Choosing Partners Based on Availability in P2P Networks

STEVENS LE BLOND, INRIA Sophia Antipolis/INRIA Rhône-Alpes
FABRICE LE FESSANT, INRIA Saclay - Ile de France
ERWAN LE MERRER, Technicolor

Availability of applications or devices is known to be one of the most critical variables impacting the performances of software systems. We study in this article 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 one 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. This solution is purely distributed as it does not rely on any central server or oracle to operate. 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.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Computer Systems Organization**]: Performance of Systems—*Reliability, availability, and serviceability*

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Distributed peer matching, gossip, availability prediction

ACM Reference Format:
Le Blond, S., Le Fessant, F., and Le Merrer, E. 2012. Choosing partners based on availability in P2P networks. ACM Trans. Autonom. Adapt. Syst. 7, 2, Article 25 (July 2012), 14 pages.
DOI = 10.1145/2240166.2240175 http://doi.acm.org/10.1145/2240166.2240175

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 [Godfrey et al. 2006]. Fortunately, it has been shown that, for many peers, these events globally obey some availability patterns [Saroiu et al. 2002; Stutzbach and Rejaie 2006; Bhagwan et al. 2003], and so, can be predicted from the uptime history of those peers [Mickens and Noble 2006].

A preliminary version of this article appeared in the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS09) [Le Blond et al. 2009].

Authors' addresses: S. Le Blond (corresponding author), INRIA Sophia Antipolis/INRIA Rhône-Alpes, France; email: stevens.le_blond@inria.fr; F. Le Fessant, INRIA Saclay—Ile de France; E. Le Merrer, Technicolor.

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, for example, for file replication [Kim 2008].

In this article, 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 propose to use standard epidemic protocols for topology management to solve these problems (see, for example, Jelasity and Babaoglu [2005] and Voulgaris et al. [2007]); such protocols have proven to be efficient for a large panel of applications, from overlay slicing [Jelasity and Kermarrec 2006] to IP-TV overlay maintenance [Kermarrec et al. 2009] 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; this allows each peer to be able to make preferential choices of connections among a pool of possible peers. Such metrics and a well-known epidemic protocol, T-Man [Jelasity et al. 2009; Jelasity and Babaoglu 2005], are described in Section 4.

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 6.

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 5, 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 5.

Our simulation results showed that our epidemic-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, epidemic protocols for peer matching are scalable and inexpensive, making the solution usable for any application and network size. These results are detailed in Section 7.

2. RELATED WORK

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 [Bhagwan et al. 2004; Duminuco et al. 2007; Sacha et al. 2006; Chun et al. 2006; Xin et al. 2004; Pamies-Juarez et al. 2008]. For example, in Duminuco et al. [2007], a control-based approach is used to smooth the bandwidth used to maintain the replication ratio of some data; this is done by computing the global availability of peers (obtained from an oracle), in order to avoid bursts of reactive repairs. While well-known distributed storage systems [Dabek et al. 2001; Bhagwan et al. 2004; Chun et al. 2006] store replicas on peers regardless of their availabilities, papers [Godfrey et al. 2006; Pamies-Juarez et al. 2008] show that strategies based on the longest current uptimes are more efficient than uptime-agnostic strategies for replica placement. A recent work [Kondo et al. 2008] shows the interest of forming clusters of peers according to some proximity

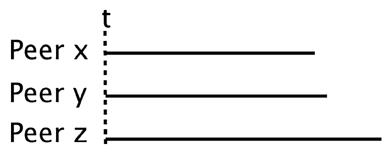

Fig. 1. Disconnection matching: peer y is a better match than peer z for peer x.

metrics (availability or speed); optimal clusters are computed in a centralized way for proof of concept. To the best of our knowledge, our proposal is the first to deal with the problem of practically finding the best partners according to availability patterns of peers in a large-scale network.

On the side of pure prediction of availability, paper Mickens and Noble [2006] introduces sophisticated predictors and shows that they can be very successful. Such a work on efficient predictors is complementary to our proposal, since we only introduce a basic predictor for illustrative purposes. Our approach can then take as an input any predictor, including theirs, adapted to the particular needs of the targeted application.

