J Grid Computing (2007) 5:319–335
DOI 10.1007/s10723-006-9057-1

Fault-tolerant Resource Discovery in Peer-to-peer Grids

Peter Merz · Katja Gorunova

Received: 31 August 2005 / Accepted: 4 November 2006 / Published online: 22 December 2006

Abstract Peer-to-peer overlay technologies offer several advantages over centralized solutions to managing desktop Grids. We present a new hybrid approach to resource discovery in P2P Grids, i.e. desktop Grids based on peer-to-peer overlays. This approach combines the advantages of information propagation based on spanning trees in chord-like structured overlays and epidemic algorithms. We provide a mathematical model for predicting the process of information dissemination and verify its prediction quality in various evaluations. Moreover, we show the failure resistance of the proposed approach in several scenarios. In particular, we demonstrate the efficiency of our approach even in scenarios where 50% of the peers in the overlay fail in short time.

Key words peer-to-peer · desktop Grids · resource discovery · job distribution

P. Merz (✉) · K. Gorunova
Department of Computer Science,
University of Technology Kaiserslautern,
Kaiserslautern, Germany
e-mail: pmerz@informatik.uni-kl.de

K. Gorunova
e-mail: gorunova@informatik.uni-kl.de

1 Introduction

To utilize the computational power of modern desktop PCs, desktop Grids have been proposed for automated deployment of computational tasks to idle machines in computer networks. To avoid a central management of these systems and to enable scalability on wide area networks, it is promising to use peer-to-peer (P2P) overlays as a basis for desktop Grids.

These P2P Grids can be designed to allow all peers to submit jobs to the system. Hence they require efficient mechanisms for job distribution. Once a job is submitted to the system, the goal is to find a number of m peers willing to participate in the computation, for example if the job is comprised of m independent tasks. This kind of resource discovery in P2P overlays is highly decentralized since the decision of a peer to join the computation is influenced by local properties such as the current load on the peer and the users preferences.

In this paper, we focus on resource discovery mechanisms for job submission/distribution in P2P overlays. Unlike previously proposed approaches, we concentrate on highly fault-tolerant algorithms by considering scenarios in which a high proportion of the peers can fail at any time. Therefore, we propose a hybrid combining efficient chord-like spanning tree algorithms and

robust epidemic algorithms for job information propagation. We show that the resulting hybrid is both highly fault-tolerant and efficient in terms of worst-case time to propagate job information.

The paper is organized as follows. In Section 2, an overview of existing approaches utilizing peer-to-peer technology for Grid computing is provided. A new approach for resource discovery in peer-to-peer Grids is presented in Section 3. In Section 4, the results for several evaluations showing the feasibility of the proposed approach are discussed. The paper is concluded and an outline of future research is provided in Section 5.

2 Peer-to-peer Grids

2.1 Desktop Grids

Unlike other Grid computing technologies, desktop Grids focus on utilization of CPU power of ordinary desktop computers. Several approaches have been proposed for utilizing unused power of idle PC's one of which is the famous SETI@Home project [2]. There are other, similar projects such as Entropia [6, 13], GIMPS [14] or distributed.net [8]. They have in common that they are centralized and allow clients to run dedicated computational tasks. There is no (at least easy) way for a client to submit its own jobs to the system as common in Grid computing. Both BOINC [1] and XTREMWEB [12] allow users to submit jobs or set up their own application for distributed computing but utilize a centralized network architecture.

2.2 Peer-to-peer Overlay Networks

Peer-to-peer (P2P) systems on the other hand are self-organizing overlay networks in which desktop computers participate to share resources with each other. These systems were originally designed for file-sharing but can also be used for CPU resource sharing. P2P overlays can be either unstructured networks like Gnutella [15], or structured networks, based e. g. on rings [30], or hypercubes [29]. Recently, distributed hash tables (DHTs) based on structured overlays like Chord [30], CAN [27] or Pastry [28] have been shown to be effective for looking up resources in P2P networks.

Epidemic algorithms [7, 11] are based on unstructured overlays and are known to be easy to implement, fault-tolerant and scalable. The outline of an epidemic algorithm is provided in Fig. 1.

In each time step, a peer selects a random peer from its host list (the list of his currently known peers in the overlay) and updates its state (s) with the state of the selected peer. Analogously, the selected pair updates his state after the communication. With this mechanism information can spread through the overlay. Note that the peers' host lists are part of its state and hence may be updated after each communication.

2.3 Towards P2P Grids

In desktop Grids based on P2P technology – denoted P2P Grids in the remainder of this article – each peer should be able to participate in computations as well as submit his own jobs

Fig. 1 The epidemic algorithm

```
do forever                          do forever
  wait(I_com time units);             (r, s_r, m) := receive();
  r := randomPeer();                  if (m = "request")
  send(r, s, "request");                send(r, s, "reply");
                                      s := update(s, s_r, r);
```

(a) active thread (b) passive thread

to the system. In order to achieve this, different approaches are feasible:

(a) Each idle peer searches for available jobs, or
(b) Each peer submitting a job to the system searches for peers willing to participate in the computation.

If we assume that each peers' decision whether or not to participate in a computation is based on the job requirements (Runtime system, operation system, memory requirements, ...), trust or accounting issues, as well as user preferences and the current load, each peer has to decide on its own. The decision process is fully decentralized. Hence, either (a) an idle peer queries for a job until he has found an acceptable one or (b) the job submitting peer queries for idle peers. In the worst case if all peers reject the job or there is no job announcement, both queries result in a multicast to all peers in the overlay.

For structured P2P architectures like Chord [30], CAN [27] or Pastry [28], special multicast communication methods were proposed that make use of regular network architecture. For example, the multicast algorithm for Chord introduced in [9] has minimal message complexity and spreads very fast over the network. The partitioning of the network address space into equal-sized slices for multicast distribution was considered in [16]. In the remainder of this paper, we will refer to these methods as 'regular' multicast distribution. The systematic comparison of regular routing algorithms and flooding for multicast communication in [4] shows that regular methods consistently outperform the flooding approach. However, continuous fluctuations of real P2P Grids (joins, leaves and failures of nodes) may induce areas that cannot be reached by regular distribution during self-stabilization of the overlay. The problem of fault tolerance was, up to now, not intensively discussed in the context of regular multicast distribution. Usually, a stable network structure without failures is assumed. This assumption is clearly unrealistic in the context of P2P networks.

Unstructured P2P networks with epidemic communication [21, 26] distribute multicast information by periodically contacting some randomly chosen neighbors (infection). Epidemic multicast (also known as gossip, or rumor mongering) performs the distribution using a significantly lower number of messages than aggressive flooding [10, 23, 26]. Although this method has probabilistic guarantees for reaching all nodes [7, 17], it is slower than other approaches and has – compared to 'regular' distribution – substantially higher message complexity.

For efficiency reasons, the multicast distribution can be limited by a depth or lifetime. For instance, in the rumor mongering scenario, the node stops propagating the rumor after several forwarding attempts to other nodes that have already seen it [7]. Another example is the Gnutella protocol that limits the lookup multicasts to a certain fixed depth from initiator. However, queries based on these solutions may not reach the desired peers for a computation and may be therefore ineffective.

2.4 Current Research on P2P Grids

There are several approaches using peer-to-peer technologies for (desktop) Grids. In the Organic Grid [5], peers are organized in trees. Each compute peer is responsible for finding tasks to compute. However, a mechanism for to maintaining the overlay is missing, so it is unclear how the approach can work in practice.

