Finding Good Partners in Availability-aware P2P Networks

Stevens Le Blond, Fabrice Le Fessant, Erwan Le Merrer

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Finding Good Partners in Availability-aware P2P Networks

Stevens Le Blond — Fabrice Le Fessant — Erwan Le Merrer

N° 6795

January 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6795--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Finding Good Partners in Availability-aware P2P Networks

Stevens Le Blond [*] , Fabrice Le Fessant [†] , Erwan Le Merrer [‡]

Thème COM — Systèmes communicants
Équipes-Projets Asap et Planète

Rapport de recherche n° 6795 — January 2009 — 14 pages

Abstract:

In this paper, we study the problem of finding peers matching a given availability pattern in a peer-to-peer (P2P) system. We first prove the existence of such patterns in a new trace of the eDonkey network, containing the sessions of 14M peers over 27 days. We also show that, using only 7 days of history, a simple predictor can select predictable peers and successfully predict their online periods for the next week. Then, motivated by practical examples, we specify two formal problems of availability matching that arise in real applications: *disconnection matching*, where peers look for partners expected to disconnect at the same time, and *presence matching*, where peers look for partners expected to be online simultaneously in the future. As a scalable and inexpensive solution, we propose to use epidemic protocols for topology management, such as T-Man; we provide corresponding metrics for both matching problems. Finally, we evaluated this solution by simulating two P2P applications over our real trace: *task scheduling* and *file storage*. Simulations showed that our simple solution provided good partners fast enough to match the needs of both applications, and that consequently, these applications performed as efficiently at a much lower cost. We believe that this work will be useful for many P2P applications for which it has been shown that choosing good partners, based on their availability, drastically improves their efficiency.

Key-words: availability,peer-to-peer,matching,epidemic,protocols

[*] INRIA Sophia Antipolis
[†] INRIA Saclay
[‡] INRIA Rennes

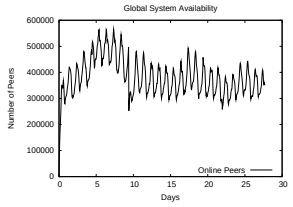
Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Comment trouver de bons partenaires dans un réseau pair-à-pair en fonction de sa disponibilité

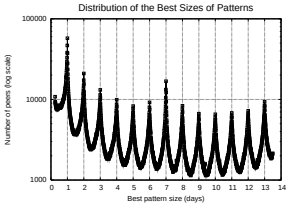
Résumé :

Dans ce papier, nous étudions le problématique de trouver des partenaires suivant un critère de disponibilité dans un réseau pair-à-pair. Nous commençons par montrer l'existence de régularités de disponibilité dans une nouvelle trace du réseau eDonkey, contenant les sessions de 14M de pairs sur 27 jours. Nous montrons aussi que, en utilisant 7 jours d'historique, une prédicteur simple peut sélectionner des pairs prévisibles et prédire avec succès leurs périodes de disponibilité sur la semaine suivante. Ensuite, nous spécifions deux problèmes formels de sélection en fonction de la disponibilité, qui se présentent dans des applications réelles: la sélection pour la déconnexion, qui recherche les pairs qui se déconnecteront probablement en même temps, et la sélection pour la présence, qui recherche les pairs qui seront probablement présents en même temps dans le futur. Comme solution peu coûteuse et passant à l'échelle, nous proposons d'utiliser des protocoles épidémiques de gestion de topologie, tels que T-Man; nous fournissons les métriques correspondant à nos deux problèmes. Finalement, nous avons évalué cette solution en simulant deux applications pair-à-pair sur notre trace réelle. Les simulations ont montré que notre simple solution fournit de bons partenaires suffisamment vite pour les besoins des deux applications, et qu'en conséquence, ces applications fonctionnent aussi efficacement à un coût bien moindre. Nous pensons que ce travail sera utile pour toutes les applications pair-à-pair pour lesquels il a été montré que le choix de bons partenaires peut augmenter considérablement les performances.

