Heterogenous Dating Service with Application to Rumor Spreading

Olivier Beaumont
LaBRI – University of Bordeaux, INRIA Futurs
351 cours de la Libération,
FR-33405 Talence, France
obeaumon@labri.fr

Philippe Duchon
LaBRI – University of Bordeaux
351 cours de la Libération,
FR-33405 Talence, France
duchon@labri.fr

Miroslaw Korzeniowski*
Wroclaw University of Technology
ul. Wybrzeze Wyspianskiego 27,
PL-50370 Wroclaw, Poland
Miroslaw.Korzeniowski@pwr.wroc.pl

Abstract

Peer-to-Peer overlay networks have proven their efficiency for storing and retrieving data at large scale, but new services are required to take the actual performances of resources into account. In this paper, we describe a fully decentralized algorithm, called "dating service" meant to organize communications in a fully heterogeneous network, that ensures that communication capabilities of the nodes are not exceeded. We prove that with high probability, this service ensures that a constant fraction of all possible communications is organized. Interestingly enough, this property holds true even if a node is not able to choose another node uniformly at random. In particular, the dating service can be implemented over existing DHT-based systems. In order to illustrate the expressiveness and the usefulness of proposed service, we also present a possible practical application of the dating service. As an illustration, we propose an algorithm for rumor spreading that enables to broadcast a unit-size message to all the nodes of a P2P system in logarithmic number of steps with high probability.

*The author is partially supported by Emerging Technologies Programme of the EU under EU Contract 001907 DELIS "Dynamically Evolving, Large Scale Information Systems" and by MNiSW grant number PBZ/MNiSW/07/2006/46. The work was done when the author was in LaBRI and INRIA Futurs in Bordeaux.

1. Introduction and Related Work

Peer-to-Peer overlay networks have proven their efficiency for storing and retrieving data. Structured Peer-to-Peer networks based on Distributed Hash Tables (DHT for short) such as Pastry [9] or Chord [10], provide efficient mechanisms for routing in time logarithmic in the number of nodes in the system, but due to hashing, their use is limited to exact query searches. New services are therefore highly required in the context of large scale distributed networks. Moreover, heterogeneity in terms of communication resources is not taken into account and all resources play a symmetric role, whatever their capacity, whereas large scale platform exhibit a high level of heterogeneity, in terms of both processing and communication resources.

In this paper, we describe a distributed algorithm, called *dating service* which can be used e.g. to organize communication in a heterogeneous network, so that communication capabilities of nodes are not exceeded. The abstract purpose of our scheme is to randomly join demands and supplies of some resource of many nodes into couples. In a round it produces a matching between demands and supplies which is of linear expected size (compared to optimal one) and is chosen uniformly at random from all matchings of this size.

We illustrate the usefulness of this dating service by designing an application to rumor spreading in a heterogenous network, where demand and supply of a single node mean its incoming and outgoing bandwidth, respectively.

The only result similar to ours of which we are aware is on choosing a random peer in a distributed hash ta-

ble [6] by King and Saia. The authors give a method with which a peer can choose another one so that the distribution is uniform. The algorithm works in logarithmic time (expected and with high probability) and can also be run by many peers in a round yielding a linear number of connections distributed like balls thrown into n bins. Since in such models, when the number of balls is also n, some bins get $\Omega(\log n/\log\log n)$ balls, in some situations our scheme has an advantage here, as it respects incoming bandwidths.

The usual setting for rumor spreading (see for example [5]) is that a single node knows the rumor originally and wants to broadcast it to everyone else. The communication is organized in rounds and the algorithm only decides (usually in some random fashion) which nodes send messages to which nodes in each round. This allows for extensions such as rumors appearing in the network in course of time and also dynamics of the networks, including node failures. The rumor is assumed to have a unit size, i.e. it takes exactly one round to send it from a node to another node. In our scheme we use additional small bandwidth for maintenance, that is for the dating service.

Traditional algorithms, named PUSH, PULL and PUSH&PULL, (also described in [5]) work as follows. In each round each node chooses another node uniformly at random. In the PUSH model the first one sends a rumor to the second one. In the PULL model it is the other way around. In case of PUSH&PULL scheme the nodes exchange rumors. Main results of [5] are not only bounding time to $O(\log n)$ but also bounding total communication cost to $O(n \cdot \log\log n)$.

The results presented in this paper strongly differ from existing ones on rumor spreading. First, the algorithm we propose can be implemented over existing DHT systems such as Pastry [9] or Chord [10]. These systems provide efficient mechanism to deal with new and leaving nodes, even in the presence of high churn rate. This feature is necessary in the case of large scale platforms, where the set of participating nodes changes quickly over time. Thus, since dating service lies on top of a DHT, it can be implemented in practice. Moreover, it is known that DHT like systems are not perfectly balanced and some nodes are allocated more work than others. Nevertheless, we prove that the dating service is able to generate a random linear size matching between demands and supplies even if the DHT is not perfectly balanced.

