

Reorganization and Discovery of Grid Information with Epidemic Tuning

Agostino Forestiero ^a, Carlo Mastroianni ^a,
Giandomenico Spezzano ^a

^a*ICAR-CNR*
87036 Rende (CS), Italy
mastroianni@icar.cnr.it
phone +39-0984-831725
fax +39-0984-839054

Abstract

This paper examines a multi agent approach to spatially sort and discovery information about the resources offered by a Grid. Agents, whose behavior is inspired by ant colonies, replicate and distribute resource descriptors according to the class to which the corresponding resources belong. This facilitates resource discovery operations: query messages are attracted towards hosts that store information about a large number of resources having the required characteristics. The presented reorganization and discovery protocols feature self-organization and decentralization characteristics, since operations are performed only on the basis of local information. Agents can either *replicate* or simply *relocate* resource descriptors. These two operation modes are aimed, respectively, at fostering the *dissemination* or the *reorganization* of information. The balance between these two objectives can be modulated by setting the parameter of an ant-inspired pheromone mechanism. Balance can be static, i. e., decided a priori, or dynamic, in the case that user and network requirements change with time. In the latter case, an “epidemic” mechanism is used to communicate the value of this parameter to the hosts and agents of the Grid. Simulation analysis confirms the effectiveness of the reorganization and discovery protocols and of the mentioned epidemic tuning mechanism.

1 Introduction

Grid computing [13] is an emerging computing model that provides the ability to perform higher throughput computing by taking advantage of many networked computers and distributing process execution across a parallel infrastructure. Modern Grids are based on the service oriented paradigm; for example, in the Globus Toolkit 4 based on the Web Services Resource Framework (WSRF [24]), resources are offered through the invocation of Web services, which boast enriched functionalities such as lifecycle and state management.

Grids are becoming more and more popular, but their effective usage is made problematic both by their ever increasing size and by the heterogeneity of hosts and resources of which they are composed. In this scenario, an urgent issue is the deployment of an information system featuring decentralized characteristics, opposed to the centralized or hierarchical information systems which are today provided by most Grid platforms, such as those based on the WSRF framework.

This paper discusses a novel approach for the construction of a Grid information system which allows for an efficient management and discovery of resources. The approach, introduced in [12] in its basic version, exploits the features of (i) epidemic mechanisms tailored to the dissemination of information in distributed systems [20] and (ii) self adaptive systems in which “swarm intelligence” emerges from the behavior of a large number of agents which interact with the environment [4, 8]. This approach is partly inspired by biological systems, such as ant and termite colonies, which boast self-organizing, decentralized and scalable features that can profitably be exploited in distributed computer systems and specifically in Grids.

It is assumed that the resources are classified into a number of classes, according to their semantics and functionalities. The rationale of this classification is that users generally issue a query not to search for a single specific resource, but to collect information about resources having specified characteristics [7], for example a host with a given CPU power or a bioinformatic software able to perform particular operations on protein data. A *class of resources* is therefore defined as a set of Grid services/resources having specified properties. After issuing a query, a user can discover a number of resources of a given class, and then can choose the resources which are the most appropriate for his/her purposes.

As most Grid and P2P information systems, the bio-inspired system described in this paper does not cope with actual resources, but handles metadata documents, or *descriptors*, that provide relevant information about Grid resources (for example, name, author, publisher, location, access policy, etc.), in order to

advertise their presence and foster their use. A descriptor can be composed of a syntactical description of the service (i.e. a WSDL - Web Services Description Language - document) and/or an ontology description of service capabilities. Specifically, two protocols are presented in this paper: the ARMAP protocol, for the replication and reorganization of resource descriptors, and the ARDIP protocol, for resource discovery.

The ARMAP protocol (*Ant-based Replication and MApping Protocol*) disseminates resource descriptors in a controlled way, by spatially sorting (or *mapping*) such descriptors according to the class to which the corresponding resources belong. Each ARMAP agent travels the Grid through peer-to-peer (P2P) interconnections among Grid hosts, and uses simple probability functions to decide whether or not to *pick* descriptors from or *drop* descriptors into the current Grid host. Reorganization of descriptors results from pick and drop operations performed by a large number of agents. A self-organization approach based on ant pheromone [27] enables each agent to tune its *operation mode* on the basis of local information. Indeed, each agent can work in the *copy* mode (in this mode, it can generate new descriptor replicas), or in the *move* mode (it can only move descriptors from one host to another, without generating new replicas).

The ARMAP protocol can effectively be used to build a Grid information system in which (i) descriptors are properly replicated and (ii) the overall entropy is reduced. These two features can be contrasting, as shown in the evaluation section, so they must be correctly balanced. The replication degree is an important parameter, as the presence of many replicas can reduce the search latency but the system may consume more memory resources and increase the communication costs required for replica updates [29]. Therefore, the replication degree should be properly set in order to balance the search performance gain and the resource overhead. In this work, this goal is achieved by regulating a parameter of the pheromone mechanism which is maintained by each agent. The value of this parameter affects the time interval in which the agent operates under the *copy* mode. This mechanism can be tuned in a *static* or *dynamic* fashion. In the case of static tuning, the value of the mentioned parameter is set before ARMAP protocol is started, whereas, in the case of dynamic tuning, it can be tuned by a supervisor agent while ARMAP is running. This introduces a twofold control mechanism: each agent uses local information to self-regulate its activity on the base of the parameter value that it stores. Moreover, a supervisor agent can decide to increase or decrease the value of this parameter for all agents, in order to foster or limit the replication of descriptors. To achieve this, the new value of the pheromone parameter is spread on the Grid via an *epidemic* mechanism [10]. The supervisor initially communicates a new value of the parameter only to the peer in which such agent resides. Since then, each agent that visits this “infected” peer will also be infected and its own parameter will be changed. In turn, whenever an in-

ected agent visits a non-infected peer, the latter will be contaminated and will subsequently infect other agents. So, in a short time, all or most agents will be infected with the new value of the pheromone parameter.

The ARDIP (*Ant-based Resource Discovery Protocol*) protocol is a semi-informed discovery protocol that exploits the logical resource reorganization achieved by ARMAP. The rationale is the following: if a large number of descriptors of a specific class are accumulated in a restricted region of the Grid, it is convenient to drive search requests (issued by hosts to search for descriptors of that class) towards that region, in order to maximize the number of discovered descriptors and minimize the response time. An ARDIP discovery operation is performed in two phases. In the first phase a *blind* mechanism, specifically the *random walk* technique [19], is adopted: a number of query messages are issued by the requesting host and travel the Grid through the P2P interconnections. In the second phase, whenever a query gets close enough to a Grid region which is collecting descriptors of the needed class of resources, the search becomes *informed*: the query is driven towards this Grid region and will easily discover a large number of useful descriptors.