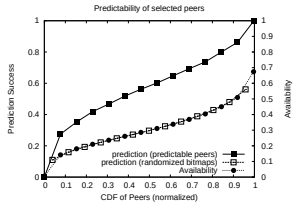
Mots-clés : disponibilité,pair-à-pair,épidémique,protocole



(a) System Availability



(b) Auto-Correlation on Sessions



(c) Prediction versus Availability

Figure 1: (a): Diurnal patterns are obviously visible on the global system availability. (b) The auto-correlation on the sessions shows that the best pattern size is one day, followed by one week. (c) Whereas availability determines the prediction with random bitmaps, daily patterns improve the prediction with real bitmaps (e.g. for 60% of peers (x=0.4), 50% of predictions (y=0.5) are successful, but only 25% with random bitmaps).

1 Introduction

Churn is one of the most critical characteristics of peer-to-peer (P2P) networks, as the permanent flow of peer connections and disconnections can seriously hamper the efficiency of applications [9]. Fortunately, it has been shown that, for many peers, these events globally obey some availability patterns ([18, 19, 2]), and so, can be predicted from the uptime history of those peers [15].

To take advantage of these predictions, applications need to be able to dynamically find good partners for peers, according to these availability patterns, even in large-scale unstructured networks. The intrinsic constitution of those networks makes pure random matching techniques to be time-inefficient facing churn.

In this paper, we study a generic technique to discover such partners, and apply it for two particular matching problems: *disconnection matching* , where peers look for partners expected to disconnect at the same time, and *presence matching*, where peers look for partners expected to be online simultaneously in the future. These problems are specified in Section 3. We then explain that T-Man [12], a standard epidemic algorithm for topology management, is a good candidate to solve these problems. However, in order to converge to the desired state or topology (here matched peers), T-Man needs an accurate metric to compute the distance between peers. In Section 4, we describe how T-Man works and propose a particular metric for each of our matching problems.

To evaluate the efficiency of our proposal, we simulate an application for each matching problem: an application of *task scheduling*, where tasks of multiple re-

mote jobs are started by all the peers in the network (disconnection matching), and an application of *P2P file-system*, where peers replicate files on other peers to have them highly available (presence matching). To run our simulations on a realistic workload, we collected a new trace of peer availability on the eDonkey file-sharing network. With the connections and disconnection of 14M peers over 27 days, this trace is the largest available workload, concerning peers' availability. In Section 2, we show that peers in this trace exhibit availability patterns, and, using a simple 7-day predictor, that it is possible to select predictable peers and successfully predict their behavior over the following week.

Our simulation results, in Section 5, show that our T-Man based solution is able to provide good partners to all peers, for both applications. Using availability patterns, both applications are able to keep the same performance, while consuming 30% less resources, compared to a random selection of partners. Moreover, T-Man is scalable and inexpensive, making the solution usable for any application and network size.

We believe that many P2P systems and applications can benefit from this work, as a lot of availability-aware applications have been proposed in the literature [3, 8, 17, 5, 22]. Close to our work, [9] shows that strategies based on the longest current uptime are more efficient than uptime-agnostic strategies for replica placement; [15] introduces sophisticated availability predictors and shows that they can be very successful. However, to the best of our knowledge, this paper is the first to deal with the problem of finding the best partners according to availability patterns in a large-scale network. Moreover, previous results are often computed on synthetic traces or small traces of P2P networks.

2 Availability Patterns in eDonkey

In this section, we describe the characteristics of the trace we collected for the needs of this study. With a few thousand peers online at the same time, most other traces collected on P2P systems [18, 10, 2] lack massive connection and disconnection trends, for the study of availability patterns on a large scale.

2.1 The eDonkey Trace

In 2007, we collected the connection and disconnection events from the logs of one of the main eDonkey servers in Europe. Our trace, available on our website [1], contains more than 200 millions of connections by more than 14 millions of peers, over a period of 27 days. To analyse this trace, we first filtered useless connections (shorter than 10 minutes) and suspicious ones (too repetitive, simultaneous or with changing identifiers), leading to a *filtered trace* of 12 million peers.

The number of peers online at the same time in the filtered trace is usually more than 300,000, as shown by Fig. 1(a). Global diurnal patterns of around 100,000 users are also clearly visible: as shown by previous studies [11], most eDonkey users are located in Europe, and so, their daily offline periods are only partially compensated by connections from other continents.

