F. Giroire, J. Monteiro, and S. Pérennes, "P2p storage systems: How much locality can they tolerate?" INRIA, Tech. Rep. 7006, 2009.

[25] 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.