Simulation analysis shows that ARMAP and ARDIP protocols, if used together, allow a very high effectiveness in discovery operations to be achieved. The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the ARMAP protocol and Section 4 analyzes its performance, in particular when using the epidemic mechanism that enables dynamic tuning. Subsequently, Section 5 and 6 introduce and analyze the ARDIP protocol for resource discovery. Conclusions are given in Section 7.

2 Related Work

In most Grid frameworks deployed so far, the information system is structured according to centralized or hierarchical approaches, mostly because the client/server approach is still used today in the majority of distributed systems and in Web services frameworks. For example, the version 4 of the Globus Toolkit, the standard de facto Grid framework, is based on the Web Service Resource Framework, WSRF [24]. The central component in the GT4 information system is the Index Service, which collects information about Grid resources and makes this information available to users and clients. An Index Service retrieves information from multiple data sources or other Index Services, giving the possibility of building the information system according to hierarchical, decentralized or hybrid architectures. However, the hierarchical model is still the most frequently used [21].

Nowadays, the research and development community agrees that the centralized approaches are becoming unbearable, since they create administrative bottlenecks and are not scalable. Conversely, the adoption of decentralized approaches, like the P2P paradigm, can favor Grid scalability [14] [23]. A hierarchical information system can be viable within a single Organization or in a small-scale Grid, but it can become impractical in a large multi-institutional Grid for several reasons, among which:

- fault-tolerance is limited by the presence of a bottleneck at the tree root;
- a significant amount of memory space must be reserved in high level information servers to store information about a large number of resources, limiting the scalability of the Grid;
- information servers belonging to different levels must carry very different computation and traffic loads, which leads to challenging problems concerning load imbalance;
- the hierarchical organization can hinder the autonomous administration of different Organizations.

Novel approaches for Grid management, and in particular for the construction of a scalable and efficient information system, should have the following properties: (i) self-organization (meaning that Grid components are autonomous and do not rely on any external supervisor), (ii) decentralization (decisions are to be taken only on the base of local information) and (iii) adaptivity (mechanisms must be provided to cope with dynamic characteristics of hosts and resources).

Requirements and properties of Self Organizing Grids are sketched in [9]. Some of the issues presented in the latter paper are concretely applied in this work: for example, clustering of resources in order to facilitate discovery operations, election of representative nodes within clusters, adaptive dissemination of information.

A self-organizing mechanism is also exploited in [5] to build an adaptive overlay structure for the execution of a large number of tasks in a Grid. Similarly to the latter work, the ARMAP and ARDIP protocols discussed in this paper exhibit several characteristics of both biological systems and mobile agent systems (MAS), as discussed in the Introduction. In many aspects these protocols are specifically inspired by ant algorithms, which aim to solve very complex problems by imitating the behavior of some species of ants [4].

The two main objectives of ARMAP and ARDIP are the replication and dissemination of resource descriptors and the their efficient discovery. These issues are obviously correlated, since an intelligent dissemination can facilitate discovery. Nevertheless, *information dissemination* is also functional to other important requirements, such as fault tolerance, load balancing, re-

duced access latency and bandwidth consumption. Information dissemination is indeed a fundamental and frequently occurring problem in large, dynamic and distributed systems. In [6], the authors examine a number of techniques that can improve the effectiveness of blind search by proactively replicating data. In particular, two natural but very different replication strategies are described: uniform and proportional. The uniform strategy, replicating everything equally, appears naive, whereas the proportional strategy, where more popular items are more replicated, is designed to perform better but fails to do so. Actually, it is shown that any strategy that lies between the two performs better than the two extreme strategies. Gossip-based schemes have attracted interest as a solution for information dissemination, as they are scalable, easy to deploy and resilient to network and process failures. However, they have two major drawbacks [17]: (i) they rely on each peer having knowledge of the global membership and (ii) being oblivious to the network topology, they can impose a high load on network links when applied to wide-area settings. In [1] information dissemination is combined with the issue of effective replica placement, since the main interest is to place replicas in the proximity of requesting clients by taking into account changing demand patterns. Specifically, a metadata document is replicated if its demand is higher than a defined threshold and each replica is placed with a multicast mechanism that aims to discover the data server which is the closest to demanding clients. In [15] it is proposed to disseminate information selectively to groups of users with common interests, so that data is sent only to where it is wanted. In our paper, instead of classifying users, the proposal is to exploit the classification of resources: descriptors are replicated and disseminated with the purpose of creating regions of the network that are specialized in specific classes of resources.

As for the *resource discovery* issue, the ARDIP algorithm puts itself along the research avenue of P2P resource discovery protocols. Such protocols can be classified into *blind* and *informed* [26]. If nodes have no information on where the resources are actually located, a search request is performed through a random exploration of the network; therefore a *blind* search mechanism is adopted, such as “flooding” or “random walk” [19]. If a centralized or distributed information service maintains information about resource location, it is possible to drive search requests through an *informed* mechanism, such as “routing indices” [7] or “adaptive probabilistic search” [25].

The ARDIP protocol can be categorized as *semi-informed*, since it combines the benefits of both *blind* and *informed* resource discovery approaches. In fact, a pure blind approach is simple to implement but has limited performance and can cause an excessive network load, whereas a pure informed approach (e.g. based on routing indices [7] or adaptive probabilistic search [25]) generally requires a very structured resource organization which is impractical in a large, heterogeneous and dynamic Grid.

Finally, ARMAP and ARDIP assume the pre-existence of an algorithm for the classification of resources. This assumption is common in similar works: in [7], performance of a discovery technique is evaluated by assuming that resources have been previously classified in 4 disjoint classes. Classification can be made by characterizing the resources with a set of parameters that can have discrete or continuous values. Classes can be determined with the use of Hilbert curves that represent the different parameters on one dimension [2]; alternatively, an n-dimension distance metric can be used to determine the similarity among resources [18].

3 Reorganization of Resource Descriptors

The aim of the ARMAP (*Ant-based Replication and Mapping Protocol*) protocol is to achieve a logical organization of Grid resources by spatially sorting related descriptors on the Grid according to the class to which they belong. It is assumed that Grid resources are categorized into a number of classes, which will be referred to as N_c .