For every peer in the filtered trace, the auto-correlation on its availability periods was computed on 14 days, with a step of one minute. For a given peer, the period for which the auto-correlation is maximum gives its best pattern

size. The number of peers with a given best pattern size is plotted on Fig. 1(b), and shows, as could be expected, that the best pattern size is a day, and much further, a week.

2.2 Filtering and Prediction

We implemented a straightforward predictor, that uses a 7-day window of availability history to compute the *daily pattern* of a peer: for each interval of 10 minutes in a day, its value is the number of days in the week where the peer was available during that full interval.

This predictor has two purposes: (1) It should help the application to decide which peers are predictable, and thus, can benefit from an improved quality of service. This gives an incentive to peers to participate regularly to the system; (2) it should help the application to predict future connections and disconnections of the selected peers. To select predictable peers, the predictor computes, for each peer, the maximum and the mean covariance of the peer daily pattern. For this paper, we computed a set, called *predictable set*, containing 19,600 peers whose maximum is at least 5 (prediction threshold), and whose mean covariance is greater than 28 (clear behavior). We also removed the peers whose availability was smaller than 0.1 (useless peers) or greater then 0.9 (they would bias positively our experiments).

For every peer in the predictable set, the predictor predicts that the peer will be online in a given interval if the peer's daily pattern value for that interval is at least 5, and otherwise predicts nothing (we never predict that a peer will be offline). The ratio of successful predictions after a week for the full following week is plotted on Fig. 1(c). It shows that predictions cannot be only explained by accidental availability, and prove the presence of availability patterns in the trace.

We purposely chose a very simple predictor, as we are interested in showing that patterns of presence are visible and can benefit applications, even with a worst-case approach. Therefore, we expect that better results would be achieved using more sophisticated predictors, such as described in [15], and for an optimal pattern size of one day instead of a week.

3 Problem Specification

This section presents two availability matching problems, disconnection matching and presence matching. Each problem is abstracted from the needs of a practical P2P application that we describe afterward. But first, we start by introducing our system model.

3.1 System and Network Model

We assume a fully-connected asynchronous P2P network of N nodes, with N usually ranging from thousands to millions of nodes. We assume that there is a constant bound n_c on the number of simultaneous connections that a peer can engage in, typically much smaller than N. When peers leave the system, they disconnect silently. However, we assume that disconnections are detected after a time Δ_{disc}, for example thirty seconds with TCP keep-alive.

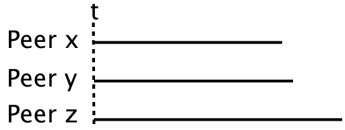
For each peer x, we assume the existence of an availability prediction $Pr^x(t)$, starting at the current time t and for a period T in the future, such that $Pr_x(t)$ is a set of non-overlapping intervals during which x is expected to be online. Since these predictions are based on previous measures of availability for peer x, we assume that such measures are reliable, even in the presence of malicious peers [16, 14].

We note $\bigcup Pr^x(t)$ the set defined by the union of the intervals of $Pr^x(t)$, and $||S||$ the size of a set S.

3.2 The Problem of Disconnection Matching

Intuitively, the problem of *Disconnection Matching* is, for a peer online at a given time, to find a set of other online peers who are expected to disconnect at the same time.

Formally, for a peer x online at time t, an online peer y is a *better match* for Disconnection Matching than an online peer z if $|t^x - t^y| < |t^x - t^z|$, where $[t, t^x[\in Pr^x(t)$, $[t, t^y[\in Pr^y(t)$ and $[t, t^z[\in Pr^z(t)$. The problem of *Disconnection Matching* $DM(n)$ is to discover the n best matches of online peers at anytime.



The problem of disconnection matching arises in applications where a peer tries to find partners with whom it wants to collaborate until the end of its session.

