Selfish Neighbor Selection in Peer-to-Peer Backup and Storage Applications

Pietro Michiardi, Laszlo Toka

EURECOM

Abstract. In this work we tackle the problem of on-line backup with a peer-to-peer approach. In contrast to current peer-to-peer architectures that build upon distributed hash-tables, we investigate whether an uncoordinated approach to data placement would prove effective in providing embedded incentives for users to offer local resources to the system. By modeling peers as selfish entities striving for minimizing their cost in participating to the system, we analyze equilibrium topologies that materialize from the process of peer selection, whereby peers establish bi-lateral links that involve storing data in a symmetric way. System stratification, that is the emergence of clusters gathering peers with similar contribution efforts, is an essential outcome of the peer selection process: peers are lured to improve the "quality" of local resources they provide to access clusters with lower operational costs. Our results are corroborated by a numerical evaluation of the system that builds upon a polynomial-time best-response algorithm to the selfish neighbor selection game.

1 Introduction

During the last few years the on-line backup and storage market has witnessed an increasing interest from both academia and industry. Current commercial solutions and research projects present a variety of approaches to the problem of reliable, scalable and available on-line storage for heterogeneous users requiring to store and access a large amount of data from anywhere on the Internet [1–3]. While it is possible to draw a line to separate storage-centric (e.g. Amazon S3 [1]) from sharing-centric (e.g. Wuala [2], AllMyData [3]) approaches, the latter adding the possibility for users to operate fine-grained access control policies on their on-line data with a social-networking flavor, from an architectural point of view on-line backup and storage services can be broken down into those based on server farms [1] and those embracing the peer-to-peer (P2P) paradigm [2, 3].

In this work we focus on P2P approaches, and study an architecture wherein peers are allowed to optimize the amount of resources they dedicate to the system. Specifically, the focus of this paper is on the *neighbor selection* algorithm, which is used by peers to decide where to place fractions of data they need to store. The lack of attention to neighbor selection is mainly due to the structured approach suggested by current system design, e.g. Wuala [2]. In a structured approach (based on a Distributed Hash Table (DHT)), neighbor selection is *implicit* because data is uniformly stored on peers. The main benefit of DHT-based approaches is that they achieve load balancing by spreading data on every peer, irrespectively of their characteristics. Peer heterogeneity in terms of the amount of resources they dedicate to the system cannot be easily taken into account. As a consequence, such systems require additional layers to elicit users' cooperation to the system.

In this paper, we propose an unstructured architecture and study the implications of an uncoordinated neighbor selection algorithm wherein peers are responsible for building up their neighborhood which will store their data. Neighbor selection is modeled as a non-cooperative game in which users selfishly minimize the cost they bear for storing data. In our setting we introduce a global rank of peers in terms of their *profiles*, *i.e.*, the amount and quality of resources they offer to the system. We show that the neighbor selection process reaches an equilibrium in which the system is *stratified*: peers with similar profile cooperate by building bi-lateral links that are used to exchange and store data. The higher the peers' profiles are, the less costly the service they receive from one another is. The consequence of system stratification is a natural incentive for peers to improve the amount and quality of resources they offer to other peers.

The contributions of this paper are the following: *i)* in Sec. 3 we present a novel system design that has built-in incentives for peers to offer resources and to improve the quality thereof; *ii)* we define a simple model, in Sec. 4, that incorporates the cost for storing data in a selfish setting; *iii)* in Sec. 5 we design a polynomial-time algorithm to compute the equilibrium state of our system; *iv)* we show in Sec. 6, using a simulator, that the implication of peer selfishness is a stratified system in which peers can lower their cost of storing data.

2 Related work

Reliable on-line backup and storage has been the subject of a very large number of prior works, both from the research community and from industry.

A notable example of current commercial solutions is represented by Amazon S3 [1], which is based on large clusters of commodity hardware running a custom-built distributed data structure discussed in [4]. The main goals of Amazon S3 are availability and reliability, which are achieved using replication. Availability and reliability do not come for free: end users are compelled to pay for the amount of space they occupy on the data centers and the amount of traffic their content generate [1].

