Uncoordinated peer selection in P2P backup and storage applications

Laszlo Toka and Pietro Michiardi

***Abstract*—In this work we tackle the problem of on-line backup and storage with a peer-to-peer approach. We propose a novel system architecture involving the users' devices that confederate by pooling their resources and offer an alternative to capital-intensive data-centers. 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, colluding 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 reach lower operational costs. Our results are corroborated by both a game-theoretic analysis and a numerical evaluation of several system configurations.**

I. 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 (user generated content, personal data, etc.) from anywhere on the Internet [1]–[4]. 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]. Our goal is to design a server-less P2P architecture wherein peers (e.g. users' set-top-boxes (STBs) deployed by their Internet Service Provider (ISP)) optimize the amount of local resources they dedicate to the system.

Specifically, the focus of this paper is on the *peer selection* algorithm, which is used by peers to decide where to place fractions of data they need to store or backup. To the best of our knowledge, peer selection has not been studied extensively in the context of on-line backup and storage applications. The lack of attention to peer selection is mainly due to the structured approach suggested by current design of such applications, e.g. Wuala [2]. In a structured approach (based on a distributed hash table (DHT)), peer selection is *implicit* because data is uniformly stored on peers based on the consistent hashing concept [5]. The main benefit of DHT-based approaches is that they achieve load balancing by spreading data on every peer, irrespectively of their characteristics. However, 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 an additional layer to elicit users' cooperation to the system.

In this work, we propose an unstructured architecture and study an uncoordinated, utility-based peer selection algorithm. We present a system design that enables on-line backup and storage service in which the ISP is only involved in monitoring the edge devices that form the P2P network. Monitoring serves the purpose of building a global ranking of peers in terms of their *profiles, i.e.*, the amount and quality of resources they offer to the system. Distributed monitoring could be foreseen, but it is outside of the scope of this work. Nevertheless, peers are responsible for building up their neighborhood which will store their data: peer selection is modeled as a game in which users selfishly minimize the cost they bear for joining the system by adjusting their profile.

We show that the peer 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. Due to the bi-lateral nature of data exchange among peers, the presence of a central server to store excess data that cannot be placed in the P2P network is not required.

The paper is organized as follows: in Sec. III we overview the design of our system, we discuss its key components and propose an objective function that describes and drives the behavior of peers involved in the P2P backup and storage application. Sec. IV is devoted to the definition of peer selection, by casting the problem as a game. We then take an algorithmic perspective and discuss on the implication of an uncoordinated peer selection process. Finally, in Sec. V we present a numerical evaluation of the peer selection process. We review related works in Sec. II and conclude in Sec. VI.

II. Related work

P2P approaches to on-line backup and storage have proliferated in the research literature, and the little space we devote to the state-of-the art in this paper cannot give justice to all of them. OceanStore [7], FarSite [8] and TotalRecall [9] represent influential design of such systems, the first based on a mesh of peers that cooperate in storing replicas (for active data) or redundant (for permanent data) blocks, the second using a distributed, iterative, randomized placement algorithm to place data replicas and the last using a DHT-based approach in selecting the placement of erasure coded data blocks. [10] discusses on the benefits of using network coding in alleviating the costs of data maintenance as opposed to approaches based on source, and erasure coding.

The hybrid P2P design of aforementioned Wuala [2] and AllMyData [3] requires a centralized component to ensure a minimum storage space to end users which is complemented by storage space at all available peers taking part to the application. In Wuala, files are split up into pieces, which are encrypted and spread on the P2P network, with each piece being stored on at least five different peers. Data placement is achieved through a double 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[1] [6] and 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 their bandwidth to the P2P network.

Several works have defined subtle economic frameworks to design and analyze incentive schemes to enforce user cooperation (e.g., [11] and references therein). Our work is related also to the "selfish neighbor selection" problem, initiated with the seminal work on network creation games [12]. Related to the formulation of peer selection that we study in this paper, the uncoordinated creation of routing overlays have been investigated in [13].