An example of such an application is *task scheduling* in P2P networks. In Zorilla [7], a peer can submit a computation task of n jobs to the system. In such a case, the peer tries to locate n online peers (with expanding ring search) to become partners for the task, and executes the n jobs on these partners. When the computation is over, the peer collects the n results from the n partners. With disconnection matching, such a system becomes much more efficient: by choosing partners who are likely to disconnect at the same time as the peer, the system increases the probability that (1) if the peer does not disconnect too early, its partners will have time to finish executing their jobs before disconnecting and he will be able to collect the results, and (2) if the peer disconnects before the end of the computation, partners will not waste unnecessary resources as they are also likely to disconnect at the same time.

3.3 The Problem of Presence Matching

Intuitively, the problem of *Presence Matching* is, for a peer online at a given time, to find a set of other online peers who are expected to be connected at the same time in the future.

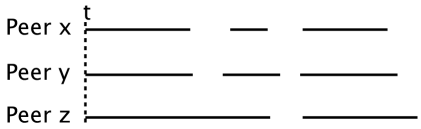
Formally, for a peer x online at time t, an online peer y is a better match for *Unfair Presence Matching* than an online peer z if:

$$||\bigcup Pr^z(t) \cap \bigcup Pr^x(t)|| < ||\bigcup Pr^y(t) \cap \bigcup Pr^x(t)||$$

This problem is qualified as unfair, since peers who are always online appear to be best matches for all other peers in the system, whereas only other always-on peers are best matches for them. Since some fairness is wanted in the system, offline periods should also be considered. Consequently, y is a better match than z for *Presence Matching* if:

$$\frac{||\bigcup Pr^z(t) \cap \bigcup Pr^x(t)||}{||\bigcup Pr^z(t) \cup \bigcup Pr^x(t)|} < \frac{||\bigcup Pr^y(t) \cap \bigcup Pr^x(t)||}{||\bigcup Pr^y(t) \cup \bigcup Pr^x(t)||}$$

The problem of *Presence Matching* $PM(n)$ is to discover the n best matches of online peers at anytime.



The problem of presence matching arises in applications where a peer wants to find partners that will be available at the same time in other sessions. This is typically the case when huge amount of data have to be transferred, and that partners will have to communicate a lot to use that data.

An example of such an application is storage of files in P2P networks [4]. For example, in Pastiche [6], each peer in the system has to find other peers to store its files. Since files can only be used when the peer is online, the best partners for a peer (at equivalent stability) are the peers who are expected to be online when the peer itself is online.

Moreover, in a P2P backup system[8], peers usually replace the replica that cannot be connected for a given period, to maintain a given level of data redundancy. Using presence matching, such applications can increase the probability of being able to connect to all their partners, thus reducing their maintenance cost.

4 Uptime Matching with Epidemic Protocols

We think that epidemic protocols [20, 21, 13] are good approximate solutions for these matching problems. Here, we present one of these protocols, T-Man[12] and, since such protocols rely heavily on appropriate metrics, we propose a metric for each matching problem.

4.1 Distributed Matching with T-Man

T-Man is a well-known epidemic protocol, usually used to associate each peer in the network with a set of good partners, given a metric (distance function) between peers. Even in large-scale networks, T-Man converges fast, and provides a good approximation of the optimal solution in a few rounds, where each round costs only four messages in average per peer.

In T-Man, each peer maintains two small sets, its *random view* and its *metric view*, which are, respectively, some random neighbors, and the current best candidates for partnership, according to the metric in use. During each round, every peer updates its views: with one random peer in its random view, it

merges the two random views, and keeps the most recently seen peers in its random view; with the best peer in its metric view, it merges all the views, and keeps only the best peers, according to the metric, in its metric view.

This double scheme guarantees a permanent shuffle of the random views, while ensuring fast convergence of the metric views towards the optimal solution. Consequently, the choice of a good metric is very important. We propose such metrics for the two availability matching problems in the next part.

4.2 Metrics for Availability Matching

To compute efficiently the distance between peers in T-Man, the prediction $Pr^x(t)$ is approximated by a bitmap of size m, $\mathsf{pred}^{\times}$, where entry $\mathsf{pred}^x[i]$ is 1 if $[i \times T/m, (i+1) \times T/m[$ is included in an interval of $Pr^x(t)$ for $0 \leq i < m$.