The hybrid P2P design of some on-line backup and storage services such as Wuala [2] and AllMyData [3] require a centralized component to ensure a minimum storage space to end-users which is complemented by storage space at all peers taking part to the application. In current P2P systems availability and reliability are guaranteed by data redundancy through coding. For example, in Wuala data placement is achieved through a DHT layer, in which super-nodes are in charge of uniformly spreading the data on storage nodes. Incentives constitute a key component of Wuala: users must offer an amount of local space inversely proportional to their on-line time [5]. Super-nodes are involved in constantly checking that this constraint is satisfied. Additionally, a distributed reputation mechanism serves the purpose of providing tit-for-tat incentives for users to allocate a large fraction of bandwidth to the P2P network. In contrast to these systems, our goal is to come up with an alternative system design that does not require external incentive mechanisms to support the system operation.

Research on distributed backup and storage applications has proliferated in the literature, although targeting different scenarios than the one we consider here. OceanStore [6], FarSite [7] and TotalRecall [8] represent influential design of such systems, the first based on a mesh of peers that cooperate in storing replicated (for active data) or redundant (for permanent data) blocks, the second using a randomized placement algorithm, the last using a DHT-based approach in selecting the placement of erasure coded data blocks. [9] provided insights to the performance of different data replication strategies in terms of data availability and durability.

Several prior works tackle the analysis of backup and storage applications, although from a more theoretic perspective. For example, [10] studied the potential benefits of a monopoly driven currency-based economy in P2P storage systems, and is orthogonal to this work. Among many other works that offer solutions to ensure fairness in the contributed and consumed storage, [11] suggested to create incentives to users by exploiting their social relationships.

Interestingly, the approach we take in this work can be seen as a network formation game [12], although we depart from the original mathematical tools used to analyze stable topologies.

3 System design

Due to the uncoordinated nature of P2P backup applications, data availability, *i.e.*, ensuring that files can be retrieved in any moment, is an important issue that needs to be addressed. Similarly to related works such as Wuala, in our system we adopt data redundancy using erasure coding[1]: files are split into c equally sized pieces which are then encoded to obtain n blocks. The original file can be reconstructed from any c fragments, where the combined size for the c fragments is approximately equal to the original file size. We term $k = n/c$ the redundancy factor of the coding scheme. File availability can be expressed as

$$\sigma = \sum_{i=c}^{n} \binom{n}{i} p^i (1-p)^{n-i} \qquad (1)$$

where p is the *average* on-line time of peers that compose the system, and σ is the probability that the file is available. Given the average on-line time of peers and the number of fragments c that compose the original file, it is possible to

[1] See [8] for related works on replication and redundancy techniques to achieve data availability.

derive the redundancy factor k that meets the target file availability, which is achieved only if each of the n encoded blocks are placed on *distinct* peers. We explicitly derive the expression of k later in this Section.

Current P2P backup applications assume the average peer on-line availability p to be known and use a global redundancy factor for the whole system: once file fragments are encoded, they are spread uniformly at random on remote locations. *Asymmetric* data placement calls for complex mechanisms to enforce contribution of local space. Indeed, as opposed to the barter-based nature of exchanges we study in this work where direct retaliation is possible, the multilateral nature of asymmetric systems calls for auxiliary instruments, e.g. virtual currency, storage claims [13], to foster peer cooperation. Furthermore, the randomized nature of data placement also implies that the price of unreliable peers is shared among all system participants. Assume, for example, the on-line availability of peers to be distributed according to the normal distribution, that is $p \sim \mathcal{N}(\mu, \sigma^2)$. Then, the probability for a peer i whose availability is $p_i >> \mu$ to store data on a peer j with availability $p_j \geq p_i$ is very small, and vice-versa. Hence, there is no reason for a peer to improve availability, and an additional mechanism compelling peers to offer more resources is required. We now define the resources playing an important role in P2P backup and storage applications.