Compu-P2P [18] uses Chord DHTs for looking up resources. Similarly, P-Grid [20] makes use of DHT overlays, with the major difference that range queries are supported by the overlay which is essential when compute peers perform lookups for tasks with an acceptable number of CPU cycles. SWORD [25] uses DHT overlays to support range queries for available compute nodes. If fault tolerance is desired, these approaches require replication which increases the cost of maintaining the DHT considerably.

Both P3 [24] and OurGrid [3] make use of the concept of super-peers. In the former, manager peers are responsible for mapping jobs to idle peers while in the latter OurGrid Peers cooperate to provide resources to other administrative domains. In both cases, scalable fault-tolerant mechanisms for group communication are required for

the super-peers. Both papers do not address this issue.

Triana [31] is a problem solving environment built upon the JXTA framework [22]. JXTA's resource discovery is relatively slow as shown in [19], even in simple scenarios. However, JXTA is still under development, so this may change in the future. Currently, resource discovery is performed by a loosely-consistent DHT and replication.

The distributed resource machine (DRM) [21] is based on epidemic algorithms and therefore provides a fault-tolerant way for resource discovery. Independent tasks are distributed by using the epidemic information dissemination principle. As a consequence some tasks may get lost. Moreover, depending on the communication interval the time to distribute a job may be higher than with other (multicast) schemes. Decreasing the communication interval leads to reduced distribution times but also increases the permanent communication overhead.

In all the mentioned work, fault-tolerance has not been considered thoroughly or it was achieved at the expense of efficiency. Therefore, we will focus on failure-resistant but efficient resource discovery in the following sections.

3 A New Approach for Resource Discovery in P2P Grids

As discussed in the previous section, epidemic algorithms are easy to implement, provide a high fault-tolerance, and have good scaling properties. Hence they work even for large peer-to-peer networks. However, compared to other information propagation mechanisms they are slow since communication is restricted to random periodic message passing. Mechanisms based on spanning trees are much more efficient since they are optimal in respect to the number of messages required to propagate information such as job announcements. As a consequence they are not fault-tolerant and the problem of constructing a spanning tree in the overlay has to be solved. Therefore, we propose a hybrid approach that combines the fault-tolerance properties of epidemic algorithms with the efficiency of information propagation using spanning trees. The approach uses epidemic dissemination for maintaining the overlay but also includes a mechanism for multicasting information very efficiently.

3.1 Overlay Structure

Our approach differs from the epidemic algorithm described in the previous section in the following aspects:

- Each peer is assigned a (unique) id corresponding to a location on a virtual ring
- The host list of each peer stores peers with id's close to certain id's
- A spanning tree is constructed ad-hoc based on the peer id's and the entries of the peers' host lists, when a peer wishes to propagate information (such as a job announcement)

Every peer maintains an (incomplete) host list of other peers in the network. This host list is similar to the finger tables in Chord [30] and contains entries (e.g. addresses) on some other peers together with timestamps of the last successful contact to that peer. Additionally, every node n_i is associated with an id value $id(n_i) \in [0..idMax]$ (e.g. using SHA-1 as base hash function). For any two nodes n_x and n_y, the distance in the circular id space can be defined as follows:

$$dist(n_x, n_y) = \begin{cases} id(n_y) - id(n_x), & id(n_x) < id(n_y) \\ id(n_y) + (idMax - id(n_x)), & id(n_x) > id(n_y) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Every node stores in its neighbor list (finger table) ids and addresses of some other nodes nearest to 'ideal' positions in the id space. Analogously to Chord, these positions have distances of 2^k from a node's id as 'ideal' host locations for that node. Each entry in the finger table is associated with

a timestamp of the last known successful contact to that peer. The table is initialized when the node joins the network, using chord-like address lookups. Later, the nodes use periodical epidemic contacts to randomly chosen peers from the finger table to refresh timestamps and to identify 'better peers' with smaller distances to stated 'ideal' positions. The push-pull refreshing algorithm for finger tables can be summarized as follows. Regularly, once within a given time interval the following steps are performed:

1. Every peer n_i chooses a random address of some n_j from its finger table;
2. The peers n_i and n_j then exchange their host lists and add all obtained entries to their own host lists. If the host list of n_i and n_j both contain peer n_s, the timestamp of n_s is changed in both lists to the youngest value;
3. Entries older than a given maximum age (MaxAge) are removed from the list;
4. n_i then uses its list of ideal positions: $x_k : dist(n_i, x_k) = 2^k, k = 0..log_2(idMax + 1)$
5. For every ideal location $x_k \in \{x_0..x_{log_2(idMax+1)}\}$, n_i finds the nearest node in its host list using Eq. (1).
6. n_i removes from its host list all candidates that do not correspond to any ideal location.

Failure or leave of a peer does not require any administration at all. The failed address will be removed from all finger tables at latest after the maximum age is reached.

The difference to the common epidemic algorithm is the Chord-like organization of host lists. The role of epidemic contacts in our framework is twofold: (1) they ensure periodical refreshing of finger tables and (2) they increase the failure tolerance for information propagation (e.g. multicast).

3.2 Multicast for Resource Discovery

The multicast can be started by every node in the network. The multicast message contains the information to be multicasted, and a *limit* argument. This *limit* is used to restrict the forwarding space of a receiving node. When a node n_i receives the multicast message, it picks from its finger table the peer n_m nearest to the middle between its own address and *limit*: $middle = id(n_i) + 0.5 \cdot dist(n_i, limit)$ and sends to n_m the multicast message with *limit* argument (in other words, n_m then must forward the multicast between its address and *limit*). The node n_i itself continues the multicast distribution within an interval between its address and $id(n_m) - 1$ (Fig. 2). This distribution scheme follows the algorithm proposed in [9]: each node that is involved into multicast distribution halves its sending interval after every contact. This distribution scheme forms a spanning tree covering all the nodes in the system. The scheme splits the id space into disjoint distribution intervals. Thus, each node will receive the multicast message only once, in other words, there is no message redundancy. In the rest of the paper, we will refer to this method as 'regular' multicast

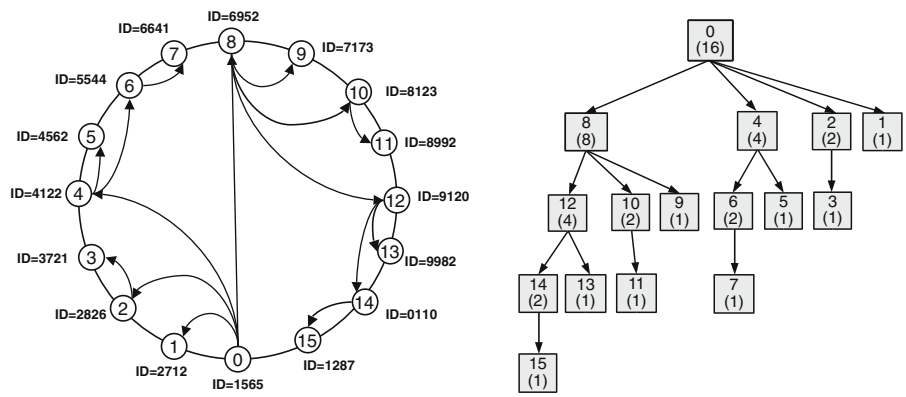

Fig. 2 Regular multicast and its distribution tree

distribution. Figure 2 illustrates the distribution tree for previously introduced example with 16 nodes.