[1]The probability for a user to be found on-line should be *at least* 0.17.

Despite a large literature on P2P storage and backup applications, to the best of our knowledge, the question of whether it would be possible to design a P2P system with incentives *embedded* in the early stages of the system architecture, without requiring additional mechanisms, has not been addressed in prior works.

III. System design

In this section we present an overview of our system. We assume the related application to be executed as a service on users' STBs which allows end-users to set a bootstrap value to the amount of dedicated resources: users can select the service on-line time and the fraction of bandwidth (both up-link and down-link) allocated to the backup and storage service. On the ISP side, we assume the presence of a monitoring infrastructure measuring the status of the devices and the amount of available bandwidth dedicated to each of them. We push the execution of the basic algorithms that constitute the application to edge devices to minimize the ISP implication in operating the system.

Let $\mathcal{I}$ denote the set of peers taking part to the system. Every peer i splits its content to be stored in the system in c_i unit-sized pieces, and is responsible for establishing (logical) links to remote peers that will potentially store its data, *i.e.*, the *peer selection* algorithm is explicit. Any link between two peers is assumed to be *symmetric*: both ends of the link are required to store data for each other. We denote by n_i the indicator vector of links established by peer i: $|n_i| = \mathcal{I}$, $n_i = (l_{i,1}, l_{i,2}, ..., l_{i,\mathcal{I}})$ and $l_{i,j} = 1$ *iff* peer i has a link to peer j and peer j has a link to peer i. Furthermore, let also c_{ij} denote the number of data pieces peer i stores on peer j. In order to maintain symmetric collaboration, $c_{ij} = c_{ji}$, otherwise link $l_{i,j}$ cannot be established.

In the following we focus on the design issues we addressed in our P2P backup and storage system, while a detailed description of the peer selection process is given in Sec. IV.

A. Data availability

The first and foremost issue of a distributed storage system is to ensure data *availability*. Due to the uncoordinated nature of the P2P setting we are considering, in which peers may join and leave at any time, persistent data storage is difficult to achieve and calls for sophisticated data management mechanisms. Replication or redundancy have been suggested as effective means to cope with poor peer availability [9], [14]. In our system we adopt data redundancy using erasure coding: backup data is split into c unit-sized pieces which are then encoded to obtain n blocks. The key property of erasure codes is that the original data can be reconstructed from any c fragments, where the combined size for the c fragments is approximately equal to the original data size. We term $k = n/c$ the redundancy factor of the coding scheme. The value of k must be set appropriately, depending on the desired per data availability target, which can be expressed as [9]:

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

where p is the *average* on-line time of peers that compose the system, and ϵ is the probability that the data is unavailable. The expression to compute the redundancy factor, given a target data availability, an average on-line time, and the number of fragments c writes as [9]:

$$k = \left(\frac{\sigma \sqrt{\frac{p(1-p)}{c}} + \sqrt{\frac{\sigma^2 p(1-p)}{c} + 4p}}{2p} \right)^2 \quad (1)$$

where σ is the number of standard deviations in a normal distribution for the required level of file availability[2].

Given an average peer on-line availability p, it is common to assume a *global* redundancy factor for the whole system. This is done for example in traditional P2P systems such as Wuala. The price of unreliable peers with low on-line availability is then shared among all system participants. This potentially unfair design choice can be mitigated through subtle incentive mechanisms and our peer selection algorithm.

B. Incentives and fairness

We now define the resources that peer i contributes to the system: they play important roles for the P2P backup and storage application.

- storage space, $\hat{c}_i \in \mathbb{N}$, 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.

[2]Data availability of 0.9999 results in $\sigma = 3.7$.