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. We first start by introducing our system model.

3.1. System and Network Model

We assume a synchronous and weakly connected P2P network of N nodes, with N usually ranging from thousands to millions of nodes. There is a constant bound on the number of simultaneous connections that a peer can engage in, generally much smaller than N (typically around $\log N$ for scalability constraints). When peers leave the system, they disconnect silently, according to the fail-stop model. However, disconnections are detected after a bounded 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 nonoverlapping 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 [Morales and Gupta 2007; Le Fessant et al. 2008].

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 $[t, t^z[\in Pr^z(t)$; Figure 1 presents an example. 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 [Drost et al. 2006] 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.

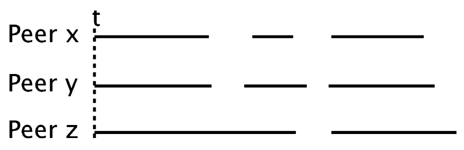

Fig. 2. Presence matching: peer *y* is a better match than peer *z* for peer *x*.

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.

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

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

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 (mostly to provide incentives for peers to participate more), offline periods should also be considered. Consequently, y is a better match than z for *presence matching* if

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

An example is depicted on Figure 2.

In practice, such a temporal synchronization (often named *correlated availability*) between peers, for example, occurs due to day/night connectivity behaviors [Kondo et al. 2008].

The problem of *presence matching* $PM(n)$ is to discover the n best matches of online peers at anytime. This 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 [Busca et al. 2005]. For example, in Pastiche [Cox et al. 2002], 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 [Duminuco et al. 2007], peers usually replace the replicas that cannot be contacted 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 [Jelasity and Babaoglu 2005; Voulgaris et al. 2005; Killijian et al. 2006; Voulgaris et al. 2007] are good approximate solutions for these matching problems. Their purely distributed nature has shown to provide scalability, fast convergence, and resilience to churn. Here, we present one of these protocols, namely T-Man [Jelasity et al. 2009; Jelasity and Babaoglu 2005] 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 (disorder decreases exponentially fast), and provides a good approximation of the optimal solution in a few rounds, where each round costs only four messages per peer on average.

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, the prediction $Pr^x(t)$ up to time 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 $0 \leq i < m$. In other words, each bit in the bitmap indicates that a peer will be available for a given unit of time if that peer has been predicted as available for the entire duration of that unit.

To update distances among its neighbors, a peer computes its own bitmap, and the distance (according to some metric) between that bitmap and its neighbors' received through T-Man. The frequency at which to update the bitmap depends on the implementation and the considered system; however, we will see in Section 5.1 that peers are the most predictable for T equals to 1 and 7 days in our case. At this frequency, updating bitmaps is computationally inexpensive and comes with no additional communication cost.

We insist that the metrics presented after can be used with any epidemic protocol, not only with T-Man, and the whole process of distance computation and matching is done on a distributed fashion, as opposed to a centralized computation (or oracle) based on the constant observation of the whole system, as done in the literature (see, for example, Duminuco et al. [2007] and Kondo et al. [2008]); this thus provides a practical solution to the problem.

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.

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

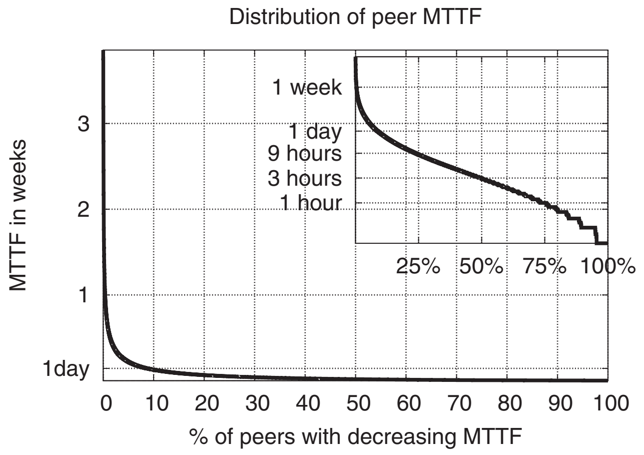

Fig. 3. MTTF distribution on the filtered trace shows a median of 3 hours and an average of 9 hours.

