Finding Good Partners in Availability-aware P2P Networks

Stevens Le Blond[1], Fabrice Le Fessant[2], Erwan Le Merrer[3]*

[1] INRIA Sophia Antipolis,
stevens.le_blond@inria.fr,
[2] INRIA Saclay,
fabrice.le_fessant@inria.fr,
[3] INRIA Rennes,
elemerre@irisa.fr

We study the problem of finding peers matching a given availability pattern in a peer-to-peer (P2P) system. 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; we provide corresponding metrics for both matching problems. We evaluated this solution by simulating two P2P applications, *task scheduling* and *file storage*, over a new trace of the eDonkey network, the largest available with availability information. We first proved the existence of regularity patterns in the sessions of 14M peers over 27 days. We also showed that, using only 7 days of history, a simple predictor could select predictable peers and successfully predicted their online periods for the next week. Finally, 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 performance and stability.

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

* Supported by project P2Pim@ges, of the French Media & Networks cluster

shown that, for many peers, these events globally obey some availability patterns ([21, 22, 2]), and so, can be predicted from the uptime history of those peers [18].

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. Basic usage of prediction based on node availability exists in the literature, as *e.g.* for file replication [16].

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 2.

We then propose to use standard epidemic protocols for topology management to solve these problems (see *e.g.* [12, 24]); such protocols have proven to be efficient for a large panel of applications, from overlay slicing [13] to IP-TV overlay maintenance [14] for example. However, in order to converge to the desired state or topology (here matched peers), those protocols require good metrics to compute the distance between peers. Such metrics and a well known epidemic protocol, T-Man [12], are described in Section 3.

To evaluate the efficiency of our proposal, we simulated an application for each matching problem: an application of *task scheduling*, where tasks of multiple remote 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). These applications are specified in Section 5.

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 4, 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. The new eDonkey trace and this simple predictor are studied in Section 4.

Our simulation results showed 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. These results are detailed in Section 6.

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, 20, 5, 25]. Close to our work, Godfrey et al. [9] show that strategies based on the longest current uptime are more efficient than uptime-agnostic strategies for replica placement; Mickens et al. [18] introduce 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 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.

2.1 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 30 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 [19, 17].

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.

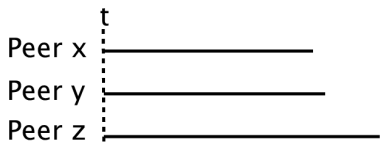

Fig. 1. Disconnection Matching: peer y is a better match than peer z for peer x.

2.2 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 typically arises in applications where a peer tries to find partners with whom it wants to collaborate until the end of its session, in particular when starting such a collaboration might be expensive in terms of resources.

An example of such an application is *task scheduling* in P2P networks. In Zorilla [7] for example, 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:

- 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;
- 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.

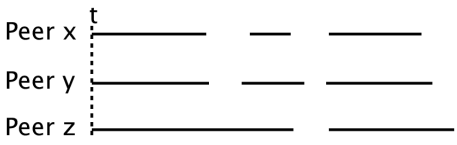

Fig. 2. Presence Matching: peer y is a better match than peer z for peer x.

2.3 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 called *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 most P2P systems, 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.

3 Uptime Matching with Epidemic Protocols

We think that epidemic protocols [12, 23, 15, 24] 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.

3.1 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.

3.2 Metrics for Availability Matching

To compute efficiently the distance between peers, the prediction $Pr^x(t)$ is approximated by a bitmap of size m, pred^x, 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 $0 \leq i < m$. Note that these metrics can be used with any epidemic protocol, not only with T-Man.