Our objective in this work is two-fold. Firstly, peers should be compelled to offer a fraction of their local storage to other peers in the system; secondly, we want to provide incentives for users to increase the on-line time and bandwidth they dedicate to the service. Intuitively, the first objective refers to the “quantity”, while the second goal addresses the “quality” of resources a peer offers. Besides the specific incentive mechanism used to achieve these goals, measuring the amount of resources a peer dedicates to the system also represents an important issue. In this work, we leverage on the monitoring capabilities of the ISP, and on its intentions to use them. Since the application is assumed to operate on ISP-owned STBs, that provide controlled and secure environment, the ISP is able to offer the additional backup and storage service to its subscribers for a relatively low cost of monitoring by exploiting its already deployed hardware resources.

Peer characteristics are encoded into a single scalar value, α, that we term the *profile*. We suggest the following heuristic to compute α. The ISP estimates the redundancy factor k_i peer i would impose on the system using a combination of the on-line time p_i and the dedicated bandwidth b_i. k_i is obtained from Exp. 1 where the average on-line time p is substituted by $p_i^{b_{ref}/b_i}$. The latter term modulates p_i by the fraction of bandwidth b_i compared to a reference value b_{ref} which is set to $\max_{i \in \mathcal{I}}(b_i)$. Hence, the redundancy factor k_i is slightly overestimated: this is intentional and serves as an additional incentive for peers to improve the quality of resources they offer (note that large k_i implies high redundancy due to low peer availability). Then the profile of peer i is given by $\alpha_i = \frac{1}{k_i}\ \forall i \in \mathcal{I}$. Peer profiles are used in two flavors: they constitute a global ranking that is used during the execution of the peer selection mechanism and also they dictate, for a particular peer, the exact amount of storage space that should be dedicated to the system. Peer i holding profile α_i who needs to store c_i chunks will be required to offer an amount of local space of $\hat{c}_i = \frac{c_i}{\alpha_i} = c_i k_i\ \forall i \in \mathcal{I}$.

This definition implies that peer i will allocate an amount of storage space equal to the number of encoded chunks it would inject into a system consisting of peers with its same profile. The *symmetric* nature of data exchange in our system and the *strict correlation* between the load imposed on the system and the amount of storage each peer is compelled to offer render an additional mechanism to enforce peers’ contribution unnecessary.

C. User model

As opposed to selecting remote storage locations uniformly at random, in our system peers build their neighbor set while trying to optimize a local objective function that we now describe in detail. For sake of simplicity, let's assume every peer will store the same amount of data chunks in the system, that is: $c_i = c$ $\forall i \in \mathcal{I}$. We now introduce the cost function peer i minimizes when establishing links to remote peers. Note, that in the presented theoretical model we consider a static system set where no application-level churn occurs. Once the synchronized process of peer 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. The cost function $\mathcal{C}_i$ we define in this section *mimics* the "physical" key factors that affect the behavior of a real-world P2P backup and storage application.

Definition 1: The cost $\mathcal{C}_i$ that peer i with profile α_i "pays" for a neighborhood n_i is $\mathcal{C}_i = D_i(\alpha_i, \alpha_{j \in n_i}) + O_i(\alpha_i) + E_i(\alpha_i)$, where the additive terms represent:

Degradation cost (D_i) is a function of peer i's profile and of the profile α_j of every peer j that is selected to be in the neighbor set n_i: the technique based on erasure codes used in the system guarantees a certain probability to recover encoded chunks if every encoded chunk can be placed on a different remote peer, hence the degradation cost accounts for the decreased data availability occurring when the size $|n_i|$ of peer i's neighborhood is not sufficient to store one unit of encoded data on each neighbor;

Opportunity cost (O_i) describes the price payed by peer i for the loss of local storage space dedicated to the service;

Effort cost (E_i) accounts for the cost incurred by peer i to maintain the profile α_i.