The randomness property is essential when dealing with rumor spreading algorithm and all existing rumor spreading algorithms with logarithmic complexity rely on the possibility of choosing a random node in the network. However, the rumor spreading algorithm we propose ensures this property in practice, and is able to deal with heterogeneous networks, without violating incoming and outgoing communication capabilities, contrarily to what happens with PUSH, PULL and PUSH&PULL algorithms for rumor spreading. We prove that if the message starts from a node with sufficiently large outgoing bandwidth (at least $\Omega(\log n)$), nodes with at least average bandwidth $\Omega(m/n)$, where m is the sum of bandwidths of all nodes, will receive the message after only $O(\log n/\log(m/n))$ rounds with high probability, which opens the way for hierarchical content distribution, where nodes receive different messages according to their communication capabilities.

To the best of our knowledge, this paper represents the first attempt to prove theoretical results on an implementable scheme for rumor spreading on heterogeneous platforms.

The paper is organized as follows. In Section 2, we describe the dating service and prove that it organizes a linear number of dates between supplies and demands at each round. Then, we present in Section 3 an application of the dating service to rumor spreading and prove that our algorithms spread a unit-size message onto a fully heterogeneous platform in logarithmic time in the size of the network. The practical implementation of the dating service is analyzed in Section 4.1 and its results are assessed through a set of simulations in Section 4.2 and Section 4.3.

2. Dating service

2.1. Algorithm

The dating service is a tool which provides some service to some applications. We describe it as a routine running completely independently of any applications, which means that we have two separate networks, one for the dating service and one for an application and the latter can send requests to the former and receive answers.

The purpose of the dating service is to produce random matchings between two types of requests which, for convenience, we name *supplies* and *demands*. The routine works in rounds and in each round all peers of the application send information about their supplies and demands to the dating service, their requests are joined into couples and the peers can use these connections somehow and proceed to the next round.

In our description of the dating service, nodes taking part in providing the service are called *servers*, to distinguish them from their role as users. Note that this difference between servers and nodes only stands to ease the description of the dating service. In practice, all the

nodes will be used as servers and none of them are more important than others. In Section 4.1, we describe how one can implement such a dating service in a distributed hash table.

The centralized dating service based on a single server would work as follows. All nodes of an application would submit information about all their supplies and demands to the server, the server would choose uniformly at random a maximum matching and send information about connections to all nodes of the application. Producing a matching chosen uniformly at random is actually trivial: it is sufficient to permute uniformly at random both sequences of supplies and demands and match the i-th supply with the i-th demand for all i between 1 and $\min\{\#\text{supplies},\#\text{demands}\}$.

In the distributed setting, where the whole set of nodes is used to produce the random matching, the service works as follows. Each node sends each request, either demand or supply to a random server chosen according to a fixed probability distribution $P = (p_1, p_2, ...p_n)$. Then, the server is responsible to build a local random matching between the demands and supplies it has received. Note that the fact that the probability distribution of the servers is not a priori uniform (even if it is desirable for load balancing) has a strong influence on the effective possibility of implementing the dating service on a large scale platform. The actual implementation will be discussed in Section 4.1.

A pseudo-code of the dating service for a single round is shown in Algorithm 1.

Algorithm 1 Dating service(probability distribution $P = (p_1, ...p_n)$)

for each node i in parallel **do**
  for each unit of supply or demand at i **do**
    send a request to a random server chosen according to P
for each server j in parallel **do**
  $r \leftarrow$ number of demand requests at j
  $s \leftarrow$ number of supply requests at j
  $q \leftarrow \min\{r, s\}$
  choose uniformly at random without replacement q requests of each type
  generate a uniformly chosen random perfect matching of the chosen requests
  for each of the $r + s$ received requests **do**
    if the request is among the $2q$ chosen requests **then**
      send information about the partner to the creator
    else
      send to the creator information about no date

Note that in the above description, it is not necessary that the random choice of servers be uniform – only that all requests are sent using the same distribution P. This randomness is a load-balancing factor; as an extreme case, sending all requests to a single server would result in a centralized scheme.

2.2. Notations

Throughout this paper we use the classical notations of $O()$ and $\Omega()$, where $O(f(n))$ means at most $c \cdot f(n)$ for constant $c > 0$ and sufficiently large n and $\Omega(f(n))$ means at least $c \cdot f(n)$ for constant $c > 0$ and sufficiently large n.

Together with the above notations we also use the term *with high probability* (in short: whp) in the following way: a random variable $X(n)$ is bounded by $O(f(n))$ with high probability means that for any constant ℓ, $\Pr[X > c \cdot f(n)] \leq \frac{1}{n^\ell}$, where the constant c depends only on ℓ. The term for $\Omega()$ notation is defined similarly.