4.2.2. Presence Matching. The metric first computes the ratio of coavailability (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].

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

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

5. SIMULATION SETTINGS

We evaluated our 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 [Saroiu et al. 2002; Guha et al. 2006; Bhagwan et al. 2003] lack massive connection and disconnection trends, for the study of availability patterns on a large scale.

5.1. 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 Web-site [Repository 2010], contains more than 200 millions of connections by more than 14 millions of peers, over a period of 27 days. To analyze 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.

5.1.1. Statistics. We computed some general statistics about the behavior of peers in the filtered trace. Figure 3 plots the distribution of Mean Time To Failure (MTTF) of peers in the trace. It shows that peer sessions are long (9 hours on average), but with a high variance: 50% of the peers have a MTTF of less than 3 hours. We explain this by the fact that eDonkey is a file-sharing network specialized in movie files, and downloading such files can take hours. A typical session will start by a short search for files, and then, depending on the results, the user will either disconnect, if disappointed by the results, or will stay until the end of the download.

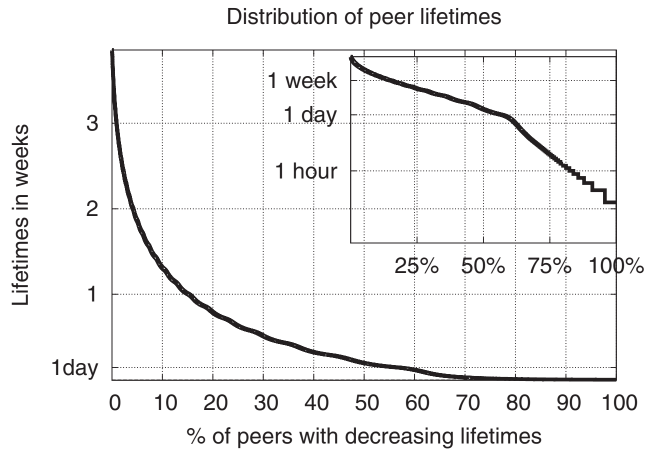

Fig. 4. Distribution of lifetimes (time between the first and the last connection) shows a median of around 1 day and an average of 3 days. Majority of the peers do not appear long enough to be predictable, but more than 15% of peers appear on more than one week, and are then more likely to be predictable.

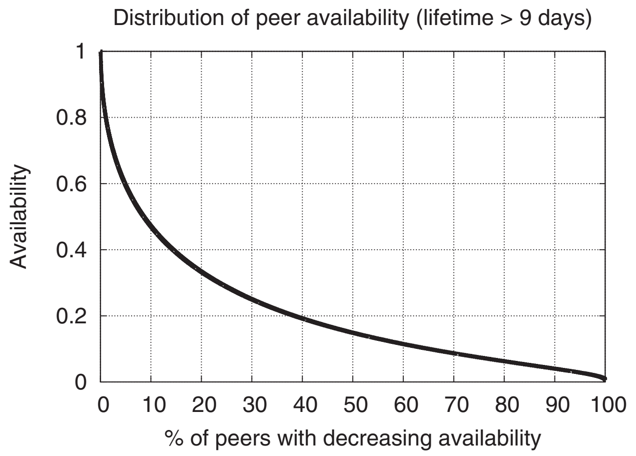

Fig. 5. Average availability of the more active peers (for a lifetime of at least 10 days) is near the median (20%).

Figure 4 plots the distribution of user lifetimes, where the lifetime of a user is defined as the delay between its first connection and its last disconnection for all its sessions. It shows similar results, with an average lifetime of 3 days and a median lifetime of 1 day. This is typical of a network where many users connect just to try the system for a short while and never come back, while a few are satisfied and remain in the system for a long time. On the figure, only 15% of users appear on more than 1 week: such users are very interesting, as one week is a good learning period for prediction algorithms, as we will show later.