IV. User-driven peer selection

In this section we study the peer selection process using tools akin to non-cooperative game theory. First, we give a formal definition of the game that involves a peer i participating to the P2P backup and storage application. We then focus on the algorithmic nature of the optimization problem imposed by the peer selection process. As illustrated in [16] and [17], we cast the problem as a *matching game* in which: i) players optimize the profile selection using techniques belonging to the family of simulated annealing [19]; ii) players execute an extended version of Irving's algorithm [15] to the stable fixtures problem to find a stable matching. We conclude the section discussing the implications of the solution to the stable exchange game from a system's perspective.

A. The stable exchange game

We cast the peer selection process as a game, labeled the stable exchange game, that is built on the user model defined in Sec. III.

Definition 2: The stable exchange game can be defined as follows:

- $\mathcal{S}$ depicts the strategy sets available to players: $\mathcal{S} = (\mathcal{S}_i)$ $\forall i \in \mathcal{I}$; $\mathcal{S}_i$ accounts for the combination of the two strategic variables: $\alpha_i \in [0,1]$ and $n_i \subseteq \mathcal{N}_i = \{\{i, j, c_{ij}\} : j \in \{\mathcal{I} \setminus i\}, c_{ij} \in \mathbb{N}_{[0, \min(\hat{c}_i, \hat{c}_j)]}\}$;
- $\mathcal{C}_i$ denotes the cost to player i on the combination of the strategy sets.

In the stable exchange game every peer seeks to minimize its cost by setting appropriately the two strategic variables α_i and n_i in response to the strategic choice of other players. The creation of a link between two peers is conditioned to a bilateral agreement [18], which depends on peers' profiles and storage requirements. The optimal user strategy $s_i^* = (\alpha_i^*, n_i^*) \in \mathcal{S}_i$ is obtained by solving the equation: $\arg\min_{\alpha_i, n_i}(\mathcal{C}_i(s_i))$. In (Nash) equilibrium $\mathcal{C}_i(s_i^*, s_{-i}^*) \leq \mathcal{C}_i(s_i', s_{-i}^*)$ for any player i and for any alternative strategy tuple $s_i' \neq s_i^*$, where $s_{-i}^* = (\alpha_{-i}, n_{-i})^*$ depicts the composition of equilibrium strategy tuples of players other than i.

There are three forces that drive the decision process of player i, expressed in the hybrid cost function $\mathcal{C}_i$. On the one hand, 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. On the other hand, 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 c_i, the number and profiles of remote peers, peer i could be better off increasing or decreasing α_i.

B. An algorithmic perspective

In this work we split the optimization problem that the players face regarding their strategic variables α and n: 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, players randomly increase or decrease α by a discrete, fixed value and *estimate* the alteration of their costs due to the change. Only a new value of α that entails lower total cost is adopted, and a new neighbor selection process is thus started. The consequence is that we let players create their peerings from scratch in every iteration of the best-response algorithm. In practice, however, this does not mean that repeated peerings will involve actual data transfers: once a stable matching is found for every peer, the storage process takes place. This interleaving technique has been illustrated in [16], in which we show that every new instance of the stable matching problem that arise when peers update their profiles can be solved in linear time using an extension [17] of the Irving's algorithm to the stable fixtures problem [15].

C. Implications of user-driven peer selection

The iterated-best response algorithm builds the solution to the stable exchange game, which is a stable overlay graph representing (logical) neighborhood relations among peers taking part to the backup and storage application. The price to pay for the advantages of a system based on symmetric and unstructured peering relations we discuss here may appear to be *scalability* and computational issues. Although our solution yields linear runtime, with the knowledge of the equilibrium stratification result one might propose even faster approaches for a real system. In a practical implementation, where asynchronous user arrivals and leaves may happen, the operation of a single peer i is very simple: based on its current profile α_i, the peer queries the ISP *only* for the required number of other peers with a similar profile, to be able to store c_i units of data. As shown in [17], we know that any random deviation from this behavior would not be worth. Furthermore, due to the bi-lateral nature of peering agreements, a peer will not be able to "cheat" and request the ISP for peers with higher profiles, as any tentative of establishing a link to those peers will fail. A secure and unique identification scheme is however required to prevent attackers from tampering with our scheme, e.g. by "spoofing" identities. In our agenda we plan to implement the necessary security measures (data encryption, user authentication, backup verification), and we address application-level churn (*i.e.*, dynamic user set) issues that require to reshape our model to asynchronous user memberships.