Disconnection Matching The metric computes the time between the disconnections of two peers. In case of equality, the PM-distance of 3.2 is used to prefer peers with the same availability periods:

DM-distance$(x, y) = |I^x - I^y|+$ PM-distance(x, y) where

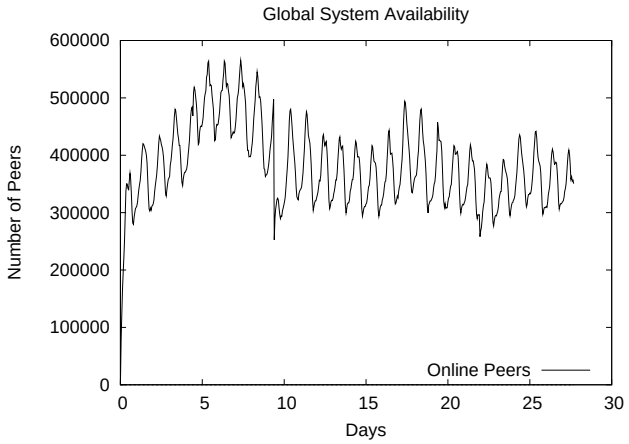

Fig. 3. Diurnal patterns are clearly visible when we plot the number of online peers at any time in our 27-day eDonkey trace. Depending on the time of the day, between 300,000 and 600,000 users are connected to a single eDonkey server.

$$I^x = min\{0 \leq i < m | \mathsf{pred}^x[i] = 1 \wedge \mathsf{pred}^x[i+1] = 0\}$$

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].

4 Simulation Settings

We evaluated our a solution based on T-Man on two applications, one for each matching problem. In this section, we describe our simulation settings. In particular, we describe the characteristics of the trace we collected for the needs of this study, with more than 300,000 online peers on 27 days. With a few thousand peers online at the same time, most other traces collected on P2P systems [21, 10, 2] lack massive connection and disconnection trends, for the study of availability patterns on a large scale.

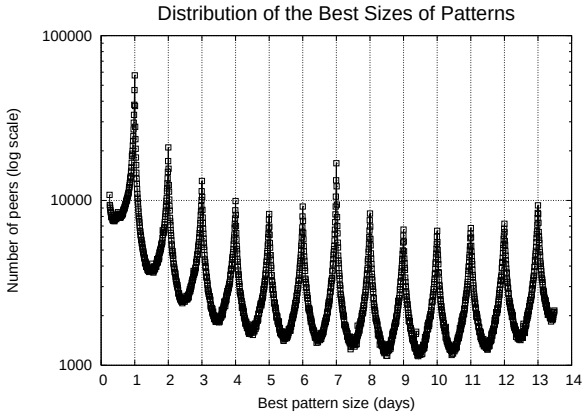

Fig. 4. Peers achieve their best auto-correlation (ressemblance between sessions after a given period) between sessions for a one-day period or a one-week period. Consequently, peers are highly likely to connect at almost the same time the next day or the next week.

4.1 A new eDonkey Trace

In 2007, we collected the connection and disconnection events from the logs of one of the main eDonkey servers in Europe. Edonkey is currently the most used P2P file-sharing network in the world. 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. 3. 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. 4, and shows, as could be expected, that the best pattern size is a day, and much further, a week.

4.2 Filtering and Prediction

Our goal in these simulations was to evaluate the efficiency of our matching protocol, and not the efficiency of availability predictors, as already done in [18]. As a consequence, we implemented a very 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:

$$pattern^p[i] = \Sigma_{d \in [0:6]} history^p[d * 24 * 60/10 + i]$$

This predictor has two purposes:

- It should help the application to decide which peers are predictable, and thus, which peers can benefit from an improved quality of service. This gives an incentive for peers to participate regularly to the system;
- 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 these simulations, we computed a set, called *predictable set*, containing peers matching with the following properties:

- The maximum value in *pattern* is at least 5: each peer was available at least five days during the last week exactly at the same time;
- The average covariance in *pattern* is greater than 28: each peer has a sharply-shaped behavior;
- Peer availability is greater than 0.1: peers have to contribute enough to the system;
- Peer availability is smaller than 0.9: peers which are always online would bias positively our simulations.