When joining or connecting to the Grid, a host generates an agent with probability P_{gen} , and sets the life time of this agent to T_{peer} , which is the value of the mean connection time of the host, calculated on the basis of the host's past activity. This mechanism allows for controlling the number of agents that operate on the Grid and assures a regular turnover of such agents. Indeed the number of agents is maintained to a value which is about P_{gen} times the number of hosts.

3.1 Agent Operations

Agents perform *pick* and *drop* operations to replicate and move descriptors from one host to another. Each host can store descriptors of local resources, as well as descriptors of resources published by other hosts. When distinction is relevant, such descriptors will respectively be referred to as *local* and *remote* descriptors.

The ARMAP algorithm is defined as follows. Periodically, each ARMAP agent performs a small number of P2P hops among Grid hosts. Whenever an agent arrives at a new Grid host, for every resource class, it evaluates the pick or drop probability function, specifically: (i) if the agent does not carry any descriptor of this class, it evaluates the *pick probability function*, so as to decide whether or not to pick the descriptors of this class from the current host; (ii) if the agent already carries some descriptors of this class, it evaluates the *drop*

probability function, so as to decide whether or not to leave these descriptors in the current host. After picking the descriptors of a class, the agent will carry them until it drops them into another host, and then will try to pick other descriptors from another host.

The pick probability function P_{pick} is defined with the intention that the probability of picking the descriptors of a given class decreases as the local region of the Grid accumulates such descriptors and vice versa. This assures that as soon as the equilibrium condition is broken (i.e., descriptors belonging to different classes begin to be accumulated in different regions), the reorganization of descriptors is increasingly fostered.

The P_{pick} function, reported in formula (1), is the product of two factors, which take into account the relative accumulation of descriptors of a given class (with respect to the other classes), and their absolute accumulation (with respect to the initial number of descriptors of that class). The f_a fraction, an index of the absolute accumulation, is computed as the number of *local* descriptors of the class of interest, stored by the hosts located in the *visibility region*, out of the overall number of descriptors of the class of interest (i.e., both *local* and *remote*) that are stored by the same hosts. The *visibility region* includes all the hosts that are reachable from the current host with a given number of hops, i.e. within the *visibility radius* R_v . As more *remote* descriptors of a class are accumulated in the local region, f_a decreases, and the first factor of the pick function decreases as well (and vice versa), which is the desired behavior. Conversely, the f_r fraction is computed as the number of descriptors of the class of interest, accumulated in the hosts located in the visibility region, divided by the overall number of descriptors that are accumulated in the same region.

The parameter k_1 and k_2 , whose values are comprised between 0 and 1, can be tuned to modulate the value of the probability function. For example, the value of the second factor is equal to 0.25 when f_r and k_2 are comparable, while it approaches 1 when f_r is much lower than k_2 (i.e., when the descriptor under evaluation is an outlier in the local region) and 0 when f_r is much larger than k_2 (i.e., if the descriptor belongs to the same class as most other descriptors in this region). k_1 and k_2 are both set to 0.1, as in [4].

$$P_{pick} = \left(\frac{f_a}{k_1 + f_a} \right)^2 \cdot \left(\frac{k_2}{k_2 + f_r} \right)^2 \quad (1)$$

The *pick* operation can be performed with two different modes. If the *copy* mode is used, the agent, when executing a pick operation, leaves the descriptors on the current host, generates a replica of them, and carries such replicas until it will drop them in another host. Conversely, with the *move* mode, as an agent picks the descriptors, it removes them from the current host, thus

preventing an excessive proliferation of replicas.

As well as the pick function, the *drop* function is first used to break the initial equilibrium and then to strengthen the mapping of descriptors belonging to different classes in different Grid regions. As opposed to the pick operation, the drop probability function P_{drop} , shown in formula (2), is proportional to the relative accumulation of descriptors of the class of interest in the visibility region. In (2) the parameter k_3 is set to a higher value than k_1 and k_2 , specifically to 0.3, in order to limit the frequency of drop operations. Indeed, it was observed that if the drop probability function tends to be too high, it is difficult for an agent to carry a descriptor for an amount of time sufficient to move it into an appropriate Grid region.

$$P_{drop} = \left(\frac{f_r}{k_3 + f_r} \right)^2 \quad (2)$$

3.2 System Entropy and Self-Organization of Agents

A spatial entropy function, based on the well known Shannon's formula for the calculation of information content, is defined to evaluate the effectiveness of the ARMAP protocol.

For each peer p , the local entropy E_p , defined in formula (3), gives an estimation of the extent to which the descriptors have been mapped within the visibility region centered in p . In (3), $f_r(i)$ is the fraction of descriptors of class C_i ($i = 1 \dots C_n$) that are located in the visibility region with respect to the overall number of descriptors located in the same region. E_p is normalized, so that its value is comprised between 0 and 1. In particular, an entropy equal to 1 corresponds to the presence of comparable numbers of descriptors belonging to all the different classes, whereas a low entropy value is obtained when the region centered in p has accumulated a large number of descriptors belonging to one class, thus contributing to the spatial ordering of descriptors. As shown in formula (4), the overall entropy E is defined as the average of the entropy values E_p computed at all the Grid hosts.

$$E_p = \frac{\sum_{i=1 \dots N_c} f_r(i) \cdot \lg \frac{1}{f_r(i)}}{\lg N_c} \quad (3)$$

$$E = \frac{\sum_{p \in Grid} E_p}{N_p} \quad (4)$$

In [11] it was shown that the overall spatial entropy can be minimized if each agent exploits both the ARMAP modes, i.e. *copy* and *move*. In the first phase,

the agent *copies* the descriptors that it picks from a Grid host, but when it realizes from its own activeness that the mapping process is at an advanced stage, it begins simply to *move* descriptors from one host to another, without creating new replicas. In fact, the copy mode cannot be maintained for a long time, since eventually every host would have a very large number of descriptors of all classes, thus weakening the efficacy of descriptor mapping. The protocol is effective only if agents, after replicating a number of descriptors, switch from *copy* to *move*.