2.3. Theoretical Analysis

Denote the total number of demands by m and the total supplies by m'. For the analysis, we assume that $m \leq m'$ (otherwise, just switch the symmetric roles of supply and demand); thus, m is the maximum number of dates that a centralized service would be able to organize in a round. In [1] we have shown that if $m = \Omega(n)$ then the number of dates organized by the dating service is $\Omega(m)$ on expectation and with high probability, i.e. it is by at most a constant factor worse than what a centralized dating service would produce.

Our first lemma in the analysis of the dating service is intuitively obvious from the construction; its proof can be found in [1].

Lemma 1. *Conditioned on the total number of dates in a round of the dating service being k, the set of matches produced is a uniform random k-matching of the sets of demands and supplies.*

The second lemma states that, provided the number of servers is not too large compared to the supply or demand (whichever is larger), the dating service matches, on average, a constant fraction of the rarer type of requests with the other type.

Lemma 2. *Let $X = X(m, m', n, P)$ denote the number of dates organized by the dating service in a single round. Assume that $m' \geq m$ and $m' \geq cn$ for some positive constant c. Then there exists a constant $\beta = \beta(c) > 0$ such that, for any P,*

$$\mathrm{E}(X) \geq \beta m. \tag{1}$$

Proof. Let $n' = \max(m', n)$, and $c = m'/n$. Sending a request to a random server chosen according to P is equivalent to having a fixed partition of the interval $(0, 1)$ into n intervals of lengths $p_1, \ldots, p_n$, and sending each request to the server corresponding to the interval containing a random variable which is uniformly distributed in $(0, 1)$. Now split each such interval into a number (possibly zero) of "good" intervals of length exactly $\frac{1}{4n'}$ and one "bad" shorter interval; the total length of "bad" intervals is thus less than $\frac{n}{4n'} \leq \frac{1}{4}$, so that the total number of "good" intervals is at least $3n'$. X is at least the number of dates one obtains by considering only those produced by requests sent to these $3n'$ "good" intervals and ignoring any request sent to a "bad" interval.

The expected number of "good" intervals which receive at least one demand request is at least $5m'/8$ (consider the m' requests as being sent sequentially: when sending the k-th request, there are at least $3n' - k + 1$ empty intervals, for a probability at least $\frac{3n'-k}{4m'} \geq \frac{3}{4} - \frac{k}{4m'}$ of hitting one of them; summing these probabilities and taking the worst case $m' = n'$, we obtain the $5m'/8$ bound). As a consequence, with probability at least $1/4$, at least $m'/2$ of these intervals get at least one request. Assuming this is the case, consider the m supply requests as being sent sequentially: the k-th request has probability at least $\frac{m'/2-k+1}{4n'}$ of hitting a "good" interval which received a demand request but has not yet received a supply request, thus generating a date. Summing over the first $\frac{m}{2} \leq \frac{m'}{2}$ such requests, the (conditional) expected number of dates is at least $\frac{m.c}{16}\left(1 - \frac{m}{2m'}\right) \geq m.\frac{c}{32}$, which proves (1) for $\beta = \frac{c}{64}$. (We make no effort to optimize the constant β, but simulations and calculations based on unproved Poisson approximations indicate that the result remains true with much larger values of β; for instance, $\beta(1) = 0.46$ seems to work.) □

In [1], we prove that the result of Lemma 2 also holds with high probability, provided m is also $\Omega(n)$.

Together, Lemmas 1 and 2 yield the following:

Lemma 3. *Assume $m \leq m' = \Omega(n)$. Then, in any round of the dating service, each unit of supply has constant probability of being matched to a unit of demand.*

Even though the events of different demands having dates in a single round are dependent, the events of a single demand getting a date over different, independent rounds are not. Thus, the following corollary easily follows.

Corollary 4. *If the assumptions of Lemma 3 hold for each round, each single unit of supply gets matched with a demand within $O(\log n)$ rounds with high probability.*

3. Rumor spreading

In this section we describe and analyze an application of the dating service to rumor spreading. The basic problem is as follows. In the beginning there is a single peer which knows some information — the rumor. We are looking for a scheme which spreads the rumor in the whole network as fast as possible. PUSH and PULL algorithms mentioned in Section 1 do it in $O(\log n)$ rounds. We describe a scheme based on the dating service which achieves the same bound, but makes efficient use of heterogeneous bandwidths by speeding up the process for average (or higher) bandwidth peers.