3.3 Job Distribution

If a job is submitted by a peer which is comprised of m independent tasks, it might be unnecessary to send a multicast/broadcast message to all peers in the overlay. In the case where all peers are willing to accept the job, it is sufficient to send $m-1$ messages along the multicast tree. Hence, $\log_2(m-1)$ steps of the multicast are sufficient. If we assume that the number of peers is much larger than the average number of tasks per job, this approach can reduce communication cost drastically. However, the depth of the multicast tree and hence the number of steps of the multicast depends not only on the number of tasks of a job, but also on the load of the system. If many of peers are busy, they will not accept the tasks and hence the depth of the multicast has to be increased. In the following section, we present a model that determines the depth of the multicast depending on the load of the system. Note that even if the prediction of the desired depth is wrong, the hybrid approach is expected to be superior to epidemic algorithms alone. Consider the case that the depth is limited to $\log_2(m-1)$ (if m is the number of independent tasks of the job). Then, m nodes are informed about the job after the regular multicast. If none of the m peers are willing to accept the job, the epidemic communication has to find the right nodes for these m tasks. Having m jobs containing one task is better than having one job containing m tasks. The epidemic algorithm will be much faster if the tasks are distributed in the system.

In general, job distribution based on limited multicasts only works reliably in combination with epidemic algorithms since epidemic algorithms are required if nodes fail and also if nodes unpredictedly reject the jobs/tasks. For this reason, using Chord with immediate repair/update of the finger tables does not work reliably with limited multicast. Always a broadcast/multicast to all peers in the overlay is required to submit a job.

3.4 Mathematical Model

The job distribution by the hybrid algorithm is based on the combination of two components: the regular multicast distribution, and the epidemic information dissemination. We restrict our model to the most important case with node failures. Node failures are crucial for the distribution progress because they substantially disturb the regular distribution scheme by inducing unreachable areas. The models for node joins and simultaneous joins/failures can be constructed in an analogous manner.

The modeled problem can be formally stated as follows. Initially, the network consists of N nodes with stabilized finger tables of certain length l. The addresses of the nodes are assumed to be randomly distributed in the (sufficiently large) address space of size R. Shortly before the distribution of a new job starts, a certain fraction of nodes X fails. Without loss of generality, we may assume that each node in the network fails with probability X/N. Since the node ids in our (DHT-based Chord-like) address space can be considered as random, this assumption covers the case of correlated host failures (e.g. all peers from the same domain) as well.

3.4.1 Model of the Regular Multicast

First, we can estimate the effect of the failure of a randomly chosen node n_i in the network. The node n_i has any possible position in the tree with probability $1/N$ and fails with probability X/N. This failure can only have some negative influence on the job distribution (in the sense of live nodes that cannot be notified about the new job), when all of its predecessors in the distribution tree do not fail. When n_i resides on depth level u from the root of the distribution tree, this probability equals

$$\textit{Prob}\,(\text{all } u \text{ predecessors of } n_i \text{ are alive}) = \left(1-\frac{X}{N}\right)^{u-1} \tag{2}$$

When the node n_i with distribution interval that contains m nodes fails, it makes its $m-1$

children unreachable. However, some of these $m-1$ children may themselves fail, too. Since the nodes fail independently, the probability that exactly k nodes of $m-1$ fail, can be estimated by the binomial formula:

$$Prob\,(k \text{ out of } m-1 \text{ nodes fail}) = \binom{m-1}{k}\left(\frac{X}{N}\right)^{k}\left(1-\frac{X}{N}\right)^{m-1-k} \tag{3}$$

The expected number $Q(m-1)$ of live nodes that are located under n_i in the distribution tree can be estimated by summation over all possible k:

$$Q(m-1)=\sum_{k=0}^{m-1} Prob\,(k \text{ of } m-1 \text{ fail}) \cdot (m-1-k) \tag{4}$$

The maximum possible length of the finger table that covers the entire address space of size R is $\log_2 R$. However, this holds only for the fully populated network. It is easy to verify that in the network with N nodes that are uniformly distributed in the address space (with average distance between nodes $step = R/N$), the first $\log_2(step)$ fingers would all point to the first successor of the given node. Thus, the actual number of peers in the host list is expected to be $\log_2 R - \log_2 R/N = \log_2 N$. This number does not depend on the dimensionality R (the number of possible addresses) of the address space.

Analogously, each node that must distribute the job notification within an interval with m nodes, would split this interval into disjoint sub-intervals with $1, 2, 4, \ldots, m/2$ nodes and has $\log_2 m$ children in the distribution tree. From this follows also that the spanning tree of the regular job distribution scheme has a maximum depth of $\log_2 N$.

This behavior is illustrated by Fig. 2. For example, the distribution interval of node 8, including this node itself, contains 8 nodes. For job distribution, this node splits its interval into sub-intervals with 1, 2, and 4 nodes and has $\log_2 8 = 3$ children (nodes 9, 10, 12).

Any given node n_i that has a distribution interval with m nodes constructs its distribution subtree with some children on depth levels $1, 2, \ldots, \log_2 m$. The number of children on the depth level d from this node n_i can be recursively computed as

$$F(d,m)=\begin{cases}\log_2 m, & d=1\\ \sum_{k=0}^{\log_2 m-1} F(d-1,2^k), & d>1 \wedge m>1\\ 0, & d>1 \wedge m=1\end{cases} \tag{5}$$

Using Eq. (5), the total number of children on multiple depth levels $1, \ldots, d$ can be computed by summation using Eq. (5):

$$F^*(d,m) = \sum_{k=1}^{d} F(k,m) \tag{6}$$

We notice that for the maximum possible distribution depth $d = \log_2(m)$, the distribution tree includes all nodes in the distribution interval:

$$F^*(\log_2(m), m) = \sum_{i=1}^{\log_2(m)} F(i,m) = m-1 \tag{7}$$

Using Eqs. (4) and (7), we can also estimate the expected total number of not yet reached live nodes on depth levels $1, \ldots, d$, $d \le \log_2(m)$ by $Q(d, m-1)$ using:

$$Q(k, m-1) = Q\left(F^*(k,m)\right) \tag{8}$$

Using Eqs. (2) and (4), we can estimate the expected number of live nodes under n_i that become unreachable when n_i fails. We assume that n_i resides on level u from the root of the distribution tree and has the distribution interval with m nodes. The number of unreachable live nodes under n_i on depth levels $1, \ldots, d$, $d \le \log_2(m)$, can be estimated by

$$K(d,u,m)=\begin{cases}Q(d,m-1)\left(1-\frac{X}{N}\right)^{u-1}\frac{1}{N}, & u>0\\ 0, & u=0\end{cases} \tag{9}$$

Given the fixed distribution depth d, the expected negative influence of one failed node can be estimated by summation over all its possible positions of n_i in the tree:

$$K^*(d,u,m) = \begin{cases} K(d,u,m) + \sum_{j=0}^{\log_2 m-1} K^*(d-1, u+1, 2^j),\ m>1,\ d>1 \\ 0, \textit{else} \end{cases} \quad (10)$$

When the probability of failure X/N is known, the number of live nodes not reached in the distribution tree of depth d can be estimated as

$$E(\text{alive nodes not reached}) = X \cdot K^*(d,0,N) \quad (11)$$

By substitution of the maximum possible depth of the distribution tree $d = \log_2(N)$ and $m = N$ in the Eqs. (9) and (10), we obtain estimators for the entire distribution tree of the network.

3.4.2 Model of the Epidemic Multicast

The result of Eq. (11) is the input for the second part of distribution algorithm, the epidemic algorithm. Within each interval between epidemic contacts I_{COM}, every node contacts one of the peers in his host list and may be contacted by some other nodes. According to this push & pull distribution scheme, we assume that both types of contact are used for distribution of the job offer. Thus, within each interval I_{COM}, our model should consider two cases:

1. The node n_j contacts a previously notified node n_k;
2. The node n_j is contacted by a previously notified node n_l;

Let a and b be the total numbers of not yet notified and notified live nodes, respectively. At the beginning of epidemic job distribution, $a = X \cdot K^*(0,N)$ and $b = N - X - a$.