In our eDonkey trace, this predictable set contains 19,600 such peers. Note that this relatively small amount of peers, *w.r.t.* the total number of peers in our trace, does not mean that eDonkey peers are not predictable: our trace concerns only a part of eDonkey users at measure time (around 10%, those connected to eDonkey Server N.2). Users that leave may join another server (*e.g.* Server N.1, a larger one), which makes them invisible in our trace, even though they are still using eDonkey. For every peer in the 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

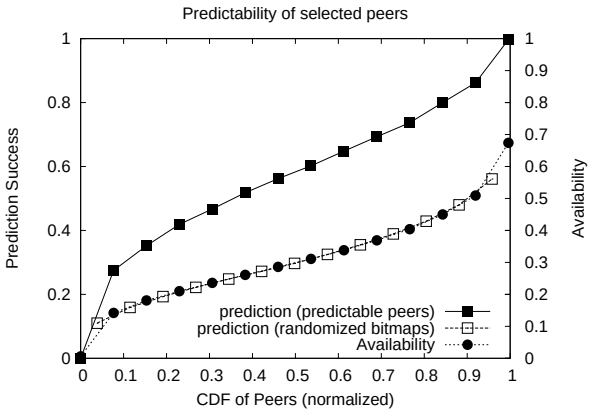

Fig. 5. Whereas availability determines the prediction with random bitmaps, daily patterns improve the prediction with real bitmaps (*e.g.* for 60% of peers (x=0.4), 50% of predictions (y=0.5) are successful, but only 25% with random bitmaps).

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. 5. 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 [18], and for an optimal pattern size of one day instead of a week.

4.3 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 4.2. 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 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 Simulated Applications

In this section, we describe the two applications that we used to illustrate the need for an efficient protocol for distributed availability matching. Our goal is not to improve the performance of these applications, as this can be done by an aggressive greedy algorithm, but to save resources using availability information.

5.1 Disconnection Matching: Task Scheduling

To evaluate the efficiency of T-Man and the DM-distance metric, we simulated a distributed task scheduling application. In this application, every peer starts a task after 10 minutes online: a task is composed of 3 jobs of 4 hours on remote partners, and is *completed* if the peer and its partners are still online after 4 hours to collect the results.

The 2 first hours of each job are devoted to the download of the data needed for the computation from a central server. As a consequence, a peer can decide not to start a task to save the bandwidth of the central server. In our simulation, such a decision is taken when the prediction of the peer availability shows that the peer is going to go offline before completion of the task.

5.2 Presence Matching: P2P File-Storage

To evaluate the efficiency of T-Man and the PM-distance metric, we simulated a P2P file storage application. In this application, every peer replicates its data to its partners, ten minutes after coming online for the first time, in the hope that he will be able to use this remote data the next time it will be online.

The size of the data of each peer is supposed to be large, hundred of megabytes of example. As a consequence, it is important for the system to use as little redundancy as possible to achieve high co-availability of data (i.e. availability of the peer and at least one of its data replica). Finding good partners in the network is expected to provide replica which are more likely to be available at the same time as the peer, thus decreasing the need for more replicas.

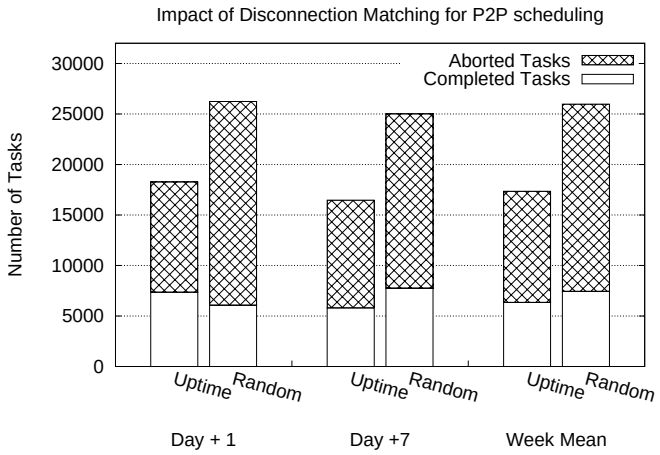

Fig. 6. A task is a set of three remote jobs of 4 hours started by every peer, ten minutes after coming online. A task is successful when the peer and its partners are still online after 4 hours to collect the results. Using availability predictions, a peer can decide not to start a task expected to 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.