Definition 1. *The resources peer i offers to the system are:*

- *storage space, $\check{c}_i \in \mathbb{N}$, that is the amount of encoded data chunks a peer stores locally for other peers;*
- *on-line availability, $p_i \in [0, 1]$, expressed as a probability for peer i to be found on-line;*
- *bandwidth $b_i = \min\{u_i, d_i\}$, where u_i, d_i represent respectively the upload and download capacity allocated by the user to the P2P application.*

These factors are tightly coupled: for example, a large amount of local space is useless when peer availability is low; similarly, high availability and space dedicated to the system operation are worth little if the bandwidth allocated to data exchange is not sufficient.

The endeavor of this work is to come up with a system architecture providing *embedded* incentives to foster peer cooperation without requiring any additional mechanisms. We advocate an *unstructured* P2P application with the following objectives: *i)* peers are compelled to offer a fraction of their local storage to other peers; *ii)* peers are incited to increase the on-line time and bandwidth they dedicate to the system. Intuitively, the first objective refers to the "quantity" of resources a peer offers while the second goal addresses the "quality" of such resources. In our system, *neighbor selection* replaces the inherent mapping of data chunks a peer stores in the system achieved by DHT-based solutions. As opposed to selecting remote storage locations uniformly at random, peers are left with the freedom of building a set of neighbors that will hold their data, and are not limited to their social acquaintances [11]. Formally, the problem can be described as follows.

Definition 2. *Let $\mathcal{I}$ denote the set of peers in the system, where $|\mathcal{I}| = N$. Every peer i splits their content in $\hat{c_i}$ equally sized pieces. Pieces are encoded so as to obtain $n_i = k_i\hat{c_i}$ chunks.*

Peers are responsible for establishing (logical) links to remote peers that will store their data, with the constraint that both ends of the link are required to agree to store data for each other: data placement is *symmetric*.

Definition 3. *Let ν_i be the set of peers $\{j \,|\, j \in \nu_i \Leftrightarrow i \in \nu_j\}$, that is the link $i \leftrightarrow j$ is bi-directional. We call ν_i the neighbor set of peer i.*

In our system, a peer is compelled to offer a fraction of local resources for the benefit of other peers. Because of the symmetric nature of our system design, peers are constrained to allocate an amount of storage space equal to the number of encoded chunks they would inject into the system. We can state the above constraint as follows:

Definition 4. *A peer i that needs to store $n_i = k_i\hat{c_i}$ chunks in the system is required to offer an amount of local space equal to: $\check{c_i} = k_i\hat{c_i}\ \forall i \in \mathcal{I}$.*

In practice, we have that $|\nu_i| = k_i\hat{c_i}$, otherwise either the backup data's availability drops due to the low number of peers based on Eq. 1, or unnecessary links are made if $|\nu_i| > k_i\hat{c_i}$. The cardinality of the neighbor set is increasing in the redundancy factor: the larger the redundancy employed in the coding scheme, the larger the number of *distinct* remote locations required to store data. Furthermore, we know that $k_i = f(p_j, \forall j \in \nu_i)$: according to Eq. 1, the redundancy factor is a decreasing function in the on-line time of remote peers that are part of the neighbor set of peer i.

Before proceeding any further, we extend the traditional definition of peer availability to account for the amount of bandwidth a peer dedicates to data exchanges: $\tilde{p_i} = p_i^{b_{ref}/b_i}$. The exponent modulates p_i by the fraction of bandwidth b_i peer i dedicates to the system compared to a reference value b_{ref}. b_{ref} is heuristically set to $\max(b_i)$ for $\forall i \in \mathcal{I}$. Hence, peer availability is slightly underestimated: the consequence for a peer is the requirement for a slightly larger neighbor set size that would compensate "slow" connections.

In this work we assume peers to be rational and selfish: intuitively, selfishness implies that peer will prefer to place data on remote peers offering resources of higher quality. Although this concept will be formalized in Sec. 4, we introduce peer selfishness in a simplified setting. Let ν_i and ν_i' be two distinct neighbor set peer i could link to, such as[2] : $\tilde{p_j} = p\,\forall j \in \nu_i$ and $\tilde{p_k} = p'\,\forall k \in \nu_i'$. Now, let's assume $p' < p$. It follows from our previous observations that:

$$k_i = f(\tilde{p_j} \in \nu_i) < k_i' = f(\tilde{p_k} \in \nu_i') \Rightarrow \check{c_i} = k_i\hat{c_i} < \check{c_i}' = k_i'\hat{c_i}$$

In words, selfish peers prefer to store data on remote peers with higher availability because this implies a reduced demand in terms of local storage space, following the constraint given in Def. 4.