(1) The node n_j contacts another node n_k, randomly chosen from its host list, once a period I_{COM}. The probability to obtain the job offer from n_k equals the probability that n_k is already notified $P(\text{infected}) = \frac{b}{N}$. Consequently,

$$Prob(\neg\text{node is pull notified}) = 1 - \frac{b}{N} \quad (12)$$

(2) The node n_j is contacted by some other node n_l. We notice that n_l can contact n_j only when its address is in the host list of n_l. In other words, the address of n_j must be the first successor of some 'optimal' finger address from the host list of n_l. The expected interval between adjacent nodes in the address space of dimensionality R with N participating peers is R/N. Thus, when the finger of n_l points to some address in the interval of width R/N before n_j, this node is expected to be the first successor and would qualify for the host list of n_l. Since the intervals of $l = \log_2(N)$ fingers are expected to be disjoint, there are $l \cdot (R/N)$ positions of n_l in the address space such that n_j comes into its host list. The probability for n_l to have one of these addresses is $l \cdot (R/N)/R = l/N$. The probability for n_j to be chosen by n_l for the next contact from its host list of length l equals $l/N \cdot 1/l = 1/N$.

Therefore,

$$Prob(\neg\text{node is push notified}) = 1 - \frac{1}{N} \quad (13)$$

Assuming that all epidemic contacts are independent, the probability for n_j to be notified about the job offer within interval I_{COM} by at least one of b previously notified nodes, can be estimated as

$$Prob(\text{node is notified}) = 1 - \left(1 - \frac{1}{N}\right)^b \cdot \left(1 - \frac{b}{N}\right) \quad (14)$$

The expected total number of notified nodes after I_{COM} can be estimated as

$$b_{new} = b + a \cdot \left(1 - \left(1 - \frac{1}{N}\right)^{b} \cdot \left(1 - \frac{b}{N}\right)\right) \quad (15)$$

The further modeling of the epidemic notification can be continued in an iterative manner by substituting of b_{new} and $a_{new} = N - X - b_{new}$ as new input values for the next step.

Under the assumption that all epidemic contacts are independent, it is also possible to estimate the probability that all remaining nodes in the network will be notified within one I_{COM}:

$$Prob\,(\text{all nodes are notified}) = \left(1 - \left(1 - \frac{1}{N}\right)^{b} \cdot \left(1 - \frac{b}{N}\right)\right)^{a} \quad (16)$$

3.4.3 The Hybrid Notification Model

As mentioned above, to reduce the communication overhead for job announcement, we limit the regular multicast in a way that the resulting spanning tree has a limited depth. The model of hybrid notification allows us to estimate the required distribution depth for job propagation. We assume that the job consists of J tasks that can be independently processed by nodes of the P2P Grid. Each node accepts on average A tasks for processing. Furthermore, we assume that a certain number X^* of nodes is busy (processing other concurrently running jobs) and does not accept new jobs; however, these nodes can forward the notification about the new job to their known peers. Without loss of generality, we can assume that each node of the network is busy with probability X^*/N. Therefore, the notification should reach at least

$$\left(1 - \frac{X^*}{N}\right) \cdot E(\text{nodes notified}) \geq \frac{J}{A} \quad (17)$$

alive peers to distribute the entire job across non-busy nodes.

The hybrid approach starts with regular notification (Section 3.4.1). The algorithm estimates the required depth of the distribution tree in order to reach the desired number of nodes (17) and initiates the multicast with corresponding depth limitation. In some situations, the maximum possible distribution depth is not sufficient to meet the requirement (17). In this case, the multicast needs to be continued after regular distribution by epidemic contacts (Section 3.4.2). The hybrid approach estimates the required duration of epidemic forwarding and includes the corresponding time limitation into the initial multicast messages.

First, we consider the regular multicast distribution. To satisfy Eq. (17), the required depth d of the distribution tree should be chosen as

$$\min(d) : \left(1 - \frac{X^*}{N}\right) \cdot \frac{F^*(d, N)}{N} \cdot (N - X - X \cdot K^*(d, 0, N)) \geq \frac{J}{A} \quad (18)$$

In the case of stable network (i.e. negligible node failures) Eq. (18) has the form

$$\min(d) : \left(1 - \frac{X^*}{N}\right) \cdot F^*(d, N) \geq \frac{J}{A} \quad (19)$$

It is possible that the condition (18) will be not satisfied for any $d \in 1, \ldots, \log_2(N)$. This means that the regular notification scheme cannot directly reach the sufficient number of non-busy nodes for job distribution. In this case, the notification should be continued by epidemic communication. The required duration t_{epid} of the epidemic notification period can be chosen accordingly to Eq. (16) using

$$t_{epid} = s \cdot I_{COM} \quad (20)$$

$$min(s) : \left(1 - \frac{X^*}{N}\right) \cdot b_s \geq \frac{J}{A} \quad (21)$$

$$b_s = b_{s-1} + a_{s-1} \left(1 - \left(1 - \frac{1}{N}\right)\right)^{b_{s-1}} \cdot \left(1 - \frac{b_{s-1}}{N}\right) \quad (22)$$

$$a_s = N - X - b_s \quad (23)$$

$$b_0 = N - X \cdot (1 + K^*(log_2(N), 0, N)) \quad (24)$$

where

- s is the minimum number of epidemic periods of length I_{COM} that is required to reach at least J/A nodes

- b_0 is the number of network nodes that were reached by regular multicast
- b_s is the number of reached nodes after s epidemic periods of length I_{COM}
- a_s is the number of remaining live nodes not reached after s epidemic periods of length I_{COM}

These formula allow for the prediction of job distribution in the proposed overlay. Hence, they are highly valuable for analyzing and evaluating the proposed approach.

4 Evaluation

We simulated several P2P Grid job distribution scenarios in order to evaluate the feasibility of the proposed hybrid approach. We therefore implemented a simulator for P2P overlays capable of simulating large networks of several thousand nodes. In the simulated networks all communication is performed asynchronously. Moreover, we consider highly dynamic P2P overlays with concurrent joining and leaving of nodes. We concentrate on the crash failure model since it is the most important model for P2P networks.

In the first two series of simulations, we focused on the fault-tolerance of the proposed approach and its scalability. Afterwards, we studied the ability of the mathematical model in Section 3.4 to predict the information distribution process. Finally in a fourth series of simulations, we evaluated the efficiency of the proposed approach for several job distribution scenarios with varying number of jobs. The evaluations were performed for various network sizes, node join/failure rates (including highly dynamic scenarios where 50% of the peers in the overlay fail or join in short time). This section shows the most characteristic results from our evaluation.

4.1 Fault-tolerance of Multicast

In the first series of simulations, we evaluated the properties of the job distribution for the proposed hybrid approach in comparison with existing epidemic and Chord-style regular multicast schemes. In this evaluation, the goal was to reach all live nodes of the network (broadcast) hence we were interested in observing the worst-case scenario where the number of tasks of a submitted job is equal to or greater than the number of peers in the system. Our metrics of interest were the ability of the broadcast to reach all nodes, its distribution time, and the resulting message complexity. We systematically simulated multiple scenarios of the dynamic behavior in P2P Grids, including:

- Constant-sized stable networks: the peers join the network at the simulation start and remain alive until the end of broadcast distribution;
- Networks with dynamic changes (pure failure/join model): an significant amount of peers joins or leaves the network within a short period of time.