6 Simulation Results

In this section, we present the results of our simulations of the two applications. We are not interested in the raw performance of these applications, but in the savings that could be achieved by using availability information and partner matching.

6.1 Results for Disconnection Matching

We compared Disconnection Matching with a Random choice of partners (actually, using partners within T-Man random view) for the distributed task scheduling application. The number of completed tasks and the number of aborted tasks are plotted on Fig. 6, for the first day, the 7^{th} day and the whole week.

Prediction of availability decreased by 68% the number of aborted tasks on average over a week, corresponding to 50% of bandwidth savings on the data server, while decreasing the number of completed tasks by only 17%.

These results were largely improved using one-day prediction, since one-week prediction is expected to be less accurate (see auto-correlation

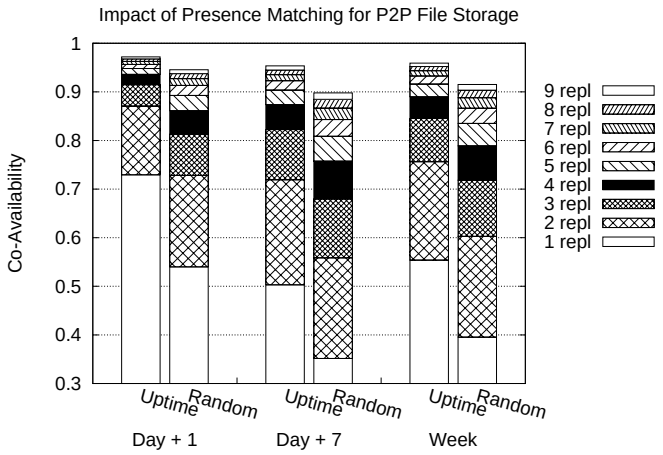

Fig. 7. 10 minutes after coming online for the first time, each peer creates a given number of replica for its data. Co-availability is defined by the simultaneous presence of the peer and at least one replica. Using presence matching, fewer replicas are needed to achieve better results than using a 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.

in Section 4.1). Indeed, bandwidth savings were about 43% for Disconnection Matching, while completing 20% more tasks. Thus, it is much more interesting from a performance point of view to use one-day prediction every day instead of one-week prediction, although savings are still possible with one-week predictions.

6.2 Results for Presence Matching

We compared Presence Matching with a Random choice of replica locations for the P2P file-system application. The co-availability of the peer and at least one replica is plotted on Fig. 7, for different number of replicas.

Using presence matching, fewer replicas were needed to achieve better results than using a random choice of partners. For example, 1 replica with Presence Matching gives a better co-availability than 2 replicas with Random Choice; 5 replicas with Presence Matching give a co-availability of 95% which is only achieved using 9 replicas with Random Choice. As for the other application, week-old predictions performed still better than random choice in the same orders.

7 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[19, 17] and using these measures to predict future availability[18].