Figure 5 plots the distribution of availability for stable peers, where stable peers are defined here as those with a lifetime longer than 10 days. It shows that stable peers have a good availability, with an average of 20% (4 to 5 hours a day). This result is also interesting, as it shows that the subnetwork restricted to stable peers could be used to run P2P applications that require a higher availability than the one traditionally assumed in P2P networks.

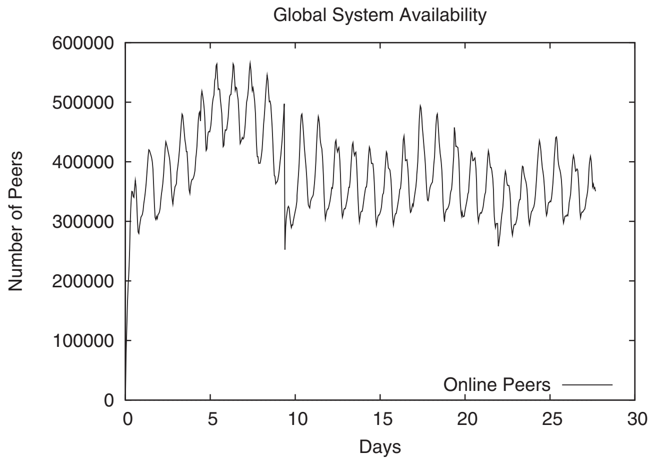

Fig. 6. Diurnal patterns are clearly visible when we plot the number of online peers at any time in our 27-day eDonkey trace. Depending on the time of the day, between 300,000 and 600,000 users are connected to a single eDonkey server.

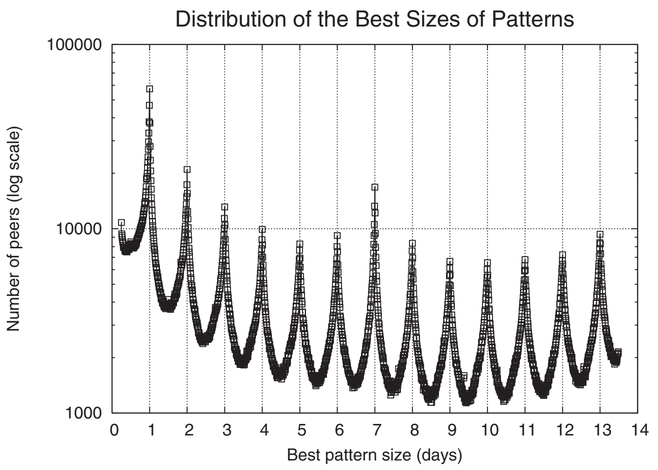

Fig. 7. Peers achieve their best auto-correlation (similarity between sessions after a given period) between sessions for a one-day period or a one-week period. Consequently, peers are highly likely to connect at almost the same time the next day or the next week.

The number of peers online at the same time in the filtered trace is usually more than 300,000, as shown by Figure 6. Global diurnal patterns of around 100,000 users are also clearly visible: as shown by previous studies [Handurukande et al. 2006], 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 Figure 7, and shows, as could be expected, that the best pattern size is a day, and much further, a week.

5.1.2. Limitations. Because eDonkey uses a distributed architecture, our trace captures a significant but partial view of the whole eDonkey network. Indeed, the eDonkey file-sharing system relies on a few big servers counting between 500K and 1M simultaneous peers each, a hundred of smaller servers, and a Distributed Hash Table (DHT) based on Kademlia. The trace we exploited in this article comes from the eDonkey Server N.2, the second largest eDonkey server at the time of the measure with up to 560,000 simultaneous peers. As eDonkey counts around 5 millions simultaneous peers, our trace has two limitations.

First, our trace captures a fraction of all the eDonkey network. Although this trace exceeds previous traces by several orders of magnitude, we still see only about 10% of all the eDonkey network at any given point in time. Hence, our trace is a partial measure of the availability in the eDonkey network.