[2] Instead of assuming all peers of ν_i to have the same availability p, it is possible to show similar results for the case in which $p = 1/|\nu_i| \Sigma_{j\in\nu_i} p_j$ or $p = \min_{j\in\nu_i} p_j$

Proposition 1. *In a symmetric system, in which peer i selfishly selects remote locations to store data we have that $k_i = f(\tilde{p_j}, j \in \nu_i) = f(\tilde{p_i})$. That is, the redundancy factor that meets per file availability requirements can be computed as a function of peer i's on-line availability $\tilde{p_i}$.*

Proof. We know by Def. 3 that $j \in \nu_i \Leftrightarrow i \in \nu_j$. Due to the selfish nature of peers and the symmetric nature of links between them, we know that $j \in \nu_i \Leftrightarrow \tilde{p_j} \geq \tilde{p_i}$ and $i \in \nu_j \Leftrightarrow \tilde{p_i} \geq \tilde{p_j}$. Hence, we have that $\tilde{p_i} = \tilde{p_j}$. The proposition follows directly. □

With Prop. 1 at hand, we can formally define the redundancy factor k_i, which can be derived from Eq. 1 using the normal approximation to the binomial distribution:

$$k_i = \left(\frac{\sigma \sqrt{\frac{\tilde{p_i}(1-\tilde{p_i})}{\hat{c_i}}} + \sqrt{\frac{\sigma^2 \tilde{p_i}(1-\tilde{p_i})}{\hat{c_i}} + 4\tilde{p_i}}}{2\tilde{p_i}} \right)^2 \tag{2}$$

We now define peer *profiles*, which summarize the salient features of peers, as they compactly represent the "quality" of resources in terms of on-line time and bandwidth dedicated by a peer to the system. Profiles constitute a global ranking that is used during the execution of the neighbor selection mechanism discussed in Sec. 5.

Definition 5. *The profile of peer i is defined as follows: $\alpha_i = \frac{1}{k_i} \forall i \in \mathcal{I}$. We also define α_i^* to be the bootstrap profile of peer i when joining the system for the first time: $\alpha_i^* = \frac{1}{k_i^*}$ where $k_i^* = f(p_j, \forall j \in \nu^*)$ and ν^* is a random neighbor set.*

Before moving to a detailed description of selfish neighbor selection, we note that in this work we are making the implicit assumption that a method for monitoring the resources a peer dedicates to the system is available. The monitoring component would collect information on peer behavior in terms of profiles and truthful reporting on stored data. Depending on the application setting, the monitoring component can be centralized or distributed. It should be noted that in this paper we also gloss over data maintenance, which is an important problem as discussed in [14]. Due to space constraints, we cannot elaborate any further on monitoring and data maintenance and we will leave these aspects for an extended version of this work.

4 Peer model

The central property of the system we investigate in this work is that peers are free to select the locations where their data chunks will be stored. Neighbor selection is based on peer profiles: peers are assumed to be selfish in establishing links to remote peers holding high profiles and we are interested in studying the re-wiring process and its convergence properties. Here we describe the objective function peers optimize locally whereas in Sec. 5 we formalize the optimization framework that underlies our system.

The complex interplay between peers hinders the task of defining a peer model that accurately mimics the P2P system we investigate in this work. For this reason we define a simple heuristic cost function that incorporates the effects of peer selection and that accounts for the quality of resources offered by peers to the system.