References

1. Trace. http://fabrice.lefessant.net/traces/edonkey2.
2. Bhagwan, R., Savage, S., and Voelker, G. Understanding availability. In *IPTPS, Int'l Work. on Peer-to-Peer Systems* (2003).
3. Bhagwan, R., Tati, K., Cheng, Y.-C., Savage, S., and Voelker, G. M. Total recall: system support for automated availability management. In *NSDI, Symp. on Networked Systems Design and Implementation* (2004).
4. Busca, J.-M., Picconi, F., and Sens, P. Pastis: A highly-scalable multi-user peer-to-peer file system. In *Proceedings of Euro-Par* (2005).
5. Chun, B.-G., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, M. F., Kubiatowicz, J., and Morris, R. Efficient replica maintenance for distributed storage systems. In *NSDI, Symp. on Networked Systems Design and Implementation* (2006).
6. Cox, L. P., Murray, C. D., and Noble, B. D. Pastiche: Making backup cheap and easy. In *OSDI, Symp. on Operating Systems Design and Implementation* (2002).
7. Drost, N., van Nieuwpoort, R. V., and Bal, H. E. Simple locality-aware co-allocation in peer-to-peer supercomputing. In *GP2P, Int'l Work. on Global and Peer-2-Peer Computing* (2006).
8. Duminuco, A., Biersack, E. W., and En Najjary, T. Proactive replication in distributed storage systems using machine availability estimation. In *CoNEXT, Int'l Conf. on emerging Networking EXperiments and Technologies* (2007).
9. Godfrey, P. B., Shenker, S., and Stoica, I. Minimizing churn in distributed systems. In *SIGCOMM, Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications* (2006).
10. Guha, S., Daswani, N., and Jain, R. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *IPTPS, Int'l Work. on Peer-to-Peer Systems* (2006).

11. Handurukande, S. B., Kermarrec, A.-M., Le Fessant, F., Massoulié, L., and Patarin, S. Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. In *EuroSys* (2006).
12. Jelasity, M., and Babaoglu, O. T-man: Gossip-based overlay topology management. In *ESOA, Intl'l Work. on Engineering Self-Organising Systems* (2005).
13. Jelasity, M., and Kermarrec, A.-M. Ordered slicing of very large-scale overlay networks. *IEEE International Conference on Peer-to-Peer Computing* (2006), 117–124.
14. Kermarrec, A.-M., Le Merrer, E., Liu, Y., and Simon, G. Surfing peer-to-peer iptv: Distributed channel switching. *Proceedings of Euro-Par* (2009).
15. Killijian, M.-O., Courtès, L., and Powell, D. A Survey of Cooperative Backup Mechanisms. Tech. Rep. 06472, LAAS, 2006.
16. Kim, K. Lifetime-aware replication for data durability in p2p storage network. *IEICE Transactions 91-B*, 12 (2008), 4020–4023.
17. Le Fessant, F., Sengul, C., and Kermarrec, A.-M. Pacemaker: Fighting Selfishness in Availability-Aware Large-Scale Networks. Tech. Rep. RR-6594, INRIA, 2008.
18. Mickens, J. W., and Noble, B. D. Exploiting availability prediction in distributed systems. In *NSDI, Symp. on Networked Systems Design and Implementation* (2006).
19. Morales, R., and Gupta, I. AVMON: Optimal and scalable discovery of consistent availability monitoring overlays for distributed systems. In *ICDCS, Int'l Conf. on Distributed Computing Systems* (2007).
20. Sacha, J., Dowling, J., Cunningham, R., and Meier, R. 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).
21. Saroiu, S., Gummadi, P. K., and Gribble, S. A measurement study of peer-to-peer file sharing systems. In *MMCN, Multimedia Computing and Networking* (2002).
22. Stutzbach, D., and Rejaie, R. Understanding churn in peer-to-peer networks. In *IMC, Internet Measurement Conf.* (2006).
23. Voulgaris, S., Gavidia, D., and van Steen, M. CYCLON: Inexpensive membership management for unstructured P2P overlays. *J. Network Syst. Manage. 13*, 2 (2005).
24. Voulgaris, S., van Steen, M., and Iwanicki, K. Proactive gossip-based management of semantic overlay networks: Research articles. *Concurr. Comput. : Pract. Exper. 19*, 17 (2007), 2299–2311.
25. Xin, Q., Schwarz, T., and Miller, E. L. Availability in global peer-to-peer storage systems. In *WDAS, Work. on Distributed Data and Structures* (2004).