The second limitation follows from this partial coverage. As an eDonkey client connects to a single server at launch time, we have no guarantee that a user will always connect to our instrumented server. eDonkey users generally try to connect to the most popular servers first, so the less popular servers are only fully used when a big server crashes. As we instrumented the second most popular server, some users must have always connected to it but others probably first tried to connect to the most popular one. As a result, our trace also gives a partial measure of each peer's availability.

Clearly, this partial measure is a worst-case scenario for our analysis because it makes it harder to predict the peers' availability. In the following section, we filter our trace to keep the set of peers that connected often enough to our instrumented server to have a representative measure of their availability.

5.2. 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 Mickens and Noble [2006]. 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 (to compensate for previous removal of very poorly available nodes).

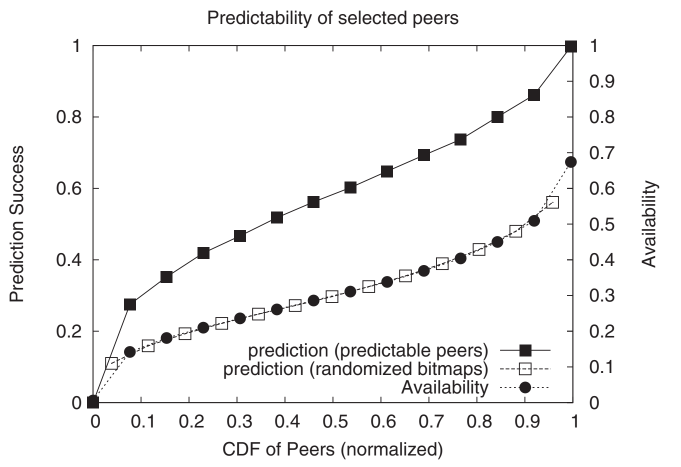

Fig. 8. Whereas availability determines the prediction with random bitmaps, daily patterns improve the prediction with real bitmaps (e.g., for 60% of peers (x = 0.4), 50% of predictions (y = 0.5) are successful, but only 25% with random bitmaps).

In our eDonkey trace, this predictable set contains 19,600 such peers. Note that this relatively small amount of peers, with respect to the total number of peers in our trace, does not mean that eDonkey peers are not predictable. Instead, we consider a relatively small set of peers because our trace captures only a fraction of all the eDonkey peers.

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 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 Figure 8. It shows that predictions cannot be only explained by accidental availability, and proves 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 Mickens and Noble [2006], and for an optimal pattern size of one day instead of a week.

5.3. 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 5.2. 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 the complete simulation did not exceed two hours and 6GB of memory footprint.

6. 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.

6.1. 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.

6.2. 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, hundreds of megabytes for example. As a consequence, it is important for the system to use as little redundancy as possible to achieve high coavailability 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 replicas which are more likely to be available at the same time, thus decreasing the need for more replicas that would mask transient unavailabilities. The same logic also applies in the case of file storage using erasure or regenerating codes instead of basic replicas [Duminuco and Biersack 2009], as reliability is improved through stable placement of redundancy.

7. SIMULATION RESULTS

In this section, we present the results of 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.

7.1. Results for Disconnection Matching

We compared disconnection matching with a random choice of partners (actually, using partners within the random view of T-Man) for the distributed task scheduling application. The number of completed tasks and the number of aborted tasks are plotted on Figure 9, 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 in Section 5.1). 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.

Before a practical deployment of our proposal, it is thus of interest for a system administrator to determine the recurrent patterns of availabilities that could be observed in the system (as analyzed in Section 5.1), so that the best fit value for prediction length could be chosen.

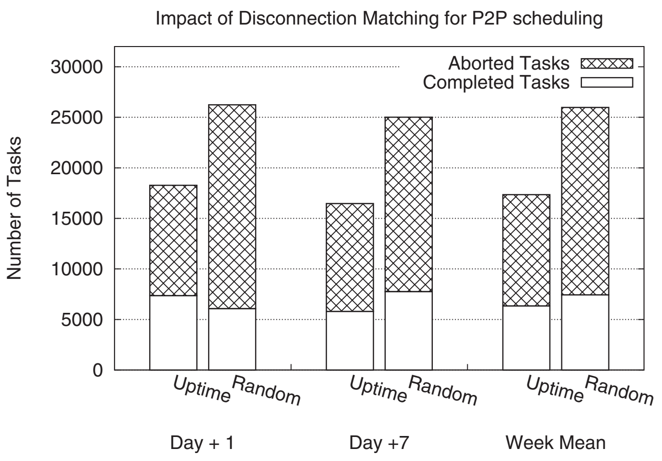

Fig. 9. A task is a set of three remote jobs of 4 hours started by every peer, ten minutes after coming online. A task is successful when the peer and its partners are still online after four hours to collect the results. Using availability predictions, a peer can decide not to start a task expected 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.