4.1.1 Simulation Scenario

In our evaluations, we simulated the network architecture with following properties:

- Number of participating peers: $N = 1,024$;
- Interval between epidemic contacts: 2,000 time units (e.g. milliseconds); higher values lead to a greater gain of the hybrid method compared to the 'pure' epidemic multicast;
- Length of neighbor lists: 30 peers;
- Average network delay for message delivery: 100 time units.

In the tests, we considered three algorithms:

1. Epidemic algorithms;
2. Regular multicast (chord-style model);
3. Hybrid algorithm (both regular and epidemic);

To illustrate the behavior of multicast distribution in dynamically changing overlay networks, the multicast was initiated in every experiment at multiple fixed time points (before, during, and after the failure/join of multiple nodes). Each simulation was repeated 30 times with randomization factors: joining/leaving the network by particular nodes, order of epidemic contacts, refreshing of neighbor lists, etc. The following section summarizes averaged results for these series.

4.1.2 Results

Figure 3 shows the multicast distribution for the 'pure failure' model where 50% of nodes randomly leave the network within a short period of time (100,000 time units). The solid line ('all nodes') shows the total number of live nodes in the network. Dashed and dotted lines illustrate the progress (number of reached nodes) for different multicast algorithms. With a growing number of unavailable nodes, the regular multicast reaches significantly fewer nodes. The amount of nodes that are reached in the hybrid method by the epidemic distribution becomes higher. Since the epidemic algorithm provides no mechanisms for preliminary exclusion of failed nodes from the host lists, they remain for a longer time (up to MaxAge) in the neighbor lists and negatively influence the multicast distribution. Therefore, the hybrid algorithm distributes the multicast information significantly faster than the 'pure' epidemic communication.

Figure 4 compares the multicast distribution for the similarly constructed 'pure join' model where 50% of nodes randomly join the network within a short period of time (100,000 time units). The solid line ('all nodes') shows the total number of live nodes in the network. Dashed and dotted lines illustrate the progress (number of reached nodes) for the different algorithms. The observed behavior is similar to our first scenario with node failures; however, the regular multicast reaches a significantly larger number of nodes. The efficiency of regular multicast distribution is caused by the position lookup procedure (see Section 3.1 for details) that helps to integrate new nodes into the network within a short period of time.

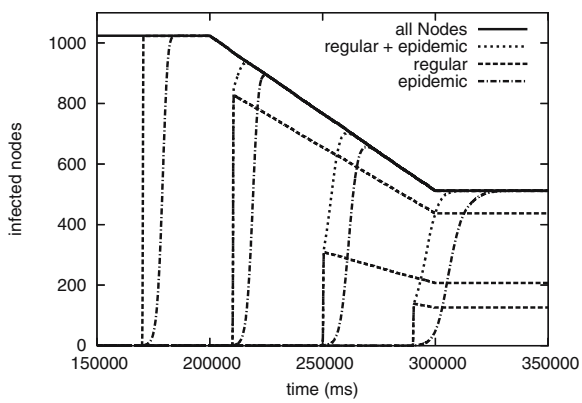

Fig. 3 Multicast progress: failure scenario for 50% nodes

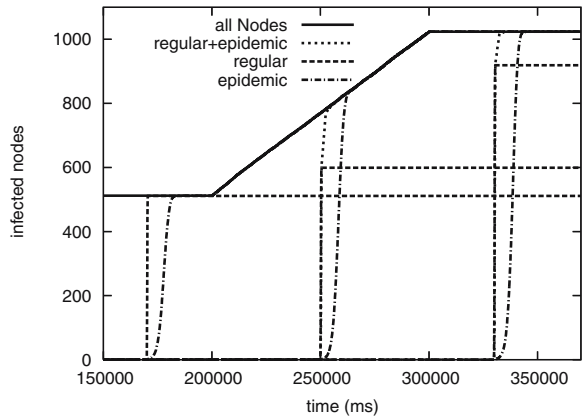

Fig. 4 Multicast progress: join scenario for 50% nodes

Figure 5 compares the message complexity for epidemic multicast and the hybrid method in the simulated 'pure failure' scenario with 50% node failure. Before the simulated failure, the hybrid algorithm distributes the jobs almost by regular messages. After the failure of half of the nodes, the amount of required epidemic messages becomes higher. However, the message complexity is consistently much lower than for epidemic communication.

Figure 6 compares the multicast distribution time for epidemic multicast and the hybrid method. It is notable that the notification time

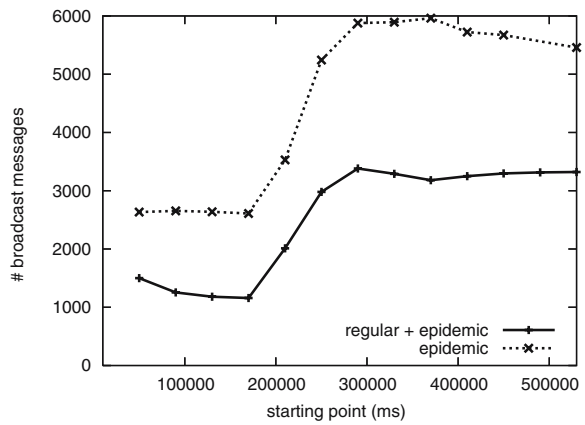

Fig. 5 Multicast message complexity (50% departing nodes)

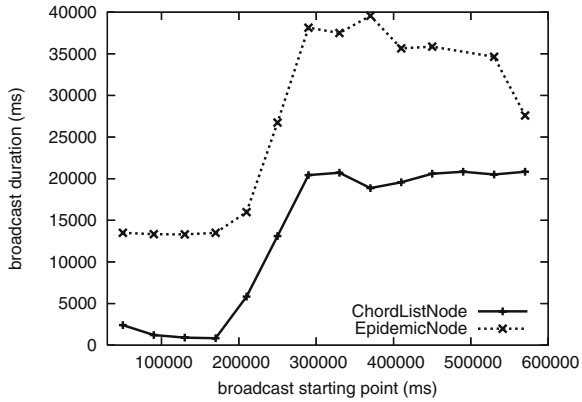

Fig. 6 Multicast duration (50% departing nodes)

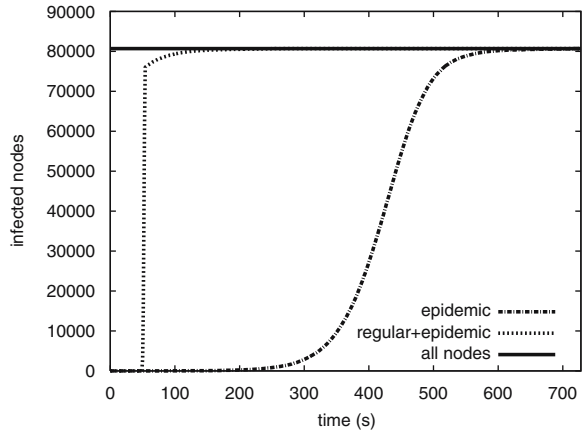

Fig. 7 Multicast progress for dynamic network with 100,000 nodes

of the hybrid method is always significantly lower than with epidemic communication.

4.2 Multicast Scalability

Basically, our hybrid approach is based on known algorithms with known scalability properties. To illustrate the scalability of the hybrid method, we additionally evaluated further scenarios for large-scale network topologies.