A self-organization approach based on an ant pheromone mechanism enables each agent to perform this mode switch only on the basis of local information. This approach is inspired by the observation that agents perform more operations when the system entropy is high, but operation frequency gradually decreases as descriptors are properly reorganized. Each agent increases its pheromone level when its activeness tends to decrease, and switches to the move mode as soon as the pheromone level exceeds a defined threshold T_f . In particular, at given time intervals, i.e. every 2000 seconds, each agent counts up the number of times that it has evaluated the pick and drop probability functions, $N_{attempts}$, and the number of times that it has actually performed pick and drop operations, $N_{operations}$. At the end of each time interval, the agent makes a deposit, into its pheromone base, which is inversely proportional to the fraction of performed operations. An evaporation mechanism is used to give a greater weight to the recent behavior of the agent. Specifically, at the end of the i -th time interval, the pheromone level Φ_i is computed with formulas (5) and (6). The evaporation rate E_{ARM} is set to 0.9.

$$\Phi_i = E_{ARM} \cdot \Phi_{i-1} + \phi_i \quad (5)$$

$$\phi_i = 1 - \frac{N_{operations}}{N_{attempts}} \quad (6)$$

The pheromone level can assume values comprised between 0 and 10: the superior limit can be obtained by equalizing Φ_i to Φ_{i-1} and setting ϕ_i to 1. As soon as the pheromone level exceeds the threshold T_f (whose value must also be set between 0 and 10), the agent realizes that the frequency of pick and drop operations has remarkably reduced, so it switches its protocol mode from *copy* to *move*. The value of T_f can be used to tune the number of agents that work in copy mode and are therefore able to create new descriptor replicas. Tuning can be *static* or *dynamic*, as discussed, respectively, in Section 4.2 and 4.3. Beforehand, parameters and performance indices are introduced in Section 4.1.

4 Performance of the Reorganization Protocol

4.1 Parameters and Performance Indices

The performance of the ARMAP and ARDIP protocols was evaluated with an event-based simulator written in Java. Simulation objects are used to emulate Grid peers and ant-inspired agents. Each object reacts to external events according to a finite state automaton and responds by performing specific operations and/or by generating new messages/events that are delivered to other objects.

For example, a peer visited by a replication agent gives it information about the descriptors that this peer stores; afterwards the agent uses probability functions to decide whether or not to pick descriptors from or drop descriptors into the peer. Finally, the agent sets the simulation time in which it will perform its next movement on the Grid and creates a related event that will be delivered to the destination peer at the specified time. Events are ordered in a common queue and are delivered to corresponding destination objects according to their expiration time, so that peers and agents can operate concurrently along the simulation time.

Table 1 reports the main simulation parameters used in our analysis. Grid networks having a number of hosts N_p equal to 2500 are considered in this work. Hosts are linked through P2P interconnections, and each host is connected to 4 peer hosts on average. The topology of the network is built by using the well-known scale-free algorithm defined by Albert and Barabasi [3], that incorporates the characteristic of preferential attachment (the more connected a node is, the more likely it is to receive new links) that was proved to exist widely in real networks. The number of Grid resources owned and published by a single peer is generated through a Gamma stochastic function with an average value equal to 15 [16]. The number of classes N_c in which resources are categorized is set to 5.

The average connection time of a specific peer, T_{peer} , is generated according to a Gamma distribution function, with an average value set to 100,000 seconds. The use of the Gamma function assures that the Grid contains very dynamic hosts, that frequently disconnect and rejoin the network, as well as much more stable hosts. Every time a peer disconnects from the Grid, it loses all descriptors previously deposited by agents, thus contributing to the removal of obsolete information. Moreover, a *soft state* mechanism [22] is adopted to avoid the accumulation of obsolete descriptors in very stable nodes. Each host periodically refreshes the descriptors of the resources owned by other hosts, by contacting those hosts and retrieving from them updated information about

Table 1
Simulation parameters

Parameter	Value
Grid size (number of peers), N_p	2500
Mean peer connection time, T_{peer}	100,000 s
Average number of neighbor peers	4
Mean number of resources published by a peer	15
Number of classes of resources, N_c	5
Number of agents, N_a	$N_p/2$
Mean time between two successive movements of an agent, T_{mov}	60 s
Maximum number of hops, H_{max}	3
Visibility radius, R_v	1
Pheromone threshold, T_f	3 to 10

those resources.

The probability that a host generates an agent, P_{gen} , is set to 0.5; as a consequence, the average number of ARMAP agents N_a that travel the Grid is $N_p/2$, as explained in Section 3. The average time T_{mov} between two successive agent movements (i.e. between two successive evaluations of pick and drop functions) is 60 s, whereas the maximum number of P2P hops that are performed within a single agent movement, H_{max} , is set to 3, in order to limit the traffic generated by agents. The visibility radius R_v , used for the evaluation of pick and drop functions (see Section 3.1), is set to 1, which means that these functions are based exclusively on very local information. Finally, the pheromone threshold T_f , defined in Section 3.2, is set to values ranging from 3 to 10.

A set of performance indices are defined for the performance evaluation of ARMAP. The overall entropy E , defined in Section 3.2, is used to estimate the effectiveness of the ARMAP protocol in the reorganization of descriptors. The N_d index is defined as the mean number of descriptors that are generated for each resource. The processing load L is defined as the number of agents per second that are received and processed by a single peer.

4.2 Static Tuning

A first set of experiments were performed to evaluate the performance of the ARMAP protocol and investigate the effect of *static* tuning. Static tuning is obtained by setting the pheromone threshold T_f *before* the ARMAP protocol is initiated. When ARMAP is initiated, all agents (about 1250, half the number of peers) are generated in the *copy* mode, but subsequently several agents switch to *move*, as soon as their pheromone value exceeds the threshold T_f . If the pheromone threshold T_f is increased, the average interval of time in which agents work in *copy* mode becomes longer. As a consequence, the average number of agents that work in *copy* (also called “copy agents” in the following) is larger, therefore such agents are able to create more descriptor replicas. Hence, a proper setting of the pheromone threshold is a very efficient method to enforce or reduce the generation of new replicas and the velocity and intensity of information dissemination. However, a more intense dissemination is not always associated to a better reorganization of descriptors, i.e. to a more effective spatial separation of descriptors belonging to different classes, and therefore to a lower overall entropy.

Figure 1 shows that lower values of the overall entropy are achieved with lower values of the pheromone threshold. For example, with $T_f = 3$, the value of the overall entropy decreases from the initial value of about 1 (maximum disorder) to about 0.7. As an extreme case, virtually no entropy decrease is observed if all the agents operate in *copy* ($T_f = 10$), which confirms that the mode switch of agents from *copy* to *move* is strictly necessary to perform an effective descriptor reorganization.

Figure 2 shows the mean number of descriptors generated per resource, and confirms that information dissemination is more intense if the pheromone threshold is increased, because a larger number of *copy* agents operate on the network. It can be concluded that *copy* agents are useful to replicate and disseminate descriptors, but it is the *move* agents that actually achieve the reorganization of descriptors and are able to create Grid regions specialized in specific classes of resources.