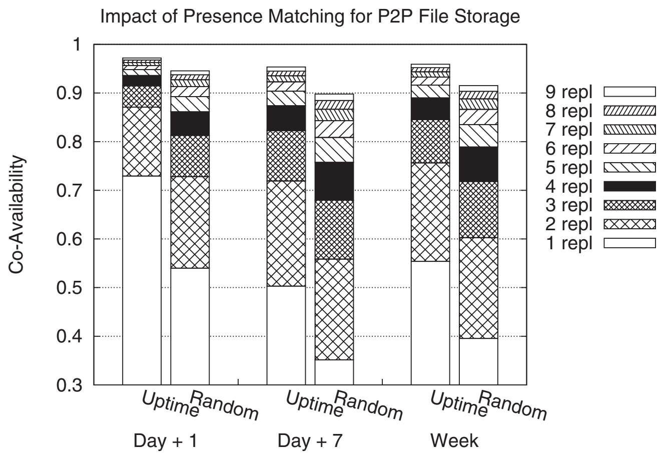

Fig. 10. Ten minutes after coming online for the first time, each peer creates a given number of replica for its data. Coavailability is defined by the simultaneous presence of the peer and at least one replica. Using presence matching, fewer replicas are needed to achieve better results than using a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

7.2. Results for Presence Matching

We compared presence matching with a random choice of replica locations for the P2P file-system application. The coavailability of the peer and at least one replica is plotted on Figure 10, 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 coavailability than 2 replicas with random choices; 5 replicas with presence matching give a coavailability of 95% which is only achieved using 9 replicas with random choices. As for the other application, week-old predictions performed still better than random choice in the same orders.

8. CONCLUSION

In this article, 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 [Morales and Gupta 2007; Le Fessant et al. 2008] and using these measures to predict future availability [Mickens and Noble 2006].

REFERENCES

Bhagwan, R., Savage, S., and Voelker, G. 2003. Understanding availability. In *Proceedings of the International Conference on Peer-to-Peer Systems (IPTPS'03).*

Bhagwan, R., Tati, K., Cheng, Y.-C., Savage, S., and Voelker, G. M. 2004. Total recall: System support for automated availability management. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'04).*

Busca, J.-M., Picconi, F., and Sens, P. 2005. Pastis: A highly-scalable multi-user peer-to-peer file system. In *Proceedings of the International European Conference on Parallel Processing (Euro-Par05).*

Chun, B.-G., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, F., Kubiatowicz, J., and Morris, R. 2006. Efficient replica maintenance for distributed storage systems. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'06).*

Cox, L. P., Murray, C. D., and Noble, B. D. 2002. Pastiche: Making backup cheap and easy. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI'02).*

Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. 2001. Wide-Area cooperative storage with cfs. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP'01).*

Drost, N., van Nieuwpoort, R. V., and Bal, H. E. 2006. Simple locality-aware co-allocation in peer-to-peer supercomputing. In *Proceedings of the Global and P2P Computing Workshop (GP2P'06).*

Duminuco, A. and Biersack, E. 2009. A practical study of regenerating codes for peer-to-peer backup systems. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS'09).*

Duminuco, A., Biersack, E. W., and En Najjary, T. 2007. Proactive replication in distributed storage systems using machine availability estimation. In *Proceedings of the ACM Conference on Emerging Network Experiment and Technology (CoNEXT'07).*

Godfrey, P. B.. Shenker, S., and Stoica, I. 2006. Minimizing churn in distributed systems. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communications.*