V. Numerical evaluation

In this section we focus on a numerical evaluation of the peer selection process we discussed in Sec. IV. The results we show in this section are based on the following assumptions. We assume a synchronous setting, in which time is slotted, and implement the iterated best-response algorithm. The set of players does not vary in time and we uniformly at random generate a bootstrap profile $\hat{\alpha_i} \in [0.1, 1]\ \forall i \in \mathcal{I}$. We assume that peer i stores $c_i = c = 10$ units of data in the system and holds $\mathcal{C}_i = D_i(\alpha_i, \alpha_{j\in n_i}) + O_i(\alpha_i) + E_i(\alpha_i) = \sum_{j\in n_i}\left(\frac{c_{ij}\alpha_i}{c_i}\right)^{1+\alpha_j} + \left(\frac{\hat{\alpha_i}}{\alpha_i}\right)^2 + \alpha_i^2$. In this heuristic cost function if peer i's neighbor set is sufficiently large to allocate one encoded block per neighbor, the predominant effect in the degradation cost becomes the "quality" of peer i's neighborhood given by the profile α_j of its members (*i.e.*, lower quality implies increased costs). Moreover the opportunity cost expresses, given peer i's bootstrap profile $\hat{\alpha_i}$ that we label "effort-less profile" and the amount of data to be stored in the system, the amount of local storage peer i will be compelled to offer relatively to the effort-less amount. Due to the randomized nature of our algorithm, the results presented in the following are averaged over 10 simulation runs.

Our goal here is to examine the properties of the equilibrium of the stable exchange game and to compare equilibrium solutions to a simulated DHT-based system in which peers do not optimize their bootstrap profile, which remains fixed in time, and select remote peers uniformly at random. Our evaluation is based on the cumulative distribution function (CDF) of equilibrium user profiles, and the redundancy factors $\hat{k}_i\ \forall i \in \mathcal{I}$ (we compute $\hat{k}_i$ using respectively the worst, mean and median profile of i's peers).

We now let $\mathcal{I} = 100$ and carry out a comparative analysis of the user driven peer selection (labeled *strategic peer selection*) against a random peer selection strategy that mimics a DHT-based system (labeled *random peer selection*). The implicit hypothesis, to make the two cases comparable, is that peers evaluate their costs with the same cost function that we defined above also when they cannot optimize the variables α_i and n_i.

We observe in Fig. 1 that the majority of peers apply a substantial improvement to their profiles as compared to the initial profile distribution (that is kept in the random selection), which corroborates our analysis on the incentive properties of our peer selection mechanism.

Fig. 2 illustrates the impact of the peer selection process from a system point of view: we plot the redundancy factor adopted by each peer in the system, when using the random or strategic peer selection policy. In the random case every peer uses the same redundancy factors considering the system lowest, average and median profiles respectively. Due to the uniformity of the bootstrap profile distribution, the median and mean

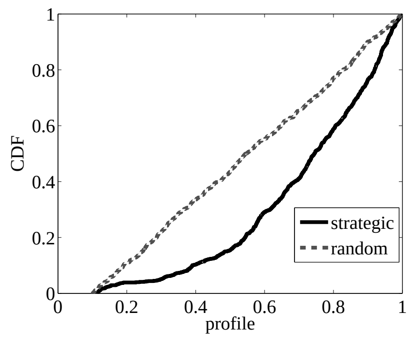

Fig. 1. Equilibrium distribution of profiles for strategic peer selection.