As opposed to the indices described so far, the processing load L , defined as the average number of agents per second that are received, and must be processed, by a peer, does not depend on the pheromone threshold. This index only depends on the probability that a host generates an agent, P_{gen} , and on the frequency of agent movements across the Grid, $1/T_{mov}$. Indeed, L can be obtained as follows:

$$L = \frac{N_a}{N_p \cdot T_{mov}} \approx \frac{P_{gen}}{T_{mov}} \quad (7)$$

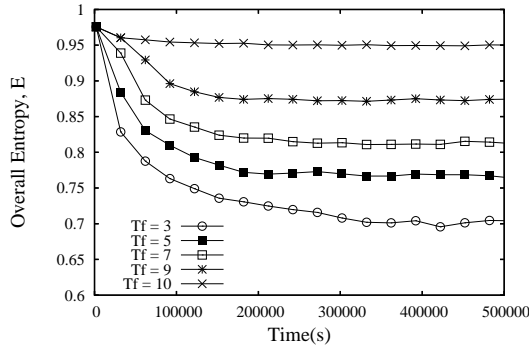


Fig. 1. Static tuning. Overall system entropy, for different values of the pheromone threshold T_f .

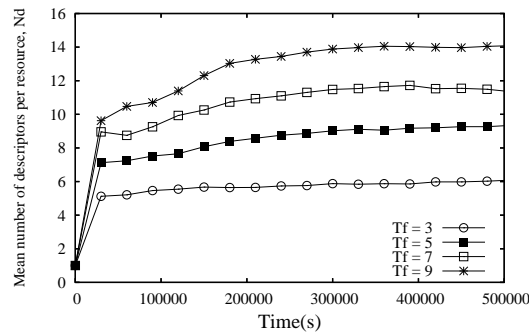


Fig. 2. Static tuning. Mean number of descriptors per resource, for different values of the pheromone threshold T_f .

In the described scenario, each host receives and processes about one agent every 120 seconds, which can be considered an acceptable load. Even though analysis is focused on a Grid with 2500 hosts, the ARMAP protocol is intrinsically scalable due to its decentralized nature, since each agent performs pick and drop operations, and tunes its operating mode, only according to local information collected by the hosts that it visits. Indeed, results not shown in this paper confirm that the value of the main performance indices (entropy and replication degree) are hardly affected by the size of the network.

4.3 Dynamic Tuning and Epidemic Mechanism

Whereas the previous subsection analyzes *static* tuning, in which the value of T_f is set before ARMAP is initiated, this section introduces *dynamic* tuning, the purpose of which is to regulate T_f while ARMAP is running. The value of T_f should be increased if more replicas are needed [29], while it should be reduced if a better spatial mapping of descriptors is desired.

Dynamic tuning can be achieved by a few *supervisor agents* that, according to the needs and the level of satisfaction of users, communicate to ARMAP agents a new value of the pheromone threshold, so as to enforce or reduce

the activity of agents. In ARMAP, an *epidemic mechanism* is exploited to transmit information to all agents. The epidemic approach is an effective solution for disseminating information in peer to peer systems [10]. This type of mechanism mimics the spread of a contagious disease in which infected entities contaminate other “healthy” entities.

ARMAP dynamic tuning works as follows. When a supervisor agent decides to change the replication degree, it communicates a new value of the pheromone threshold T_f only to the host in which such agent resides: “infection” will spread from this host. Since then, each agent that visits this infected host is contaminated and its own pheromone threshold is changed¹. In turn, whenever an infected agent visits a non-infected host, the latter is contaminated and will subsequently infect other agents. So, in a short time, all or most agents are infected with the new value of the pheromone threshold.

Figure 3 shows the number of agents that operate in the copy *mode*, referred to as N_{copy} , in the case of dynamic tuning. In this figure, dotted curves are related to the values of N_{copy} obtained, under static tuning, with $T_f = 5$ and $T_f = 9$. Continuous curves correspond to a dynamic tuning scenario: the initial threshold T_f is set to 5, but at time=400,000 seconds, T_f is changed to 9 by a supervisor agent located in one of the Grid host. The continuous line labeled with stars corresponds to an ideal scenario in which a *global control* mechanism is able to *immediately* communicate the new pheromone threshold to all agents. On the other hand, the continuous line labeled with circles is achieved by exploiting the above described epidemic mechanism. The figure shows that, with the increase in T_f , a number of agents switch from move to copy, thus increasing the value of N_{copy} . Of course, the presence of more *copy* agents fosters the replication degree of the system.

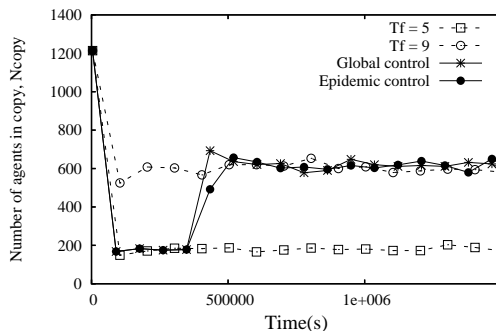


Fig. 3. Dynamic tuning. Number of copy agents N_{copy} , when the pheromone threshold is changed from $T_f=5$ to $T_f=9$. Comparison of global (ideal) and epidemic control.

This is confirmed by Figure 4, which shows the trend of the number of replicas

¹ This can affect the mode of the agent: for example, if the agent is in *move* and the new value of T_f is changed to a value that is higher than the current agent pheromone, then the agent mode switches to *copy*.

per resource N_d in the described scenario. It is noted that, after the change of T_f , the value of N_d undergoes a transition phase and then converges to the curve obtained with static tuning and $T_f = 9$. The transition phase experienced with epidemic control is slower than that measured with global control due to the time necessary to propagate information to a significant number of agents. Anyhow, the trend obtained with epidemic mechanism is very close to the trend achieved with the “ideal” global control.

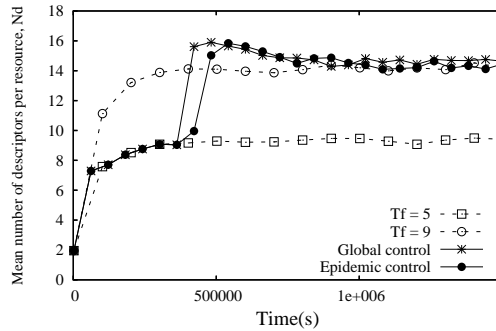


Fig. 4. Dynamic tuning. Mean number of descriptors per resource N_d , when the pheromone threshold is changed from $T_f=5$ to $T_f=9$. Comparison of global (ideal) and epidemic control.