Guha, S., Daswani, N., and Jain, R. 2006. An experimental study of the Skype peer-to-peer voIP system. In *Proceedings of the International Conference on Peer-to-Peer Systems (IPTPS'06).*

Handurukande, S. B., Kermarrec, A.-M., Le Fessant, F., Massoulie, L., and Patarin, S. 2006. Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. In *Proceedings of the European Conference on Computer Systems (EuroSys'06).*

Jelasity, M. and Babaoglu, O. 2005. T-man: Gossip-Based overlay topology management. In *Proceedings of the International Workshop on Engineering Self-Organizing Systems (ESOA'05).*

Jelasity, M. and Kermarrec, A.-M. 2006. Ordered slicing of very large-scale overlay networks. In *Proceedings of the International Conference on Peer-to-Peer Computing (P2P'06).*

Jelasity, M., Montresor, A., and Babaoglu, O. 2009. T-man: Gossip-Based fast overlay topology construction. *Comput. Netw. 53*, 13, 2321–2339.

Kermarrec, A.-M., Le Merrer, E., Liu, Y., and Simon, G. 2009. Surfing peer-to-peer iptv: Distributed channel switching. In *Proceedings of the International European Conference on Parallel Processing (Euro-Par'09).*

Killijian, M.-O., Courtes, L., and Powell, D. 2006. A survey of cooperative backup mechanisms. Tech. rep. 06472, LAAS.

Kim, K. 2008. Lifetime-Aware replication for data durability in p2p storage network. *IEICE Trans. 91B*, 12, 4020–4023.

Kondo, D., Andrzejak, A., and Anderson, D. P. 2008. On correlated availability in Internet-distributed systems. In *Proceedings of the IEEE/ACM International Conference on Grid Computing (GRID'08).*

Le Blond, S., Le Fessant, F., and Le Merrer, E. 2009. Finding good partners in availability-aware p2p networks. In *Proceedings of the International Symposium on Stabilization, Security and Safety of Distributed Systems (SSS'09).*

Le Fessant, F., Sengul, C., and Kermarrec, A.-M. 2008. Pacemaker: Fighting selfishness in availability-aware large-scale networks. Tech. rep. RR-6594, INRIA.

Mickens, J. W. and Noble, B. D. 2006. Exploiting availability prediction in distributed systems. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'06).*

Morales, R. and Gupta, I. 2007. AVMON: Optimal and scalable discovery of consistent availability monitoring overlays for distributed systems. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS'07).*

Pamies-Juarez, L., Garcia-Lopez, P., and Sanchez-Artigas, M. 2008. Rewarding stability in peer-to-peer backup systems. In *Proceedings of the IEEE International Conference on Networks (ICON'08).*

Repository. 2010. Trace. http://socialstorage.gforge.inria.fr/.

Sacha, J., Dowling, J., Cunningham, R., and Meier, R. 2006. Discovery of stable peers in a self-organising peer-to-peer gradient topology. In *Proceedings of the International Conference on Distributed Applications and Interoperable Systems (DAIS'06).*

Saroiu, S., Gummadi, P. K., and Gribble, S. 2002. A measurement study of peer-to-peer file sharing systems. In *Proceedings of the Conference on Multimedia Computing and Networking (MMCN'02).*

Stutzbach, D. and Rejaie, R. 2006. Understanding churn in peer-to-peer networks. In *Proceedings of the Internet Measurement Conference (IMC'06).*

Voulgaris, S., Gavidia, D., and van Steen, M. 2005. CYCLON: Inexpensive membership management for unstructured p2p overlays. *J. Netw. Syst. Manag. 13*, 2.

Voulgaris, S., van Steen, M., and Iwanicki, K. 2007. Proactive gossip-based management of semantic overlay networks: Research articles. *Concurr. Comput. Pract. Exper. 19*, 17, 2299–2311.

Xin, Q., Schwarz, T., and Miller, E. L. 2004. Availability in global peer-to-peer storage systems. In *Proceedings of the International Workshop on Distributed Data and Structures (WDAS'04).*

Received February 2010; accepted August 2010