Definition 6. *The cost $\mathcal{C}_i$ that peer i with profile α_i "pays" for storing $n_i = k_i\hat{c}_i$ units of data in a neighborhood ν_i is defined as follows:*

$$\mathcal{C}_i = D_i(\alpha_i, \alpha_j \in \nu_i) + O_i(\alpha_i) + E_i(\alpha_i, \alpha_i^*) = \begin{cases} \log\left(\frac{\hat{c}_i}{\alpha_i^2} + \frac{1}{\alpha_i} + \left(\frac{\alpha_i}{\alpha_i^*}\right)^2\right) & \text{if } |\nu_i| \geq n_i \\ +\infty & \text{otherwise} \end{cases}$$

where the additive terms represent:

- *Degradation cost, D_i*: a target file availability can only be achieved if n_i units of data can be stored on $|\nu_i|$ *distinct* peers; hence this term indicates whether the selected file availability can be reached, that is when the neighbor set size $|\nu_i| \geq n_i$; if this is the case, the degradation cost decreases with the "quality" of peer i's neighborhood given by the profiles α_j of its members; hence, D_i is simply the aggregate profile of the neighbor set of peer i:

$$D_i = \sum_{j \in \nu_i} \frac{1}{\alpha_j} = \sum_{j \in \nu_i} k_j = |\nu_i| k_j \equiv k_i^2 \hat{c}_i = \frac{\hat{c}_i}{\alpha_i^2}$$

- *Opportunity cost, O_i*: describes the cost for peer i due to the loss of local storage space dedicated to hold data for other peers and it is inversely proportional to peer i's profile; by Def. 4, a high profile implies a small redundancy factor thus a smaller $\check{c}_i$; hence, O_i is simply the storage overhead compared to an ideally reliable system:

$$O_i = \frac{\check{c}_i}{\hat{c}_i} = k_i \equiv \frac{1}{\alpha_i}$$

- *Effort cost, E_i*: this term describes the cost induced by a variation in the "quality" of resources peer i offers to the system; the effort cost E_i is not trivial to derive: it follows from the non-linearity of Eq. 2 and the variation in the quality of resources peer i offers compared to the initial state.

We now build upon the user model defined in this Section and formalize the neighbor selection process.

5 Selfish neighbor selection

We study selfish neighbor selection using tools akin to non-cooperative game theory. First, we give a formal definition of the game, then we focus on the algorithmic nature of the optimization problem driving the neighbor selection process. As outlined in [15], selfish neighbor selection can be casted as a *stable* *exchange* (SE) game built on Def. 6 in which peer profiles constitute a (global) preference ordering. Indeed, the SE game belongs to the family of *matching* problems and, as in their simplest version (e.g. the stable marriage problem [16]), peers prefer links to remote peers holding a high profile.

Definition 7. *The stable exchange game is defined as follows:*

- $\mathcal{I}$ *denotes the player set (*N *is the number of players);*
- $\mathcal{S}$ *is the strategy sets available to players:* $\mathcal{S} = (\mathcal{S}_i)$ *for* $\forall i \in \mathcal{I}$*;* $\mathcal{S}_i$ *accounts for the combination of the two strategic variables:* $\alpha_i \in [0, 1]$ *and* ν_i*;*
- $\mathcal{C}_i$ *denotes the cost to player* i *on the combination of the strategy sets.*

In the SE game every peer i seeks to minimize the cost function $\mathcal{C}_i$ by setting appropriately the two strategic variables α_i and ν_i. Note that $\mathcal{C}_i$ also depends on the strategic choice of other players j and that the creation of a link between two peers is conditioned to a bilateral agreement [17]. We now define the optimal strategy for a peer i and the Nash equilibrium of the SE game:

Definition 8. *The best response strategy for peer* i*,* $s_i^* = (\alpha_i^*, \nu_i^*) \in \mathcal{S}_i$ *is obtained by solving the equation* $\arg\min_{\alpha_i,\nu_i} \mathcal{C}_i(s_i)$ *In (Nash) equilibrium we have that* $\mathcal{C}_i(s_i^*, s_{-i}^*) \leq \mathcal{C}_i(s_i', s_{-i}^*)$ *for any player* i *and for any alternative strategy* $s_i' \neq s_i^*$*, where* $s_{-i}^* = (\alpha_i, n_{-i})^*$ *depicts the composition of equilibrium strategy of players other than* i*.*

Due to space limitations, the proof of the existence of the Nash equilibrium will be included in an extended version of this work.