Analogous observations can be made about the overall entropy. Figure 5 depicts the values of entropy obtained under static tuning and dynamic tuning, and compares epidemic and global control. The overall entropy increases from a value of about 0.76 with $T_f = 5$, to about 0.88 with $T_f = 9$, both with global control and epidemic control. As observed for the number of replicas, the trend of the overall entropy obtained with global control approaches the steady value more quickly than with epidemic control. However, the additional delay experienced with the epidemic mechanism is tolerable.

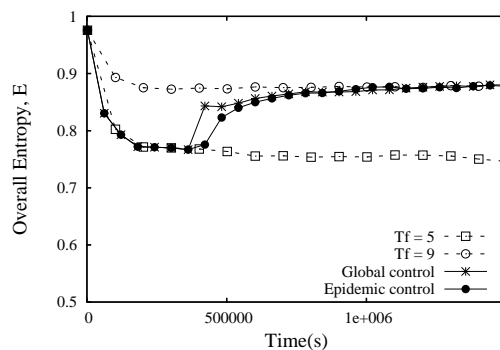


Fig. 5. Dynamic tuning. Overall system entropy E , when the pheromone threshold is changed from $T_f=5$ to $T_f=9$. Comparison of global (ideal) and epidemic control.

Overall, the epidemic mechanism is effective and requires no extra message load, since information is carried at no cost by the agents that travel the Grid. Conversely, global control requires an onerous and well synchronized mechanism to transmit information to all agents.

5 The Resource Discovery Protocol

The ARDIP (*Ant-based Resource Discovery Protocol*) protocol is used by clients to discover descriptors of Grid resources belonging to a given class. The objective is to drive an ARDIP agent, i.e. a query message, towards a region of the Grid in which the needed class of descriptors has been accumulated. Because ARDIP fully exploits the replication and spatial sorting of descriptors achieved by ARMAP, the two protocols should be used together: as ARMAP agents perform the logical reorganization of descriptors and build the Grid information system, it is increasingly likely that ARDIP agents can find a remarkable number of useful descriptors in a small amount of time.

The ARDIP protocol is based upon three main features: (a) an algorithm for the election of *representative* peers that work as attractors for query messages; (b) the semi-informed discovery algorithm; (c) a *stigmergy* mechanism that allows query messages to take advantage of the positive outcome of previous requests. These features are described in the following.

Election of representative peers

The election algorithm is completely decentralized, and every peer is autonomous in determining whether or not it should assume the *representative* role for a class of resources. The algorithm is performed by each peer periodically (i.e., every 2000 seconds), in three steps:

- (1) the peer focuses on the class for which it stores the largest number of descriptors, say class c .
- (2) the peer verifies whether, for class c , it stores a number of descriptors that is at least K_{el} times the average number of descriptors stored by a generic peer (the last value can be easily estimated, for a local region, through a simple exchange of information performed by peers). If this occurs, it means that the peer has accumulated a significant number of descriptors. In such a case, the peer elects itself as a *potential* representative peer for class c .
- (3) through a simple exchange of short-distance messages, the peer verifies if, among all the peers that are 1 and 2 hops away, there are other *potential* representative peers for the same class. In this case it compares the number of descriptors of class c with those peers. If the peer stores less descriptors than any of these neighbor peers, it abandons the election algorithm, otherwise it elects itself as a *representative peer*, and will work as an attractor for query messages issued to discover descriptors of class c .

The factor K_{el} used in the second step can be increased or decreased in order to, respectively, reduce or increase the number of representative peers. The goal of the third step is to avoid the election of several representative peers in the same region, which could give contrasting information to query messages.

Semi-informed discovery algorithm

When a user needs to discover descriptors belonging to a given class, it initiates a discovery procedure, i.e., it issues a number of *query messages*. The semi-informed discovery algorithm includes a *blind* search phase and an *informed* search phase. The random walk paradigm is used during the blind search phase: the query messages travel the Grid through P2P interconnections by following a random path. The network load is limited with the use of a TTL parameter, which is equivalent to the maximum number of hops that can be performed by a query message before being discarded.

A *blind* search procedure is switched to *informed* as soon as one of the issued query messages approaches a representative peer, i.e., when such a message is delivered to a peer which knows the existence of a representative peer and knows a route to it (see the description of the *stigmergy* mechanism below).

During the informed search phase, a query message is driven towards the representative peer and the TTL parameter is ignored, so that the query cannot be discarded until it actually reaches the representative peer. Therefore, the semi-informed walk of a query message ends in one of two cases: (i) when the TTL is decremented to 0 during the blind phase; (ii) when the query reaches a representative peer. In both cases, a *queryHit* message is created, and all the descriptors of the class of interest, which are found in the current peer, are put in this message. The *queryHit* follows the same path back to the requesting peer and, along the way, collects all the useful descriptors that are managed by the peers through which it passes.

Stigmergy mechanism

The “stigmergy” paradigm, often observed in biological systems, allows elementary entities to communicate with each other not directly, but through the shared environment. For example, in ant colonies, an ant that finds a food source leaves a *pheromone* along its way back to the nest, and such a pheromone will alert other ants to the presence of the food source. The ARDIP protocol exploits a similar mechanism: when a query message accidentally gets to a representative peer for the first time, the returning *queryHit* will deposit an amount of pheromone in the peers that it encounters as it retreats from the

representative peer. In this paper, it is assumed the pheromone is deposited only in the first two peers of the *queryHit* path.

When a query message gets to a peer during its blind search, it checks the amount of pheromone that is deposited there. If the pheromone exceeds a threshold, T_{ARD} , it means that a representative peer is close enough, and the discovery procedure becomes *informed*. An evaporation mechanism assures that the pheromone deposited on a peer does not drive *queryHits* to ex-representative peers. The pheromone level at each peer is computed every time interval of 5 minutes. The amount of pheromone Φ_i , computed after the i -th time interval, is given by formula (8).

$$\Phi_i = E_{ARD} \cdot \Phi_{i-1} + \phi_i \quad (8)$$

The evaporation rate E_{ARD} is set to 0.9; ϕ_i is equal to 1 if a pheromone deposit has been made in the last time interval by at least one agent, otherwise it is equal to 0. The threshold T_{ARD} is set to 2. With these settings, the threshold is exceeded as soon as a few *queryHits* deposit their pheromone at different time intervals, while the algorithm is more conservative when it has to recognize that a representative peer has been “downgraded”: up to 15 time intervals are necessary to let this level assume a value lower than the threshold.