4.2.1 Simulation Scenario

In our evaluations, we simulated the network architecture with properties close to real large-scale P2P overlay networks:

- Interval between epidemic contacts: 5 s;
- Length of neighbor lists: 30 peers;
- Network delay for message delivery: randomly chosen between 100 and 1,000 ms;

In our tests, we compared the same job distribution algorithms as in our first evaluation:

1. Simple epidemic algorithm;
2. Regular multicast (chord-style model);
3. Hybrid algorithm.

4.2.2 Results

Figure 7 compares the multicast progress for epidemic algorithm and our hybrid method in a stable overlay network with 100,000 nodes. To model the dynamic nature of overlay networks, we simulated the environment with 20% nodes joining and 20% nodes leaving the network during simulation. The remaining peers were modeled as alive and connected to the overlay network during the whole run. This setting results in an overlay with approx. 80,000 participating live nodes at any time.

It can be observed that the epidemic job distribution (dashed line, 'epidemic multicast') is substantially slower than our hybrid method (dotted line, 'regular+epidemic'). In an ideal network without failures, the multicast could be completed by regular distribution (the almost vertical initial part of the dotted line). However, in the dynamic network with node failures, the regular distribution algorithm is unable to reach all available nodes. The vertical initial part of the dotted curve does not reach the solid line ('all nodes') that

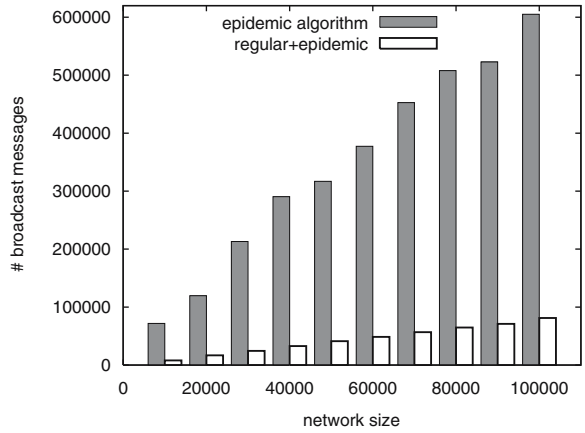

Fig. 8 Multicast message complexity for dynamic networks

corresponds to the total number of live nodes in the network. The regular multicast still reaches a substantial amount of the peers very fast; the remaining peers are notified by epidemic communication (the flatter part of the dotted curve). In further experiments, similar behavior was observed for a wide range of network sizes (from 10,000 to 100,000) and failure rates.

Figure 8 compares the multicast message complexity (the number of multicast messages exchanged between peers in order to reach all live nodes) for the stable overlay network with the same failure rate. The chart summarizes the results of multiple experiments with varying network sizes (from 10,000 to 100,000). In all experiments, the overall message complexity of the hybrid method is 10–20% higher than a regular multicast in a scenario without failures. Again, the message complexity of the hybrid method is substantially lower than the complexity of epidemic distribution; the difference rapidly grows with increasing network size.

4.3 Prediction Quality of Mathematical Model

In the third series of evaluations, we compared the predictions of the mathematical model in Section 3.4 with results from the simulation. Our tests were focused on the quality of prediction for the multicast spreading. We have systematically simulated the most critical failure scenarios of the dynamic behavior in P2P networks (a significant amount of peers leaves the network within a short period of time).

4.3.1 Simulation Scenario

In our evaluations, we simulated the network architecture with following properties:

- Number of participating peers: $N = 1,024$;
- Interval between epidemic contacts I_{COM}: 2,000 time units;
- Average network delay for message delivery: 100 time units.

To illustrate the behavior of multicast distribution in dynamically changing overlay networks, the multicast was initiated in every model at multiple fixed time points. Each simulation run was repeated 30 times with randomization factors: the order of contacts between nodes, refreshing of neighbor lists, failure points, etc. To analyze the statistical significance of deviations between predicted and observed values, we also computed 95% confidence intervals for our series:

$$\left[\bar{x} - t\left(1-\frac{\alpha}{2}; k-1\right)\cdot\frac{\sigma}{\sqrt{k}}; \bar{x} + t\left(1-\frac{\alpha}{2}; k-1\right)\cdot\frac{\sigma}{\sqrt{k}}\right] \tag{25}$$

where $\bar{x}$ and σ are common estimators for the mean and the standard deviation, and $t(1-\alpha, k-1)$ is the $(1-\alpha)$ quantile of the Student distribution (t-distribution) with $(k-1)$ degrees of freedom. In our runs, $\alpha = 0,05$ and $k = 30$ were used.

Section 4.3.2 summarizes our observations for these series.

4.3.2 Results

Figures 9 and 10 show the multicast distribution for the failure model where 30 or 50% of nodes randomly leave the network within a short period of time, but remain in the neighbor lists of other nodes. Figure 11 shows the multicast in 'ideal' network without node failures.

It is notable that the predictive model captures the multicast behavior with high accuracy. Moreover, in most cases the deviation of predicted results from the simulation is not statistically significant. The regular multicast reaches

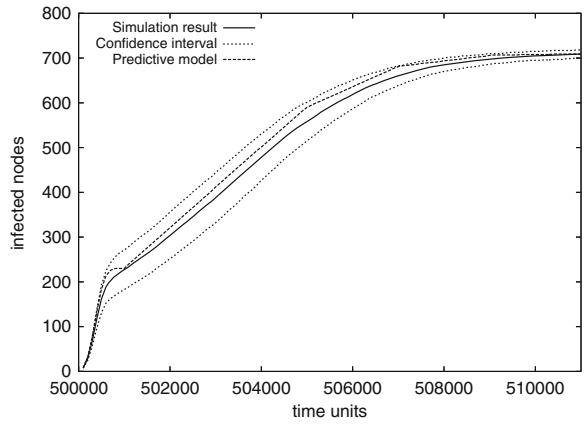

Fig. 9 Multicast progress: failure scenario for 30% nodes

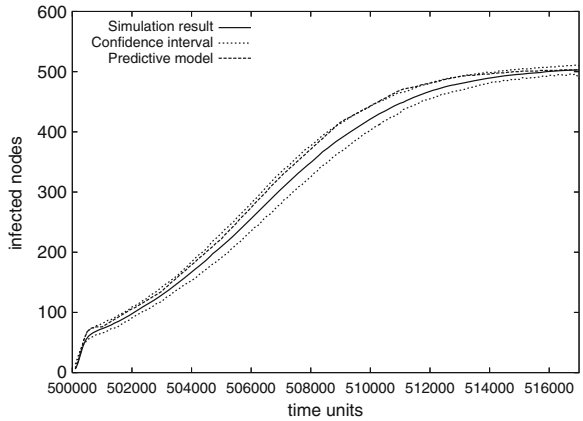

Fig. 10 Multicast progress: failure scenario for 50% nodes

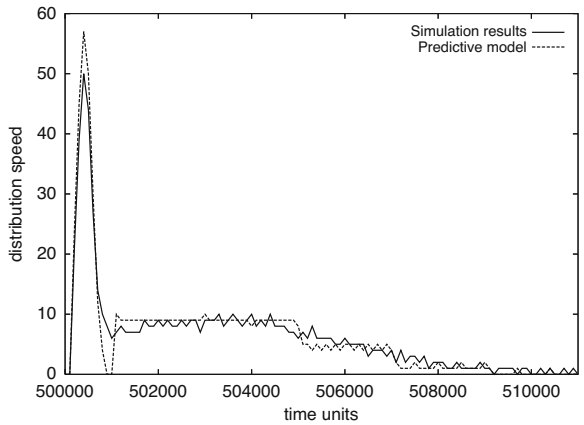

Fig. 12 The distribution speed of multicast: failure scenario for 30% nodes

most of the nodes in the middle of its distribution. In practice, this means that the regular multicast distribution can be terminated some steps earlier without substantial loss of reached nodes. It can be observed also that with a growing number of unavailable nodes, the regular multicast reaches significantly fewer nodes. The amount of nodes that are reached in the hybrid method by the epidemic distribution becomes higher.