Informally, we can interpret the SE game as follows: there are three forces that drive the decision process of player i, expressed in the function $\mathcal{C}_i$. The opportunity cost pushes player i to increase α_i because this implies a lower redundancy factor k_i hence a decreased amount of local storage offered to the system. The effort cost drives player i to a lower α_i, *i.e.*, reduced on-line probability p_i and allocated bandwidth b_i. The degradation term helps in balancing the two first opposing forces: depending on storage requirements $\hat{c}_i$, the profile α_j of other peers and the number of remote peers with $\alpha_j \geq \alpha_i$ that are eligible for a bilateral agreement, peer i could be better off increasing or decreasing α_i.

Neighbor selection brings to *system stratification*, which is the phenomenon we observe in our game. Our system stabilizes when peers are grouped into clusters, pooling users that have similar profiles[3]. A cluster of peers characterized by high profiles has lower operational costs than one with lower profiles. First, the redundancy factor used by peers in a high-profile cluster is small compared to a low-profile cluster, hence peers will have to dedicate a smaller amount of local space to other peers. Second, reliable peers store data on similarly reliable peers while unreliable peers are bound to store data on other unreliable peers, and are compelled to improve the quality of resources they dedicate to the system as the number of unreliable peers shrinks.

[3] Analytical proofs of the existence, number and size of clusters is out of the scope of this paper and will be addressed in our future work.

We now describe how we implement in an efficient way the *iterated best-response* [18] algorithm to the SE game in order to find an equilibrium. We split the optimization problem that player i faces regarding the strategic variables α_i and ν_i: profile selection and neighborhood construction are *interleaved*. The profile selection is implemented using a technique based on the simulated annealing method [19]: in each iteration of the best-response algorithm, player i randomly modifies α_i by a discrete, fixed value and *estimate* the alteration of the cost $\mathcal{C}_i$ due to the change. A decreasing function $f_k(\mathcal{C}_i^k)$ of the total cost at iteration k of the algorithm is then used to decide whether the new value of α should be adopted or discarded. Once profiles are set, the algorithm solves an instance of the stable matching problem using an extension [15] of the Irving's algorithm to the stable fixtures problem [20]. This procedure is repeated until an equilibrium is found. In [15] we show that the iterated best response algorithm described above runs in *polynomial time*.

It is important to note that, in practice, only once the process of neighbor selection reaches a steady state for all peers, that is, no peer has an incentive to re-wire to other remote peers, the actual data transfer will take place. We also emphasize that our model accounts for a static system set during the neighborhood selection process. Targeting the bootstrap issue of a real growing system is on our agenda.

6 Numerical evaluation

In this section we focus on a numerical evaluation of the neighbor selection process. We built a synchronous simulator, in which time is slotted, and implemented the iterated best-response algorithm discussed in Sec. 5. We examine a closed system in which $|\mathcal{I}| = N = 100$ and bootstrap profiles are normally distributed with mean $\alpha = 0.55$. We assume $\hat{c_i} = 10 \, \forall i \in \mathcal{I}$.

Our goal is to examine the properties of the equilibrium of the SE game (labeled *strategic*) and to compare equilibrium solutions to a simulated DHT-based system (labeled *random*). The implicit hypothesis, to make the two cases comparable, is that peers evaluate their costs similarly. Our evaluation is based on the following metrics:

- Total cost: $\mathcal{C}(t) = \sum_{i \in \mathcal{I}} \mathcal{C}_i(t)$, which cumulates the cost that every peer has to cover for storing their data in the system;
- Average user profile: $\alpha(t) = 1/N \sum_{i \in \mathcal{I}} \alpha_i(t)$, which is the average profile computed at each round of the iterated best-response algorithm;
- Cumulative Distribution Function (CDF) of equilibrium user profiles;
- Profile improvement: $\varDelta_i = \alpha_i - \alpha_i^*$ for $\forall i \in \mathcal{I}$;
- Redundancy factor: $k_i \, \forall i \in \mathcal{I}$. Since clusters may not be exactly uniform in terms of user profiles we report worst, mean and median values;

Due to the randomized nature of our algorithm, the results presented in the following are averaged over 10 simulation runs. In the following, the legend "round" stands for the iteration number of the best-response algorithm.