6 Performance of Resource Discovery

This section evaluates the performance of the ARDIP protocol and shows that discovery operations are more and more effective as descriptors are re-organized by ARMAP agents. Table 2 reports the values assumed by main ARDIP parameters, whereas performance indices are defined and explained in Table 3.

Table 2
ARDIP parameters

Parameter	Value
Number of query messages issued by the requesting peer	4
Time to live, TTL	3 to 7
Factor K_{el} , for the identification of representative peers	8
Mean message elaboration time at a peer	100 ms

Table 3
ARDIP performance indices

Performance index	Definition
F_{sq}	Fraction of queries that are successfully driven towards a representative peer
$N_{res}, N_{res}(rep), N_{res}(norep)$	Mean number of descriptors that a user discovers after its request (for all the queries, and for striking and non striking queries)
$T_r, T_r(rep), T_r(norep)$	Time interval between a request and the reception of a <i>queryHit</i> (for all the queries, and for striking and non striking queries)
$T_r(first)$	Time interval between a request and the reception of the first K results

The fraction of “striking” queries F_{sq} is essential to evaluate if representative peers are actually able to attract queries. Figure 6 proves the valuable effect caused by the combined use of ARMAP and ARDIP protocols. In fact, after a very small amount of time, the logical reorganization of descriptors produces a significant increase in F_{sq} . Moreover, F_{sq} increases with the TTL value, since a search request extends the blind phase and has more chances to get to a representative peer.

The most important performance index is N_{res} , the mean number of results that are discovered after a search request. Indeed, it is generally argued that the satisfaction of the request depends on the number of discovered descriptors returned to the user [28]. The trend for the number of results is depicted in Figure 7, which shows that it is increasingly large as descriptors are being organized by ARMAP. Notice that the mean number of results for all requests, N_{res} , is lower than that of striking queries, $N_{res}(rep)$, since it includes the requests that do not reach a representative peer (thus lowering the mean). Indeed, the striking queries can discover considerably more results than non striking queries, as Figure 7 shows.

The ARDIP protocol not only increases the number of results, but also allows users to discover them in a shorter amount of time. Figure 8 shows that the response time decreases as the ARMAP work proceeds, and also that it is notably smaller if the search request reaches a representative peer. In this case, in fact, the discovery operation is stopped even if the TTL value is still greater than 0, and a *queryHit* is immediately issued (see Section 5). This performance improvement is more evident as the TTL value increases. By comparing Figures 7 and 8, it can be seen that a larger TTL allows more

results to be obtained, but at the cost of a longer response time.

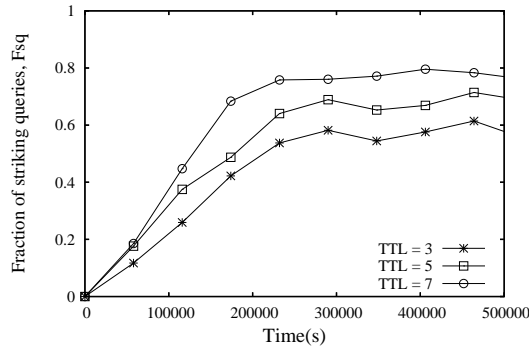


Fig. 6. Fraction of search requests that are successfully driven to a representative peer, for different values of TTL.

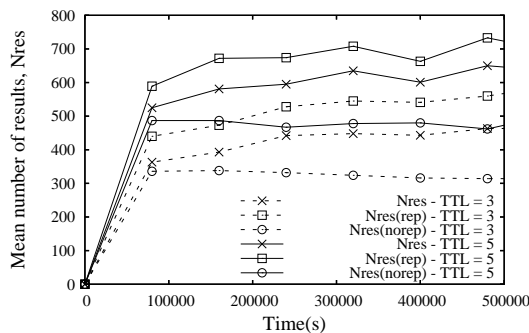


Fig. 7. Average number of results, with different values of TTL, reported for requests that reach a representative peers, for requests that do not reach a representative peers, and for all the requests.

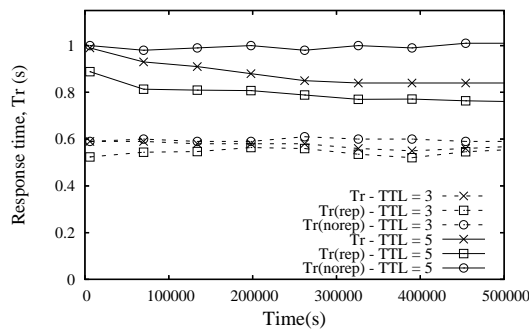


Fig. 8. Response time, with different values of TTL, reported for requests that reach a representative peers, for requests that do not reach a representative peers, and for all the requests.

If a user needs no more than a given number of results (for example, K) it is advisable to let he/she obtain them as soon as possible. To do this, the discovery protocol is modified as follows: as a query collects K results, it immediately goes back to the requesting peer, without waiting for the expiration of the TTL parameter, or to reach a representative peer. The time $T_r(\text{first})$ is calculated at the time that the first *queryHit* with K results returns to the user. It is calculated for different values of the parameter K , with the TTL

fixed at 5. Figure 9 shows that $T_r(first)$ decreases with the value of K , as expected. Comparison with the mean response time T_r , shown through a dashed line, confirms the clear saving of time that is accomplished by adopting this strategy.

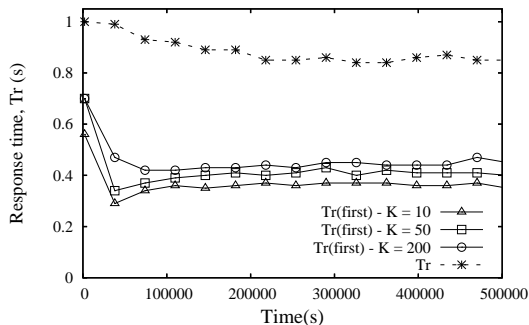


Fig. 9. Response time of the first results, $T_r(first)$, with TTL=5, for different values of the number of desired results K .

7 Conclusions

This paper introduces an approach based on the multi agent paradigm, and inspired by biological systems such as ant and termite colonies, for building an efficient information system in Grids. The approach is exploitable in very large networks, since it is fully decentralized and self-organizing. Two complex objectives, specifically reorganization and discovery of resources, are achieved through simple operations performed by two different kinds of agents. Some agents are in charge of replicating and moving resource descriptors so to create accumulation regions specialized in different classes of resources. Other agents guide query messages toward the core of such regions, which allows to discover a large number of resources belonging to a given class.