Incoming and outgoing bandwidths of peers are limited: to each peer i is associated an incoming bandwidth $b^{in}(i)$ and an outgoing bandwidth $b^{out}(i)$, which means that peer i is able to receive $b^{in}(i)$ unit size messages and to send (simultaneously) $b^{out}(i)$ unit size messages during each round. For easier notation we define the total incoming and outgoing bandwidths as $B^{in} = \sum_{i=1}^{n} b^{in}(i)$ and $B^{out} = \sum_{i=1}^{n} b^{out}(i)$. The smaller of the two we denote as m, i.e. $m = \min\{B^{in}, B^{out}\}$. Messages involved in the organization and realization of the dating service itself are assumed to be much shorter than this unit size, and are not so restricted. In order to consider peers with very different capabilities, the ratios $\frac{\max_i b^{in}(i)}{\min_i b^{in}(i)}$ and $\frac{\max_i b^{out}(i)}{\min_i b^{out}(i)}$ are not a priori bounded. On the other hand, we assume that for a given peer, the ratio between its incoming and outgoing bandwidths is bounded by a constant C:

$$\forall i, \quad \frac{1}{C} \leq \frac{b^{in}(i)}{b^{out}(i)} \leq C.$$

In order to organize communications, we rely on the dating service. The application is quite straightforward: after the dating service produces a set of pairs (incoming,outgoing link), nodes connected in such pairs communicate. This means that the first node of each pair sends a rumor to the second one, provided that it has anything to send. We prove below that using such a scheme, all nodes will know the rumor after at most a logarithmic (in n) number of rounds.

Theorem 5. *With high probability, the rumor spreading process based on the dating service sends the rumor to all nodes in $O(\log n)$ rounds.*

Proof. We split the analysis into three phases, depending on the total outgoing bandwidths of currently informed peers; we denote this bandwidth in round t by I_t:

1. from $I_0 \geq 1$ until $I_t = \Omega(\max(\frac{m}{n}, \log n))$;
2. from the end of phase 1 until $I_t \geq m/2$;

3. from the end of phase 2 until all peers are informed.

In the following we show that each phase separately takes $O(\log n)$ rounds. For the first two phases we will need the following definition and lemma:

Definition 6. *Fix a constant $\alpha < 1$. Take all I_t links outgoing from informed peers and consider them in any fixed order. We say that a link is successful in round t, if (i) one of its outgoing requests is matched with an incoming request from a peer not previously contacted (not even by any informed link already considered in round t) and (ii) the outgoing bandwidth of the contacted peer is at least $\alpha \cdot \frac{m}{n}$*

For such a definition of success we now prove a lower bound on the probability that a given link is successful in a given round during phases 1 and 2.

Lemma 7. *Fix $\alpha = 1/4$ in the previous definition and let β denote the fraction of dates organized by the dating service. Then, in every round t, under the condition that $I_{t+1} \leq \frac{m}{2}$, an outgoing link is successful with probability at least $\frac{\beta}{4C^2}$.*

Proof. In the reasoning below we condition on the number of dates in the current round being at least $\beta \cdot m$, which happens with high probability.

We consider two cases:

1. $m = B^{out} \leq B^{in}$
2. $m = B^{in} \leq B^{out}$

In the first case, the probability for an outgoing link of getting a date is at least β. Consider all uninformed peers which have outgoing bandwidth at most $\alpha \cdot \frac{m}{n}$. Their total bandwidth is at most $\alpha \cdot m$, so the total uninformed outgoing bandwidth we would like to hit is at least $(1-\frac{1}{2}-\alpha)\cdot m = \frac{m}{4}$. The total incoming bandwidth of these peers is at least $\frac{m}{4C}$, whereas the total incoming bandwidth of all peers is at most $C \cdot m$, so based on Lemma 1, the probability of hitting one of these "interesting" peers is at least $\frac{1}{4C^2}$. The joint probability of getting a date and hitting a correct peer is thus at least $\frac{\beta}{4C^2}$.

In the second case the reasoning is similar, except that an outgoing link has a probability of getting a date in round t at least $\frac{\beta}{C}$ and that the total incoming bandwidth is at most m. Thus, the resulting probability of success is also at least $\frac{\beta}{4C^2}$. □

Now we are able to bound the length of the first phase.

Lemma 8. *The first phase ends in at most $O(\log n)$ time steps, with high probability.*

Proof. We know that there is at least one outgoing link from a peer which initially has the message. Each round of phase 1, this same link has a lower bounded probability of success; since rounds are independent, the number of rounds until this link gets at least $d \cdot \log n$ successes is at most $d' \cdot \log n$, with high probability. □

For the second phase we show that in each round we multiply by a constant factor (strictly larger than 1) the number of informed outgoing links, with high probability. Since we start with at least $\frac{m}{n}$ such links, $O(\log_{(1+m/n)} n)$ rounds would then suffice to come to $\frac{m}{2}$ informed links. The precise statement is formulated as follows:

Lemma 9. *In each round of phase 2, with high probability, a constant fraction of connections succeeds. Phase 2 lasts $O(\log n/\log(1+\frac{m}{n}))$ rounds.*

Proof. Let $I_t \geq \Omega(\log n)$ be the number of informed outgoing links in round t. Each of them succeeds with probability $p_t = \Omega(1)$ (though not independently), which exactly means that, on expectation, a constant fraction of them are successful. In order to prove that this constant fraction holds with high probability, we use the Independent Bounded Differences Inequality as described in [7, Theorem 3.1].