4.2.1 Disconnection Matching

The metric computes the time between the disconnections of two peers. In case of equality, the PM-distance of 4.2.2 is used to prefer peers with the same availability periods:

DM-distance$(x, y) = |I^x - I^y|+$ PM-distance(x, y) where
$I^x = min\{0 \leq i < m | \mathsf{pred}^x[i] = 1 \wedge \mathsf{pred}^x[i+1] = 0\}$

4.2.2 Presence Matching

The metric first computes the ratio of co-availability (time where both peers were simultaneously online) on total availability (time where at least one peer was online). Since the distance should be close to 0 when peers are close, we then reverse the value on [0,1]:

$$\text{PM-distance}(x, y) = 1 - \frac{\sum_{0 \leq i < m} min(\mathsf{pred}^x[i], \mathsf{pred}^y[i])}{\sum_{0 \leq i < m} max(\mathsf{pred}^x[i], \mathsf{pred}^y[i])}$$

Note that, while the PM-distance value is in [0,1], the DM-distance value is in [0,m].

5 Simulations and Results

We evaluated the performance of T-Man plus the metrics of Section 4.2, by simulating the two applications of Section 3 on the eDonkey trace of Section 2.

5.1 General Simulation Setup

A simulator was developed from scratch to run the simulations on a Linux 3.2 GHz Xeon computer, for the 19,600 peers of the predictable set from Section 2.2. Their behaviors on 14-days were extracted from the eDonkey trace: the first 7 days were used to compute a prediction, and that prediction, without updates, was used to execute the protocol on the following seven days. During one round of the simulator, all online peers in random order evaluate one T-Man round, corresponding to one minute of the trace. As explained later, both applications were delayed by a period of 10 minutes after a peer would come online to allow

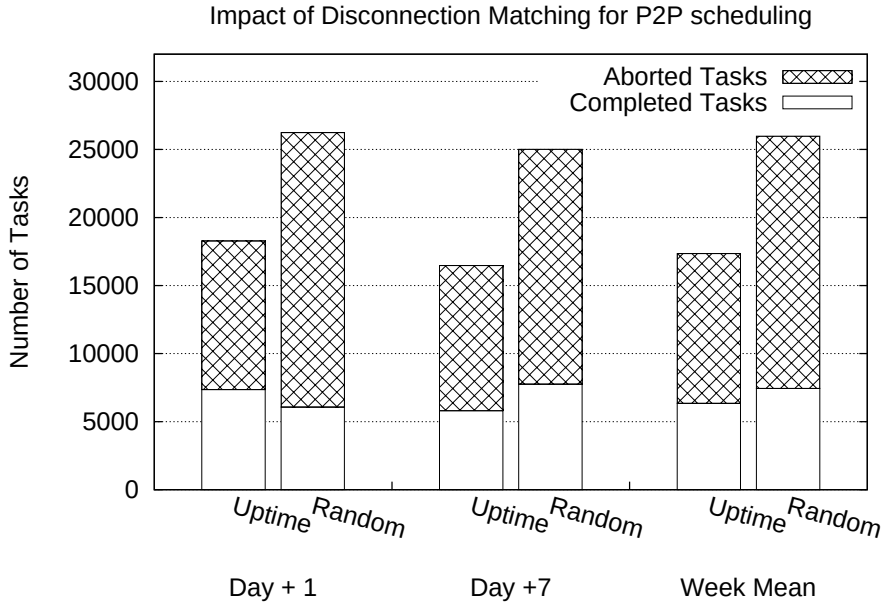


Figure 2: A task is a set of three remote jobs of 4 hours started by every peer, ten minutes after coming online. A task is successful when the peer and its partners are still online after 4 hours to collect the results. Using availability predictions, a peer can decide not to start a task expected to abort, leading to fewer aborted tasks. Using disconnection matching, it can find good partners and it can still complete almost as many tasks as the much more expensive random strategy.