Along with the performance analysis of the protocols which drive agent operations, this paper evaluates an epidemic approach used to tune the activeness of agents. In fact replication and reorganization of descriptors can be modulated by enlarging or shortening the first phase of the life of reorganizing agents, in which agents are allowed to create new descriptor replicas. This is achieved by tuning the value of a pheromone parameter. A supervisor agent, on the basis of users' requirements, is enabled to change the value of this parameter and an epidemic mechanism is used to transmit information from peers to agents and from agents to peer. Simulation shows that this epidemic mechanism is equally effective and nearly efficient as an ideal mechanism through which all agents are immediately informed regarding the value of the pheromone parameter.

In this work, it is assumed that all the hosts have comparable storage capabilities. Current work aims at the analysis of a more realistic scenario in which

the distribution of descriptors respects the different capabilities of Grid hosts. This objective can be achieved by properly modifying the pick and drop probabilities of agents, so as to foster the pick operations in hosts with low storage capabilities and the drop operations in hosts endowed with large storage space. This way, hosts with higher storage capacity will be assigned a larger number of descriptors than low capacity hosts.

8 Acknowledgements

This work has been partially supported by the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265) and by the SFIDA-PMI project co-funded by the Italian Ministry of Research and University, MIUR (reference number 4446/ICT).

References

- [1] Mehmet S. Aktas, Geoffrey C. Fox, and Marlon Pierce. Fault tolerant high performance information services for dynamic collections of grid and web services. *Future Generation Computer Systems*, 23(3):317–337, 2007.
- [2] Artur Andrzejak and Zhichen Xu. Scalable, efficient range queries for grid information services. In *Proc. of the Second IEEE International Conference on Peer-to-Peer Computing P2P'02*, pages 33–40, Washington, DC, USA, 2002. IEEE Computer Society.
- [3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.
- [4] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York, NY, USA, 1999.
- [5] Arjav J. Chakravarti, Gerald Baumgartner, and Mario Lauria. The organic grid: self-organizing computation on a peer-to-peer network. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):373–384, 2005.
- [6] Edith Cohen and Scott Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proc. of the Special Interest Group on Data Communication ACM SIGCOMM'02*, Pittsburgh, Pennsylvania, USA, 2002.
- [7] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. of the 22nd International Conference on Distributed Computing Systems ICDCS'02*, pages 23–33, 2002.
- [8] Prithviraj Dasgupta. Intelligent agent enabled peer-to-peer search using

- ant-based heuristics. In *Proc. of the International Conference on Artificial Intelligence IC-AI'04*, pages 351–357, 2004.
- [9] Deger C. Erdil, Michael J. Lewis, and Nael Abu-Ghazaleh. An adaptive approach to information dissemination in self-organizing grids. In *Proc. of the International Conference on Autonomic and Autonomous Systems ICAS'06*, Silicon Valley, CA, USA, July 2005.
- [10] Patrick T. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5).
- [11] Agostino Forestiero, Carlo Mastroianni, and Giandomenico Spezzano. Construction of a peer-to-peer information system in grids. In Hans Czap, Rainer Unland, Cherif Branki, and Huaglory Tianfield, editors, *Self-Organization and Autonomic Informatics (I)*, volume 135 of *Frontiers in Artificial Intelligence and Applications*, pages 220–236, Amsterdam, The Netherlands, 2005. IOS Press.
- [12] Agostino Forestiero, Carlo Mastroianni, and Giandomenico Spezzano. QoS-based dissemination of content in grids. *Future Generation Computer Systems*, 24(3):235–244, 2008.
- [13] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
- [14] Adriana Iamnitchi and Ian Foster. On death, taxes, and the convergence of peer-to-peer and grid computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems IPTPS'03*, Berkeley, CA, USA, February 2003.
- [15] Adriana Iamnitchi and Ian Foster. Interest-aware information dissemination in small-world communities. In *Proc. of the 14th IEEE International Symposium on High Performance Distributed Computing, HPDC*, Research Triangle Park, NC, USA, July 2005.
- [16] Adriana Iamnitchi, Ian Foster, Jan Weglarz, Jarek Nabrzyski, Jennifer M. Schopf, and Maciej Stroinski. A peer-to-peer approach to resource location in grid environments. In *Grid Resource Management*. Kluwer Publishing, 2003.
- [17] Anne-Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, 2003.
- [18] Amr Z. Kronfol. *FASD: A Fault-tolerant, Adaptive, Scalable, Distributed Search Engine*. May 2002. PhD thesis, at <http://citeseer.ist.psu.edu/571354.html>.
- [19] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of the 16th international conference on Supercomputing ICS '02*, pages 84–95, New York, NY, USA, June 2002. ACM Press.
- [20] Karin Petersen, Mike J. Spreitzer, Douglas B. Terry, Marvin M. Theimer, and Alan J. Demers. Flexible update propagation for weakly consistent

- replication. In *Proc. of the Sixteenth ACM symposium on Operating systems principles SOSP '97*, pages 288–301, New York, NY, USA, 1997. ACM Press.
- [21] Jennifer M. Schopf, Mike D’Arcy, Neill Miller, Laura Pearlman, Ian Foster, and Carl Kesselman. Monitoring and discovery in a web services framework: Functionality and performance of the Globus Toolkit’s MDS4. Technical Report ANL/MCS-P1248-0405, Argonne National Laboratory, April 2005.
- [22] Puneet Sharma, Deborah Estrin, Sally Floyd, and Van Jacobson. Scalable timers for soft state protocols. In *Proc. of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM’97*, volume 1, pages 222–229, Washington, DC, USA, 1997. IEEE Computer Society.
- [23] Ian J. Taylor. *From P2P to Web Services and Grids: Peers in a Client/Server World*. Springer, 2004.
- [24] The Globus Alliance. The web services resource framework, WSRF, 2007. <http://www.globus.org/wsrf/>.
- [25] Dimitrios Tsoumakos and Nick Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. In *Proc. of the Third IEEE International Conference on P2P Computing P2P’03*, pages 102–109, 2003.
- [26] Dimitrios Tsoumakos and Nick Roussopoulos. A comparison of peer-to-peer search methods. In *Proc. of the 6th International Workshop on the Web and Databases WebDB’03*, pages 61–66, San Diego, California, United States, June 2003.
- [27] H. Van Dyke Parunak, Sven Brueckner, Robert S. Matthews, and John A. Sauter. Pheromone learning for self-organizing agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):316–326, 2005.
- [28] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems ICDCS’02*, 2002.
- [29] Jian Zhou, Laxmi Bhuyan, and Anirban Banerjee. An effective replication placement in p2p networks. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium IPDPS’08*, Miami, Florida, USA, April 2008.