We start by conditioning on the total number of dates in the current round being k (this k is at least $\beta \cdot m$ with high probability). Then the set of matched outgoing links from informed peers is independent of the set of matched incoming links to uninformed peers, and the cardinality of the first set follows the hypergeometric distribution with parameters (k, B^{out}, I_t).

Dates can be faithfully simulated using a relatively low number of independent random variables, as follows: number all outgoing links 1 to B^{out} in an arbitrary order with the only condition that those from informed peers are numbered 1 to I_t, and take k independent uniform random permutations $\sigma_1, \ldots, \sigma_k$ of $[[1, B^{out}]]$. Similarly, number all incoming links 1 to B^{in} in an arbitrary order, and take k independent uniform random permutations σ'_1, σ'_k of $[[1, B^{in}]]$.

Now the outgoing (resp. incoming) end of each date can be chosen as follows: the i-th matched link is the first element in permutation σ_i (resp. σ'_i) that is not among the $i-1$ first matched links.

The total number of successes in the round thus appears as a function of $2k$ independent random variables, and changing just one of these random variables can only result in changing at most one of the matched links (though it can result in changing an arbitrary number of matches, at most one unmatched link can become matched); as a consequence, changing one of the permutations can change the total number of successes

by at most 1. Thus, we can apply the Independent Bounded Differences Inequality with constant $c = 1$, which proves that, if $\mu = \mu(I_t, k)$ is the expected number of successes, S is the number of successes, and X is the number of dates, we get

$$\Pr(|S - \mu| \geq t | X = k) \leq 2e^{-t^2/k}$$

Taking $t = \sqrt{a \cdot k \cdot \log n}$ above yields an upper bound of $O(n^{-a})$. Since $k = \Omega(\log n)$ and we can choose the constant to be as large as we need it (by changing the constant in the $\Omega()$ in the definition of phase 1), this proves that, with high probability, the deviation of number of successes from its expectation is less than a constant fraction of its expectation.

The claim on the duration of phase 2 follows: with high probability, we have $\Omega(I_t)$ successes in each round t of phase 2, which means that $I_{t+1} \geq (1 + \gamma \frac{m}{n}) \cdot I_t$ (where $\gamma > 0$ is the constant – which we make no attempt to make explicit or optimize – in the "constant proportion of successes" claim) with high probability. Since phase 2 ends when I_t has been multiplied by a factor of

$$\frac{m/2}{\max\left(\frac{m}{n}, \log n\right)} = \min\left(\frac{n}{2}, \frac{m}{2 \log n}\right),$$

we get a high probability upper bound on the duration of phase 2 of

$$O\left(\frac{\log\left(\min(n, m/n)\right)}{\log(1 + m/n)}\right).$$

□

Finally, we bound the length of the last phase.

Lemma 10. *With high probability, it takes at most $O(\log n)$ rounds to send the rumor to all the uninformed peers, if we start with $\Omega(m)$ outgoing connections belonging to informed peers.*

Proof. Consider a single uninformed peer, it has at least one incoming link. Reversing the roles of incoming and outgoing links from the proof of Lemma 8, we see that, each round, each such incoming link has constant probability of having a date with an outgoing link from an informed peer; thus, with high probability, $d \cdot \log n$ rounds suffice for such a date, for an appropriate constant d. Taking a union bound on the probabilities for each uninformed peer to remain uninformed, we see that phase 3 ends in $O(\log n)$ rounds with high probability. □

Overall, Lemmas 8, 9, and 10, yield Theorem 5 □

We now come to the real strength of the dating service concerning heterogenous networks. Assume that the network is heterogenous and such that $m > n \log n$, i.e. the average bandwidth is larger than $\log n$ even though there may still be some weak peers (with low bandwidth) in the network. If the rumor starts at a peer with at least average bandwidth, that is at least $\Omega(m/n)$, then all other peers of average bandwidth receive the rumor after $O\left(\frac{\log n}{\log(1+m/n)}\right)$, with high probability. This means time $O\left(\frac{\log n}{\log \log n}\right)$ for average bandwidth $\Omega(\log n)$ and constant time for average bandwidth $\Omega(\sqrt{n})$.

Theorem 11. *Assume $m = \Omega(n \log n)$. If the peer which has the rumor initially has bandwidth at least $\Omega(m/n)$, then all peers with bandwidth $\Omega(m/n)$ receive the rumor within $O\left(\frac{\log n}{\log(m/n)}\right)$ rounds, with high probability.*

Proof. Notice that Phase 1 is finished before any communication begins. As shown in Lemma 9, phase 2 lasts $O\left(\frac{\log n}{\log(1+m/n)}\right)$ rounds. Thus, we only need to show that after Phase 2 has finished, all average peers receive the rumor within $O\left(\frac{\log n}{\log(1+m/n)}\right)$ rounds.