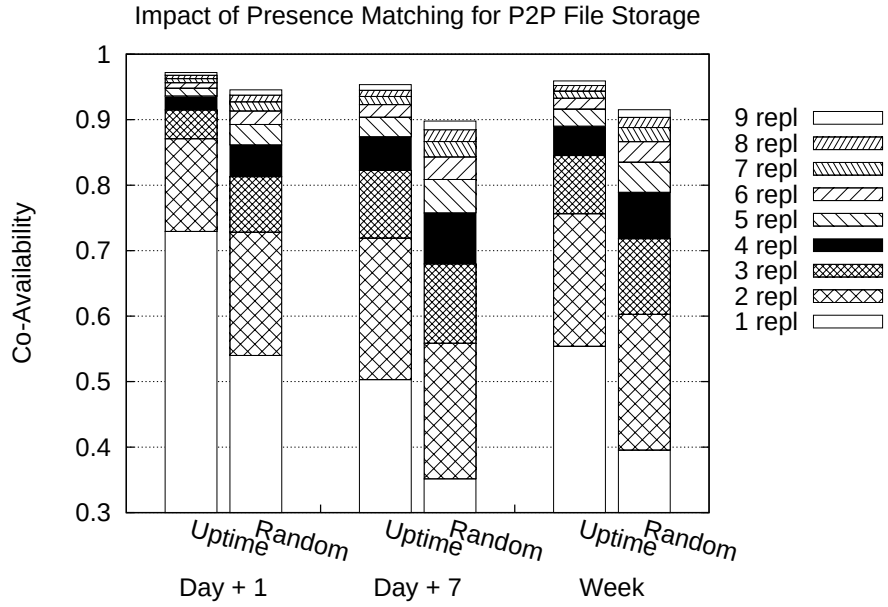


Figure 3: 10 minutes after coming online for the first time, each peer creates a given number of replica for its data. Co-availability is defined by the simultaneous presence of the peer and at least one replica. Using presence matching, fewer replicas are needed to achieve better results than using a random choice of partners. Even the 7th day, using a 6-day old prediction, the system still performs much more efficiently, almost compensating the general loss in availability.

T-Man to provide a useful metric view. The computation of a complete run did not exceed two hours and 6 GB of memory footprint.

5.2 Evaluation of Disconnection Matching

The task scheduling application of Section 3.2 was simulated to evaluate the performance of T-Man and the DM-distance metric. In the simulations, every peer started a task after 10 minutes online: a task ran three jobs of 4 hours on remote partners, and was *completed* if the peer and its partners were still online after 4 hours to collect the results. A peer could decide not to start a task if the prediction of its own availability forecast that he would go offline before completion of the task. The number of aborted/completed tasks is plotted on Fig. 2, for the first day, the seventh day and the whole week for either disconnection matching (uptime) or random choice (random peer chosen in T-Man random view).

Prediction of availability decreased a lot the number of aborted tasks, and, with fewer started tasks, disconnection matching completed almost the same number of tasks as random matching, even over the full week, when the prediction was supposed to be less accurate (see auto-correlation in Section 2.1).

5.3 Evaluation of Presence Matching

The P2P file storage of Section 3.3 was also simulated with T-Man and the PM-distance metric. Every peer replicated its data on its partners, ten minutes after coming online for the first time, in the hope of using its remote data the next time it would be online. The co-availability of the peer and at least one replica is plotted on Fig. 3, for different number of replicas.

Using presence matching, fewer replicas were needed to achieve better results than using a random choice of partners. As in the previous simulations, week-old predictions performed still better than random choice.

6 Discussion and Conclusion

In this paper, we showed that epidemic protocols for topology management can be efficient to find good partners in availability-aware networks. Simulations proved that, using one of these protocols and appropriate metrics, such applications can be less expensive and still perform with an equivalent or better quality of service. We used a worst-case scenario: a simple predictor, and a trace collected from a highly volatile file-sharing network, where only a small subset of peers provide predictable behaviors. Consequently, we expect that a real application would take even more benefit from availability matching protocols.

In particular, until this work, availability-aware applications were limited to using predictions or availability information to better choose among a limited set of neighbors. This work opens the door to new availability-aware applications, where best partners are chosen among all available peers in the network. It is a useful complement to the work done on measuring availability[16, 14] and using these measures to predict future availability[15].

References

[1] Trace. http://fabrice.lefessant.net/traces/edonkey2.