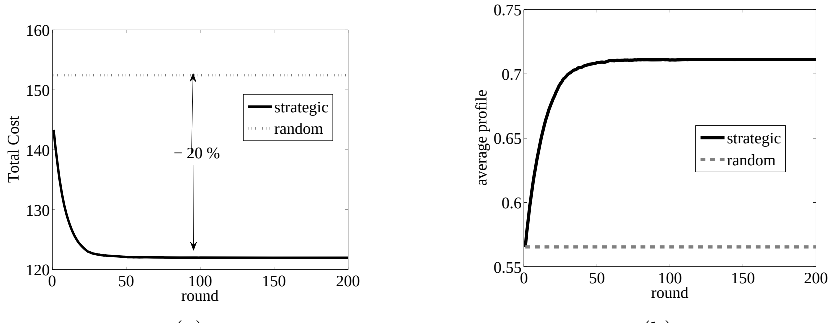

Fig. 1. Time-series of the aggregate cost $\mathcal{C}(t)$ and the average user profiles $\alpha(t)$ for the random and strategic neighbor selection.

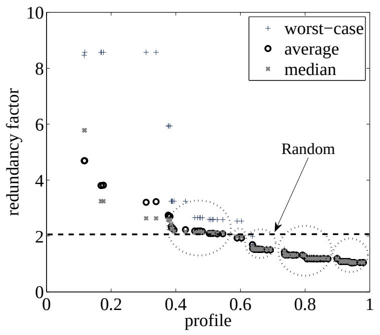

Fig. 2. Redundancy factor adopted by each peer (sorted by their profile) for random and strategic neighbor selection.

Fig. 1(a) illustrates the improvement of strategic neighbor selection when compared to the random neighbor selection: by rewiring their connections according to the iterated best-response to the stable exchange game, peers are able to reduce their costs, and the aggregate figure decreases by 20%. The underlying reason for reduced operational costs is shown in Fig. 1(b). Starting from the same random bootstrap profiles α_i^*, strategic peers increase their profiles while in the random case peer profiles do not change in time and remain fixed to bootstrap values.

Fig. 2 illustrates the redundancy factor adopted by each peer in the system, when using the random or strategic neighbor selection policy. In the random case every peer uses the same redundancy factor: due to the bootstrap profile distribution, the median and mean redundancy factors coincide, and sum to roughly 2 (indicated in the figure with a dashed horizontal line). In the worst case, the redundancy factor is 10 (which corresponds to the y-axis limit). Instead, the strategic neighbor selection differentiates peers in clusters. Peers with a high profile (close to 1) use a lower redundancy factor than peers belonging to a cluster with lower profile (closer to 0). We enriched Fig. 2 to illustrate the clustering phenomenon that emerges at the end of the rewiring process. Peer clusters are emphasized with dotted circles around groups of peers holding *similar* profile. On the upper-left corner of the figure we notice the presence of outliers: for these peers, the effort cost becomes predominant, hence their redundancy factor is very high.

Fig. 3(a) shows the difference between the bootstrap and the equilibrium profiles. We observe that peers holding "extreme" profiles (either high or low values) have less incentives to improve their ranking. We notice a maximum improvement (which amounts to almost 35% difference) for peers with a profile slightly less than the average profile. The exact value of the maximum improvement depends on the input setting to the stable exchange game. Finally, Fig. 3(b) shows that the majority of peers apply a substantial improvement to their profiles as compared to the initial profile distribution.