Consider a single uninformed peer with incoming degree $\Omega(\log n)$. Basing on the proof of Lemma 9 we know that a constant fraction of its requests for receiving are fulfilled with high probability. Each of the incoming messages contains the rumor with probability at least $1/2$, so in a single round such a peer does not get the rumor with probability inversely polynomial in n. After constant number of rounds such a peer is informed with high probability. □

Even if the rumor starts in a weak peer, we still have some chance:

Corollary 12. *Assume again that $m = \Omega(n \log n)$. Independently of the bandwidth of the peer on which the rumor starts, it takes on expectation $O\left(\frac{\log n}{\log(m/n)}\right)$ rounds to deliver the rumor to all peers with at least average bandwidths, provided that the average bandwidth is $m/n \geq \Omega(\log n)$.*

Proof. Just notice that the single peer which knows the rumor initially in each round has constant probability of sending the message to an average peer. This takes constant number of rounds on expectation. Afterwards, with high probability all the average peers are informed in $O\left(\frac{\log n}{\log(m/n)}\right)$ rounds by Theorem 11 □

4. Practical considerations and simulation results

4.1. Dating Service in Peer-to-Peer Networks

In this subsection we describe how one can implement the dating service scheme in Peer-to-Peer networks. The idea is to use a distributed hash table (DHT; see for example [10, 8, 3, 9]). These schemes connect peers into networks and provide good (poly-logarithmic in the size of the network) routing strategies and self-adapt to the dynamics of the network (joining and leaving nodes). Building new services on top of DHTs is therefore highly recommendable.

The most important functionality for the dating service is the way that data is stored in such a network. In all designs, there is an underlying virtual space (most often a $(0, 1]$ ring) which is partitioned among the peers. When a data item is inserted into a point in this virtual space, it is assigned to the peer responsible for this point.

One can easily adapt this to implement the dating service as follows. Whenever a request must be sent to a P-random server, the sending peer simply picks a random key in the virtual space according to some common distribution (in the case of a $(0, 1]$ ring this is simply a uniform random variable on $(0, 1]$), and uses the DHT scheme to route the request to the responsible peer, which acts as a dating service server; thus, the distribution of peer responsibilities in the virtual key space provides the common P distribution.

This distribution will typically not be uniform, since some nodes will be responsible for larger intervals than others. It is worth noting that all the proofs proposed in previous sections do not assume a perfect load balance between nodes. Indeed, the property stating that the dating service builds a random uniformly chosen random matching between supplies and demands is still valid, even if the DHT is badly balanced. This property strongly differs from what is needed in the case of algorithms based on the possible random uniform choice of a distant node. In this case, sophisticated and costly techniques must be added to the DHT in order to choose random distant nodes uniformly [6]. In Section 4.3, we compare our rumor spreading scheme to PUSH and PULL schemes that require the ability of choosing a random distant node.

The only consequence, in the scheme we propose, of a load imbalance in the DHT is that some nodes will be responsible for organizing more dates than others. Of course the dating service routine needs only small messages but it is still better to spread the work among many peers than to send everything to a single server. Even if one organises a DHT in the simplest way where nodes choose places on a $(0, 1]$ ring uniformly at random, the largest interval assigned to a peer will be of length $O(\log(n)/n)$ with high probability. This means that the expected fraction of requests received by a single peer (the maximum load of a peer) will exceed the average by an at most logarithmic factor. The dating service benefits also from such properties of DHTs as low congestion and routing time when routing even a linear number of messages to random places. Since the construction of the random matching is fully distributed among nodes, the cost at each node is very low. This is because it corresponds to building a matching between supplies and demands received at this node, and their expected number is 1. Therefore, a load imbalance in the DHT does not affect at all the randomness of the constructed matching and it only slightly affects the processing cost of the distributed computation of the matching.

To ensure that the dating service performs efficiently, the number of at least one type of requests (supply or demand) must be at least linear in the number of servers. To guarantee this condition, only peers with at least one request should act as servers in the dating service; depending on the application, this may mean peers have to maintain more than one DHT structure (possibly with the same virtual coordinates), at least one of which will not include all the peers. With this precaution, we ensure that $m \geq n/2$ or $m' \geq n/2$, which is what is needed in our analysis.

In order to validate the usefulness and the efficiency of the proposed dating service, we performed a set of simulations. In the simulations dedicated to the dating service itself (Section 4.2), in order to obtain a practical implementation on a large scale platform, we rely on Distributed Hash Tables (see for example [8, 10]). Routing in DHTs takes time $\Theta(\log n)$ or close and since we use it in each round, it would mean that each round takes such time. One can use pipelining of dates, that is send requests for dates in each round even before receiving the answers for the previous one. Thus, after $\Theta(\log n)$ time steps, answers will start coming each round. This means that for k rounds of dating service we need time $\Theta(\log n + k)$. On the other hand, for the analysis of rumor spreading (Section 4.3), we use a setting where a distant node can be chosen uniformly at random. As simulation results for the dating service (Section 4.2) prove, this disadvantages our rumor spreading algorithm, since the expected number of dates is larger when using DHT-based systems. Nevertheless, all other rumor spreading schemes we consider (i.e. PUSH, PULL and PUSH and PULL) rely on the possibility of choosing a uniform random nodes.