[2] Ranjita Bhagwan, Stefan Savage, and Geoffrey Voelker. Understanding availability. In *IPTPS, Int'l Work. on Peer-to-Peer Systems*, 2003.

[3] Ranjita Bhagwan, Kiran Tati, Yu-Chung Cheng, Stefan Savage, and Geoffrey M. Voelker. Total recall: system support for automated availability management. In *NSDI, Symp. on Networked Systems Design and Implementation*, 2004.

[4] Jean-Michel Busca, Fabio Picconi, and Pierre Sens. Pastis: A highly-scalable multi-user peer-to-peer file system. In *Euro-Par*, 2005.

[5] Byung-Gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon, M. Frans Kaashoek, John Kubiatowicz, and Robert Morris. Efficient replica maintenance for distributed storage systems. In *NSDI, Symp. on Networked Systems Design and Implementation*, 2006.

[6] Landon P. Cox, Christopher D. Murray, and Brian D. Noble. Pastiche: Making backup cheap and easy. In *OSDI, Symp. on Operating Systems Design and Implementation*, 2002.

[7] Niels Drost, Rob V. van Nieuwpoort, and Henri E. Bal. Simple locality-aware co-allocation in peer-to-peer supercomputing. In *GP2P, Int'l Work. on Global and Peer-2-Peer Computing*, 2006.

[8] Alessandro Duminuco, Ernst W Biersack, and Taoufik En Najjary. Proactive replication in distributed storage systems using machine availability estimation. In *CoNEXT, Int'l Conf. on emerging Networking EXperiments and Technologies*, 2007.

[9] P. Brighten Godfrey, Scott Shenker, and Ion Stoica. Minimizing churn in distributed systems. In *SIGCOMM, Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2006.

[10] Saikat Guha, Neil Daswani, and Ravi Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *IPTPS, Int'l Work. on Peer-to-Peer Systems*, 2006.

[11] Sidath B. Handurukande, Anne-Marie Kermarrec, Fabrice Le Fessant, Laurent Massoulié, and Simon Patarin. Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. In *EuroSys*, 2006.

[12] Mark Jelasity and Ozalp Babaoglu. T-man: Gossip-based overlay topology management. In *ESOA, Intl'l Work. on Engineering Self-Organising Systems*, 2005.

[13] Marc-Olivier Killijian, Ludovic Courtès, and David Powell. A Survey of Cooperative Backup Mechanisms. Technical Report 06472, LAAS, 2006.

[14] Fabrice Le Fessant, Cigdem Sengul, and Anne-Marie Kermarrec. Pacemaker: Fighting Selfishness in Availability-Aware Large-Scale Networks. Technical Report RR-6594, INRIA, 2008.

[15] James W. Mickens and Brian D. Noble. Exploiting availability prediction in distributed systems. In *NSDI, Symp. on Networked Systems Design and Implementation*, 2006.

[16] Ramses Morales and Indranil Gupta. AVMON: Optimal and scalable discovery of consistent availability monitoring overlays for distributed systems. In *ICDCS, Int'l Conf. on Distributed Computing Systems*, 2007.

[17] Jan Sacha, Jim Dowling, Raymond Cunningham, and René Meier. Discovery of stable peers in a self-organising peer-to-peer gradient topology. In *DAIS, Int'l Conf. on Distributed Applications and Interoperable Systems*, 2006.

[18] S. Saroiu, P. Krishna Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *MMCN, Multimedia Computing and Networking*, 2002.

[19] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *IMC, Internet Measurement Conf.*, 2006.

[20] Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. CYCLON: Inexpensive membership management for unstructured P2P overlays. *J. Network Syst. Manage.*, 13(2), 2005.

[21] Spyros Voulgaris and Maarten van Steen. An epidemic protocol for managing routing tables in very large peer-to-peer networks. In *DSOM, Int'l Work. on Distributed Systems: Operations and Management,*, 2003.

[22] Qin Xin, Thomas Schwarz, and Ethan L. Miller. Availability in global peer-to-peer storage systems. In *WDAS, Work. on Distributed Data and Structures*, 2004.

Contents

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399