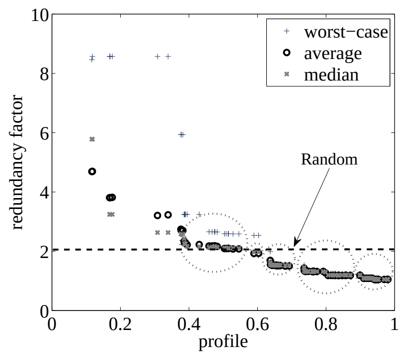

Fig. 2. Redundancy factor adopted by each peer (sorted by their profile) for random and strategic peer selection.

redundancy factors coincide (around 2 as indicated in the figure with a dashed horizontal line); the worst case redundancy factor is 10 (which corresponds to the y-axis limit). Instead, the strategic peer selection results in stratification: peers with high profiles (close to 1) will use a lower redundancy factor than peers with lower profile.

VI. Conclusion

We presented a novel system architecture for on-line backup and storage applications which involves an ISP and end-users' devices embracing the P2P paradigm whereby the system capitalized on storage resources already deployed at the edge of the network. We proposed a peer selection mechanism that allows peers to take selfish data placement decisions which results in an unstructured system that requires no additional mechanisms to align user incentives towards a good system operation. Peer selection has been modeled with the tool-set of non-cooperative game theory and we took an algorithmic perspective to efficiently compute stable solutions, moreover we identified a simple way of implementing our mechanism in a decentralized way which is robust against peers that deviate from the normal system operation. The analysis of our peer selection mechanism through a numerical evaluation of a synthetic system shows that it provides built-in incentives for peers to improve the quality of resources they offer to the system. We achieved this property due to the outcome of peer selection. *i.e., system stratification*.

Acknowledgments: This work has been partially funded by the ANR (Agence Nationale de la Recherche) Project SPREADS within the Telecommunications Program. This work has also been partially funded by the STREP Project NADA within the 7th IST Framework Program.

References

[1] Amazon S3, *http://aws.amazon.com*

[2] Wuala, *http://wua.la*

[3] AllMyData, *http://allmydata.org*

[4] UbiStorage, *http://www.ubistorage.com*

[5] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, D. Lewin, "Consistent hashing and random trees," *STOC*, 1997

[6] D. Grolimund, L. Meisser, S. Schmid, R. Wattenhofer, "Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems," *NetEcon*, 2006

[7] 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," *ASPLOS*, 2000

[8] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, R. P. Wattenhofer, "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment, " *OSDI*, 2002

[9] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, G. M. Voelker, "TotalRecall: System Support for Automated Availability Management, " *NSDI*, 2004

[10] A. G. Dimakis, P. Brighten Godfrey, M. J. Wainwright, K. Ramchandran, "Network Coding for Distributed Storage Systems", *INFOCOM*, 2007

[11] P. Antoniadis, C. Courcoubetis, R. Mason, "Comparing economic incentives in peer-to-peer networks, " *Computer Networks, Vol. 46, N. 1*, 2004

[12] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, S. Shenker, "On a Network Creation Game," *PODC*, 2003

[13] N. Laoutaris, G. Smaragdakis, A. Bestavros, J. W. Byers, "Implications of Selfish Neighbor Selection in Overlay Networks," *INFOCOM*, 2007

[14] R. Bhagwan, S. Savage, G. M. Voelker, "Understanding Availability," *IPTPS*, 2003

[15] 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

[16] L. Toka, P. Michiardi, "A dynamic exchange game," *PODC*, 2008

[17] L. Toka, P. Michiardi, "Analysis of user-driven peer selection in peer-to-peer backup and storage systems," *GameComm*, 2008

[18] J. Corbo, D. C. Parkes, "The price of selfish behavior in bilateral network formation," *PODC*, 2005

[19] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing," *Science, Vol. 220, N. 4598*, 1983.