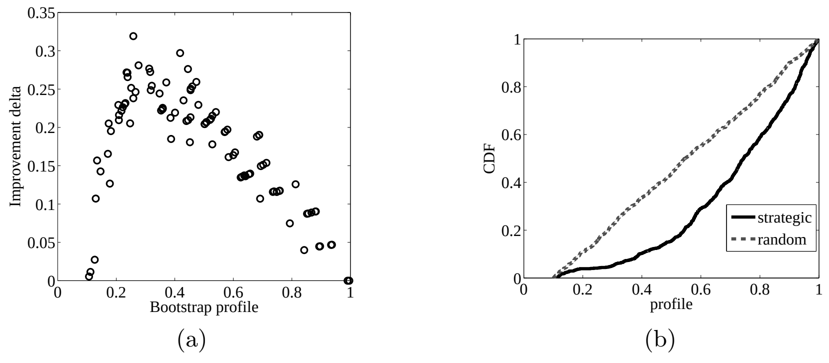

Fig. 3. Improvement and distribution of equilibrium profiles

7 Conclusion and future work

Armed with the realization that current P2P backup and storage applications require complex mechanisms to foster peer cooperation in this paper we presented an alternative system design in which peers, as opposed to previous works, are left with the choice of selecting locations to store data. We introduced a distinction in the amount and the quality of resources peers contribute to the system and showed that selfish neighbor selection alone contributes to the key feature of our approach: incentives to share local resources and to improve their quality are embedded in the system design. In this paper we modeled data placement as a game in which peers minimize the cost for storing data in the system; we also gave a polynomial-time algorithm to compute the equilibrium of the system.

We simulated the selfish neighbor selection process and showed that the system converges to a stratified state: peers are clustered based on their contributions and storage costs are inversely proportional to clusters' quality. This result represented the key motivation for a peer to improve the quality of resources dedicated to other peers.

The results presented in this paper open paths for several future directions: our ultimate goal being the real implementation of such a system we will focus on the analysis of the convergence properties of selfish neighbor selection in an asynchronous setting and on a distributed implementation. We will also focus on the evaluation of system overhead, both in terms of monitoring and repair activities.

References

1. Amazon S3, *http://aws.amazon.com*
2. Wuala, *http://wua.la*
3. AllMyData, *http://allmydata.org*
4. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A.Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, "Dynamo: amazon's highly available key-value store," *ACM/USENIX SOSP*, 2007
5. D. Grolimund, L. Meisser, S. Schmid, R. Wattenhofer, "Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems," *NetEcon*, 2006
6. J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, "Oceanstore: An architecture for global-scale persistent storage," *ACM ASPLOS*, 2000
7. A. Adya, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, M. Theimer, R. Wattenhofer, "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment," *USENIX OSDI*, 2002
8. R. Bhagwan, K. Tati, Y. Cheng, S. Savage, G. M. Voelker, "TotalRecall: System Support for Automated Availability Management," *ACM/USENIX NSDI*, 2004
9. B. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, R. Morris, "Efficient Replica Maintenance for Distributed Storage Systems," *ACM/USENIX NSDI*, 2006
10. P. Maille, L. Toka, "Managing a Peer-to-Peer Data Storage System in a Selfish Society," IEEE JSAC, 2008
11. J. Li, F. Dabek, "F2F: reliable storage in open networks," *IPTPS*, 2006
12. A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, S. Shenker, "On a Network Creation Game," *ACM PODC*, 2003
13. L. P. Cox, B. D. Noble, "Samsara: Honor Among Thieves in Peer-to-Peer Storage", *ACM/USENIX SOSP*, 2003
14. A. Duminuco, E.W. Biersack, T. En-Najjary, "Proactive replication in distributed storage systems using machine availability estimation," *ACM CONEXT* ,2007
15. L. Toka, P. Michiardi, "A dynamic exchange game," *ACM PODC*, 2008
16. D. Gale, L. S. Shapley, "College Admissions and the Stability of Marriage," American Mathematical Monthly, N. 69, 1962
17. J. Corbo, D. C. Parkes, "The price of selfish behavior in bilateral network formation," *ACM PODC*, 2005
18. D. Fudenberg, J. Tirole, "Game Theory," *MIT Press*, 1991
19. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing," Science, V. 220, N. 4598, 1983
20. R. W. Irving, S. Scott, "The stable fixtures problem - A many-to-many extension of stable roommates," Discrete Applied Mathematics, V. 155 N. 17, 2007