Figure 12 compares the distribution speed of the multicast predicted by our model and the speed observed in the simulation. The measure for distribution speed is the increment of the total number of infected nodes in the network within one time unit. It can be observed that the regular multicast has, as expected, the highest distribution speed. The epidemic multicast is slower and damps after a few periods I_{COM}. This observation can be used to construct speed-related multicast termination thresholds.

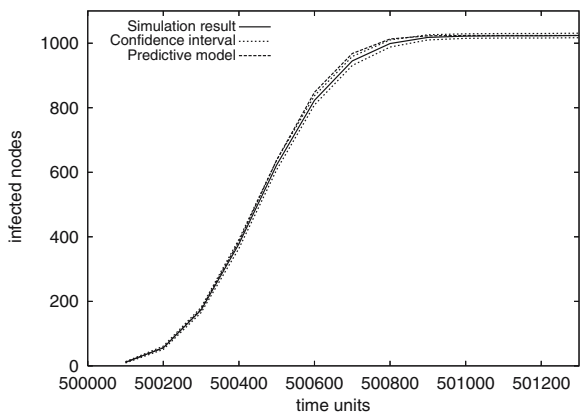

Fig. 11 Multicast progress: scenario without node failures

4.4 Evaluation of Job Distribution

In the last series of evaluations, we concentrated on the prediction quality of job distribution with the introduced theoretical model. The model predictions for various application scenarios were compared with results of multiple simulation runs for the appropriate P2P behavior. We have systematically simulated the most critical failure scenarios of the dynamic behavior in P2P networks (an significant amount of peers leaves the network within a short period of time).

4.4.1 Simulation Scenario

In our evaluations, we simulated the network architecture with following properties:

- Number of participating peers of the overlay network: $N = 1,024$
- Interval between epidemic contacts I_{COM}: 2,000 time units;
- Average network delay for message delivery: 100 time units
- Job size: $J = 2,560$ tasks
- Average capacity of the node: $A = 10$ tasks (i.e. the processing of the complete job requires on average 256 nodes)
- Number of accepted jobs per node: 1 (i.e. busy nodes do not accept further job requests)

– Processing time of one task: 100,000 time units (the processing time is substantially longer than the expected time of job distribution)

To illustrate the behavior of the job distribution in dynamically changing overlay networks, the job distribution was considered for P2P overlay networks with different load profiles X^* and with various node failure rates X. Each scenario was simulated 30 times with randomization factors: the order of contacts between nodes, refreshing of neighbor lists, failure points, etc. To analyze the statistical significance of deviations between predicted and observed values, we also computed 95% confidence intervals for our runs.

Section 4.4.2 summarizes our observations for these series. In each run, we captured the progress of job distribution (the number of free nodes that were allocated by the multicast distribution algorithm for job processing) and compared these values with results of our predictive model. Each chart shows the average distribution progress (solid line), the 95% confidence interval for simulations (dotted lines), and the distribution behavior predicted by our model (dashed line).

4.4.2 Results

In the first scenario, we considered the overlay model without node failures. To simulate various load profiles X^*, a variable number of equal-sized jobs (0, 1, or 2) was initially distributed across the network from randomly chosen starting points using the multicast distribution scheme. In the resulting overlay network with a certain number of busy nodes, we started the multicast distribution of one (additional) job from a randomly chosen node. Figures 13 and 14 illustrate the progress of job distribution in networks with 0, and 2 concurrently running parallel jobs, respectively.

In the second scenario, we simulated the dynamic failure for 30 and 50% nodes of the overlay network. The randomly chosen nodes have left the network without notification within a short period of time before job distribution, but still resided in the neighbor lists of other nodes. Figures 15 and 16 illustrate the progress of job distribution for this dynamic failure scenario and one concurrently running parallel job.

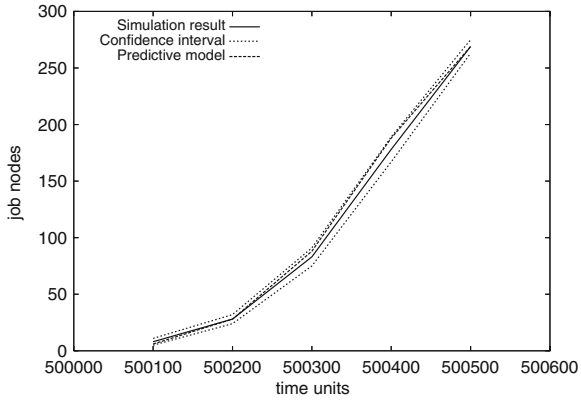

Fig. 14 Job distribution progress in the network without node failures and with two concurrent jobs

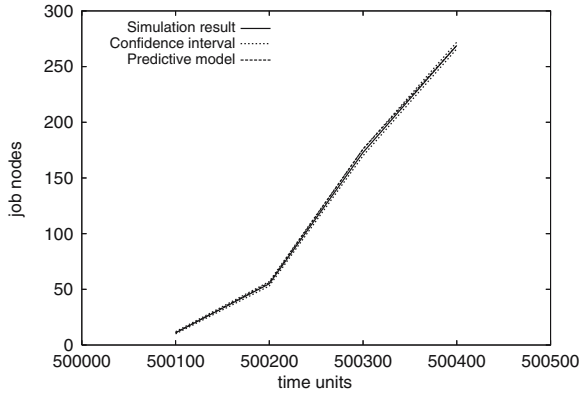

Fig. 13 Job distribution progress in the network without node failures and no concurrent jobs

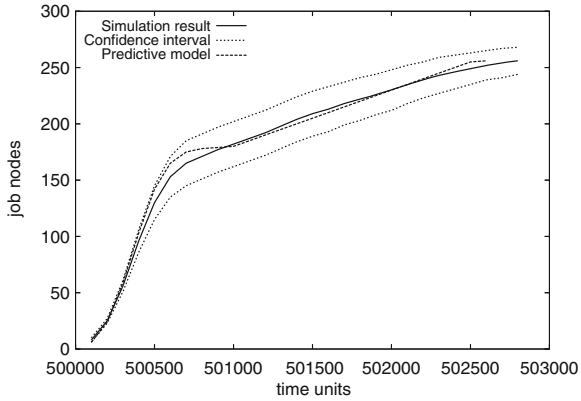

Fig. 15 Job distribution progress in a network with 30% failures and one concurrent job

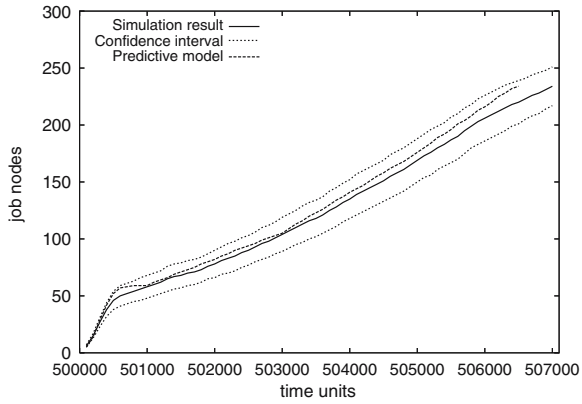

Fig. 16 Job distribution progress in a network with 50% node failures and one concurrent job

In the evaluations without node failures, the complete job distribution is performed by non-redundant regular multicast with very high distribution speed. With increasing number of busy nodes, the multicast needs to be continued longer in order to reach the sufficient fraction of free nodes and to distribute the entire job. With increasing number of failed nodes, the regular multicast reaches significantly fewer nodes. The amount of nodes that are reached by the epidemic distribution becomes higher.