4.2. Simulations for dating service

In the simulations meaning to check the actual number of dates in a round there were n nodes (for $n \in \{10, 10^2, 10^3, 10^4, 10^5\}$) and they generated n requests of each type. We performed two sets of simulations. In the first one, denoted by uniform in what follows, we assume each node has the ability of choosing uniformly at random distant nodes to send its demands and supplies. As already noted in Section 4.1, the dating service can also be efficiently implemented over a DHT. Corresponding results are denoted by DHT in what follows.

For uniform distribution of requests ($p_1 = ... = p_n$) we ran 10^4 or 10^3 rounds to calculate the average and standard deviation. The experiments were performed on a single computer — therefore for $n \in \{10^4, 10^5\}$) we were able to run only 10^3 rounds. The average number of arranged dates appears to be always slightly more than $0.47 \cdot n$.

To check how our process behaves in a DHT setting, we generated a DHT and then ran 10^4 or 10^3 rounds. Again we calculated averages and standard deviations. For the final result we took only one DHT out of 200 generated — the one that showed *the worst* average. The number of dates for these worst found DHTs are above $0.52 \cdot n$. For the best ones they reach as high as $0.67 \cdot n$ for $n = 10$ but are close to $0.55 \cdot n$ for the largest $n = 10^4$. In this case we were not able to run the experiments for $n = 10^5$. The results are summarized in Figure 1.

It is worth noting that the dating service performs better (i.e. organizes more dates) when using DHT. This property is not surprising. Indeed, in the DHT setting, load balancing is worse than in the uniform setting. Even if balancing the load is highly desirable, having nodes responsible for organizing more dates makes the system more efficient. An extreme case of load imbalance is the centralized case, where all nodes send their demands and supplies to the same node. In this case, this node would be overloaded since it would be responsible for finding a random matching of size n and would receive $2n$ messages, but the number of organized dates would be optimal (i.e. n).

4.3. Simulations for rumor spreading

In order to check how our dating service behaves in rumor spreading in comparison to other schemes, we performed another set of simulations.

In order to compare the scheme we propose to existing schemes, such as PUSH, PULL and PUSH and PULL algorithms (see [5] for a survey), we only considered the fully homogeneous case, where each node sends exactly one request of each type (supply and demand). This corresponds to a fully homogeneous platform, where each node is able to send and to receive exactly one unit size message at each round. Although the rumor spreading algorithm can be used in a much more general setting, where each node is associated to incoming and outgoing bandwidths, we restrict our set of simulations to the homogeneous case, for the sake of comparison with existing schemes. Similarly, for all simulations, we assume that each node is able to choose a distant node uniformly at random, what corresponds to the uniform setting for dating service simulation presented above. Again, this assumption is necessary for the sake of comparison since all existing algorithms rely of this ability, what requires complex schemes if implemented over a DHT [6]. The extra cost needed to perform this random choice is not taken into account, whereas the scheme we propose can be efficiently implemented using a DHT. It is also worth noting that, as mentioned in Section 4.2, the dating service is even more efficient when based on a DHT rather than on random choices.

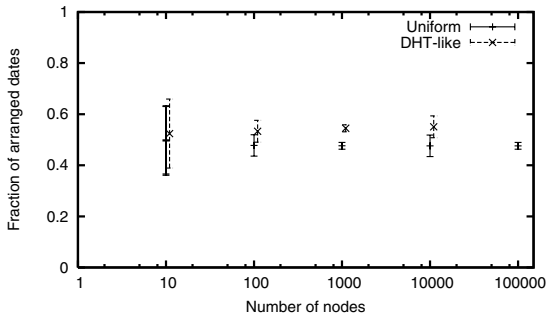

Figure 1. Simulation results for dating service for both DHT and Uniform settings

In the following simulations, the number of nodes ranges from 10 to $100,000$ and we compare the following algorithms: simple PUSH, simple PULL, simple PUSH and PULL, fair PULL and PUSH and fair PULL. In the fair PULL version, a node that has been chosen as source by several distant nodes only satisfies one request at each round, so that communications capabilities of the sending node are not exceeded. We compared them with rumor spreading based on dating service using uniform organization of dates. Again, in order to calculate averages and standard deviations, we performed each experiment 10^4 or 10^3 times.