The results demonstrate the high accuracy of the predictive model in comparison with the distribution progress in the simulations. In all shown scenarios, the deviation of predicted results from observations made in the simulated environment is not statistically significant.

5 Conclusions

We have presented a novel hybrid approach for resource discovery in P2P Grids – desktop Grids using peer-to-peer technology. The approach is a hybrid of epidemic information dissemination and information propagation in a structured chord-like overlay via spanning trees. Consequently, the approach combines the advantages of failure-tolerant epidemic algorithms and the efficiency of multicasts on spanning trees. We have shown in various evaluations that the approach is scalable and highly fault-tolerant. Even in scenarios with 50% node failure, the combination of the two methods is shown to be more efficient than the methods alone. Furthermore, the approach scales well with the network size as demonstrated in simulations of networks up to a size of 100,000 peers.

Moreover, we provided a mathematical model for predicting the process of information propagation in general and job distribution in particular and verified in evaluations that the model is highly accurate in predicting the processes: We demonstrated the progress of job allocation for scenarios with 0, 30 and 50% node failures and varying load. For all scenarios there was no statistically significant difference between prediction and simulation results. Hence, the model has proven to be very helpful in predicting the behavior of the proposed hybrid method and is therefore suitable for analyzing the hybrid strategy and comparing it with other strategies without the need for performing extensive simulations.

There are several issues for future work. The results enable a detailed comparison of the proposed method with other multicast approaches for peer-to-peer overlays as found in [4]. Moreover, a detailed comparison of DHT based resource discovery as realized in P-Grid and multicast-based resource discovery in case of failure/highly dynamic peer-to-peer networks has to be conducted in order to identify the strength of the different approaches. Another important issue is the case of malicious peers. We did not consider this in the presented study. Finally, we are currently working on a middleware integrating the hybrid approach as a method for resource discovery in P2P Grids.

References

1. Anderson, D.P.: BOINC: A system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing. Pittsburgh, USA (2004)
2. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@home: An experiment in public-resource computing. Commun. ACM **45**(11) (2002)
3. Andrade, N., Cirne, W., Brasileiro, F., Roisenberg, P.: OurGrid: An approach to easily assemble Grids with equitable resource sharing. In: Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel

Processing, the 12th IEEE International Symposium on High Performance Distributed Computing. Seattle, USA (2003)
4. Castro, M., Jones, M.B., Kermarrec, A.-M., Antony Rowstron, M.T., Wang, H., Wolman, A.: An Evaluation of Scalable Application-level Multicast Built Using Peer-to-Peer Overlays. Infocom 2003, San Francisco, CA (2003)
5. Chakravarti, A.J., Baumgartner, G., Lauria, M.: The organic Grid: Self-organizing computation on a peer-to-peer network. In: Proceedings of the International Conference on Autonomic Computing (ICAC '04). New York, NY (2004)
6. Chien, A., Calder, B., Elbert, S., Bhatia, K.: Entropia: Architecture and performance of an enterprise desktop Grid system. J. Parallel Distrib. Comput. **63**(5), 597–610 (2003)
7. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Scott, S., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database management. In: 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87). ACM, New York (1987)
8. distributed.net: The distributed.net Project. http://www.distributed.net. (1997)
9. El-Ansary, S., Alima, L., Brand, P., Haridi, S.: Efficient broadcast in structured peer-to-peer networks. In: International Workshop on Peer-to-Peer Systems (IPTPS). Berkeley, CA (2003)
10. Escalante, O., Perez, T., Solano, J., Stojmenovic, I.: RNG-based searching and broadcasting over internet graphs and peer-to-peer computing systems. Technical report, Universidad Nacional Autónoma de México, IIMAS (2002)
11. Eugster, P.T., Guerraoui, R., Kermarrec, A.-M., Massoulie, L.: From epidemics to distributed computing. IEEE Comput. **37**(5), 60–67 (2004)
12. Fedak, G., Germain, C., Néri, V., Cappello, F.: XtremWeb: A generic global computing system. In: Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGRID). IEEE Press, Piscataway, NJ (2001)
13. Folding@Home: FoldingHome Project Homepage. http://folding.stanford.edu. (2000)
14. GIMPS: The Great Internet Mersenne Prime Search Project. http://www.mersenne.org/prime.htm. (1996)
15. Gnutella: Gnutella Protocol v. 0.4. http://www.clip2.com/. (2000)
16. Gupta, A., Liskov, B., Rodrigues, R.: One hop lookups for peer-to-peer overlays. In: Ninth Workshop on Hot Topics in Operating Systems (HotOS) (2003)
17. Gupta, I., Kermarrec, A., Ganesh, A.: Efficient epidemic-style protocols for reliable and scalable multicast. In: IEEE International Symposium on Reliable Distributed Systems (SRDS). IEEE Computer Society, Washington, DC (2002)
18. Gupta, R., Somani, A.K.: CompuP2P: An architecture for sharing of computing resources in peer-to-peer networks with selfish nodes. In: Proceedings of the Second Workshop on the Economics of P2P Systems. Harvard University (2004)
19. Halepovic, E., Deters, R.: The costs of using JXTA. In: P2P '03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, p. 160. Washington, DC, USA. IEEE Computer Society, Los Alamitos, CA (2003)
20. Hauswirth, M., Schmidt, R.: An overlay network for resource discovery in Grids. In: Second International Workshop on Grid and Peer-to-Peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems (GLOBE'05), in conjunction with the 16th International Conference on Database and Expert Systems Applications (DEXA 2005). Copenhagen, Denmark (2005)
21. Jelasity, M., Preuss, M., Paechter, B.: A scalable and robust framework for distributed applications. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC) (2002)
22. JXTA: Project JXTA Community Homepage. http://www.jxta.org/. (2001)
23. Lin, Marzullo, Masini: Gossip versus deterministically constrained flooding on small networks. In: DISC: International Symposium on Distributed Computing. LNCS (2000)
24. Oliveira, L., Lopes, L., Silva, F.: P3 : Parallel peer to peer: An internet parallel programming environment. In: Proceedings of the International Workshop on Peer-to-Peer Computing; A workshop co-located with Networking 2002. CNR Research Area, Pisa, Italy (2002)
25. Oppenheimer, D., Albrecht, J., Patterson, D., Vahdat, A.: Design and implementation tradeoffs for wide-area resource discovery. In: Proceedings of the 14th IEEE Symposium on High Performance Distributed Computing (HPDC-14) (2005)
26. Portmann, M., Seneviratne, A.: Cost-effective broadcast for fully decentralized peer-to-peer networks. Comput. Commun. **26**(11), 1159–1167 (2003)
27. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content addressable network. Tech. Report TR-00-010, Berkeley, CA (2000)
28. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: 18th IFIP/ACM International Conference on Distributed Systems Platforms (2001)
29. Schlosser, M., Sintek, M., Decker, S., Nejdl, W.: HyperCuP - hypercubes, ontologies and P2P networks. In: Agents and Peer-to-Peer Computing, First International Workshop, AP2PC 2002, vol. 2530 of Lecture Notes in Computer Science, pp. 112–124. Springer, Berlin Heidelberg New York (2002)
30. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Guerin, R. (ed.) Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01), vol. 31, 4 of Computer Communication Review, pp. 149–160. ACM, New York (2001)
31. Taylor, I., Shields, M., Wang, I.: Resource management of triana P2P services. In: Nabrzyski, J., Schopf, J.M., Węglarz, J. (eds.) Grid Resource Management, pp. 451–462. Kluwer, Boston, MA (2004)