The averages together with standard deviations are shown in Figure 2. We can see that all the algorithms, including the one based on the dating service, exhibit a logarithmic number of rounds. The order of the algorithms from the best to the worst is as follows: PUSH and PULL, PUSH and fair PULL, PULL, fair

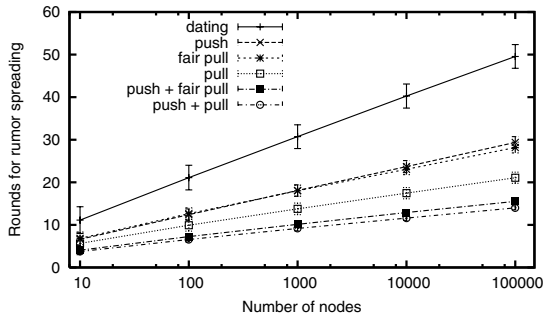

Figure 2. Simulation results for rumor spreading

PULL, PUSH, dating service. The following comment is needed here: notice that PUSH and PULL methods benefit from double communication in each round - one for PUSH and one for PULL. Also the unfair versions of the algorithms may benefit from much higher bandwidth, when many nodes PULL an informed node at the same time. Therefore, we should actually compare the rumor spreading based on the dating service only with the PUSH and fair PULL methods. It roughly 50% slower than them.

Nevertheless, recall that the algorithm we propose can be efficiently implemented in practice using a DHT whereas the practical implementation of all other schemes require the ability of choosing a distant node uniformly at random, and the extra cost induced by this operation [6] is not taken into account in these simulation results. Moreover, the rumor spreading algorithm we propose has been designed for heterogeneous platforms, where each participating node is associated to an incoming and an outgoing bandwidth. The simulation results presented here are for the fully homogeneous case (and even in this case, only the PUSH and the fair PULL versions satisfy this constraint in practice) for the sake of fair comparison only.

We therefore think that it is a fair price to pay for possibility of distributed implementation and for the extension to the much more realistic heterogenous case for large scale platforms.

5. Conclusion and Future Work

In this paper, we described a fully decentralized algorithm, called "dating service" which can be used to organize communications in a fully heterogeneous network, in a way that ensures that communication capabilities of the peers are not exceeded. We proved that with high probability, this service ensures that a constant fraction of all possible communications is organized even if it is impossible to choose nodes uniformly at random. As an illustration, we presented a practical design of the dating service based on currently available DHT systems. We also showed an application of the dating service in rumor spreading. We proved that it behaves well and copes well with heterogenous networks. It is worth stressing that in rumor spreading in a heterogenous network peers with higher bandwidths get the rumor faster whereas the weaker peers are still not forgotten.

We strongly believe that the dating service can be used as a building block for many distributed services where nodes offer and request for resources. The first possible extension consists in considering the case of rumor mongering (or equivalently the broadcast of a large message). In this context, the message is split into smaller parts and is sent in a pipelined fashion through the network. In this case, we can make a deeper use of the dating service mechanism, since both incoming and outgoing bandwidths can be used efficiently. The most challenging problem consists in organizing the communications, so as to ensure that each part of the message is received exactly once. To achieve this goal, randomized network coding techniques [4] have proven their efficiency [2]. The dating service may also be used in distributed replicated storage systems. In this context, each node offers room (in terms of block) to store remote objects and requests room to store remotely its local objects. In this case, the dating service may be used to organize block exchanges between nodes.

As already stressed, the first main advantages of the dating service scheme is that it can be implemented over existing DHT-based systems, without requiring the ability to perform sophisticated requests such as the choice of a uniform random node. The second main advantage is that it can be used in non uniform settings, where the number of demands and supplies differ at all nodes. This corresponds, in the case of rumor spreading, to fully heterogeneous platforms, where each node is associated to an incoming and outgoing bandwidth. To the best of our knowledge, this paper represents the first attempt to prove theoretical results on an implementable scheme for rumor spreading on heterogeneous platforms.

References

[1] O. Beaumont, P. Duchon, and M. Korzeniowski. Heterogenous dating service with application to rumor spreading. Research Report RR-6168, INRIA, Apr. 2007. Available at the url http://www.labri.fr/perso/obeaumon/publis/datingservice.pdf.

[2] S. Deb, M. Médard, and C. Choute. Algebraic gossip: A network coding approach to optimal multiple ru-

mor mongering. *IEEE ACM Transaction on networking*, pages 2486 – 2507, 2006.

[3] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao. Distributed Object Location in a Dynamic Network. In *Proc. of the 14th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 41–52, 2002.

[4] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger. On randomized network coding. In T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, editors, *Proceedings of 41st Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[5] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *IEEE Symposium on Foundations of Computer Science*, pages 565–574, 2000.

[6] V. King and J. Saia. Choosing a random peer. In *Proc. of the 23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 125–130, 2004.

[7] C. McDiarmid. *Concentration*, volume 16 of *Algorithms and Combinatorics*, pages 195–248. Springer, 1998.

[8] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A Scalable Content Addressable Network. In *Proc. of the ACM SIGCOMM*, pages 161–172, 2001.

[9] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[10] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. of the ACM SIGCOMM*, pages 149–160, 2001.