Antares: an Ant-Inspired P2P Information System for a Self-Structured Grid

Agostino Forestiero
Institute for High Performance Computing and Networks
ICAR-CNR
87036 Rende (CS) Italy
forestiero@icar.cnr.it

Carlo Mastroianni
Institute for High Performance Computing and Networks
ICAR-CNR
87036 Rende (CS) Italy
mastroianni@icar.cnr.it

Giandomenico Spezzano
Institute for High Performance Computing and Networks
ICAR-CNR
87036 Rende (CS) Italy
spezzano@icar.cnr.it

ABSTRACT

This paper introduces Antares, a bio-inspired algorithm that exploits ant-like agents to build a P2P information system in Grids. The work of agents is tailored to the controlled replication and relocation of metadata documents that describe Grid resources.

These descriptors are indexed through binary strings that can either represent topics of interest, specifically in the case that resources are text documents, or be the result of the application of a locality preserving hash function, that maps similar resources into similar keys. Agents travel the Grid through P2P interconnections and, by the application of ad hoc probability functions, they copy and move descriptors so as to locate descriptors represented by identical or similar keys into neighbor Grid hosts. The effectiveness of the Antares algorithm has been verified by event-driven simulation which proves that ant operations allow to achieve replication and spatial sorting of descriptors.

The resulting information system is here referred to as *self-structured*, because it exploits the self-organizing characteristics of ant-inspired agents, and also because the association of descriptors to hosts is not pre-determined but easily adapts to the varying conditions of the Grid. This self-structured organization combines the benefits of both *unstructured* and *structured* P2P information systems. Indeed, being basically unstructured, Antares is easy to maintain in a dynamic Grid, in which joins and departs of hosts can be frequent events. On the other hand, the aggregation and spatial ordering of descriptors can improve the rapidity and effectiveness of discovery operations, and also enables range queries, which are beneficial features typical of structured systems.

Keywords

Ant Algorithms, Grid, Information Dissemination, Information System, Peer-to-Peer

1. INTRODUCTION

Grid computing [11] 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 [17]), resources are offered through the invocation of Web services, which boast enriched functionalities such as lifecycle and state management.

The *information system* is an important pillar of a Grid framework, since it provides information that is critical to the operation of the Grid and the construction of applications. In particular, users turn to the information system to discover suitable resources or services that are needed to design and execute a distributed application, explore the properties of such resources and monitor their availability.

Due to the inherent scalability and robustness of P2P algorithms, several P2P approaches have been recently proposed for resource organization and discovery in Grid environments [18]. The ultimate goal of these approaches is to allow users to rapidly locate Grid resources or services (either hardware or software) which have the required characteristics; this is generally reduced to the problem of finding related *descriptors*, through which it is possible to access the corresponding resources. A descriptor may contain a syntactical description of the resource/service (i.e. a WSDL - Web Services Description Language - document) and/or an ontology description of resource/service capabilities. In a Grid, after issuing a query, a user can discover a number of descriptors of possibly useful resources, and then can choose the resources which are the most appropriate for their purposes.

In P2P systems, descriptors are often indexed through bit strings, or *keys*, that can have two different meanings. The first is that each bit represents the presence or absence of a given *topic* [5, 15]: this method is particularly used if the resource of interest is a document, because it is possible to define the different topics on which this document focuses. Alternatively, a resource or service (for example a computation resource) can be mapped by a hash function into a binary string. The hash function is assumed to be locality preserving [3, 14], which assures that resources having similar characteristics are associated to similar descriptor keys. Similarity between two resources can be measured as the cosine of the angle between the bit vectors through which the corresponding descriptors are indexed.

In this paper, we propose Antares (ANT-based Algorithm for RESource management in Grids) a novel approach for the construction of a Grid information system, which is inspired by the behavior of some species of ants [2]. The Antares algorithm is able to disseminate and reorganize descriptors and, as a consequence of this, it facilitates and speeds up discovery operations. More specifically, Antares concurrently achieves multiple objectives: (i) it *replicates* and *disseminates* descriptors, with the possibility of fostering the dissemination of descriptors associated to dynamic and/or high QoS resources [1]; (ii) it *spatially sorts* descriptors, so that descriptors indexed by similar keys are placed in neighbor hosts; (iii) thanks to the self-organizing nature of the ant-based approach, the reorganization of descriptors spontaneously adapts to the ever changing environment, for example to the joins and departs of Grid hosts and to the changing characteristics of resources.

The Antares approach can be positioned along a well known research avenue whose objective is to devise possible applications of *ant algorithms*, i.e., algorithms inspired by the behavior of ants [2, 6]. We recently proposed an approach for the reorganization and discovery of resources which are pre-classified in a given number of classes [8, 9]. This enables the creation of Grid regions specialized in a particular class, which improves the performance of discovery operations, but does not allow for the spatial sorting of descriptors, thus preventing the possibility of efficiently managing range queries.

Antares has been specifically designed to tackle the case in which the access keys of resource descriptors are bit strings and, since similar resources are assumed to be mapped into similar strings, it is possible to define a similarity measure among resources, through the comparison of related keys. In this sense our work is partly inspired by the work of Lumer and Faieta [13], who devised a method to spatially sort data items through the operations of simple robots.

However, the approach of Lumer and Faieta has been adapted to our purposes, by making the following main modifications: (i) descriptors are not only sorted, as in [13], but also *replicated*, in order to disseminate useful information on the Grid and facilitate search requests; (ii) each cell can contain a number of descriptors, not just one item as in [13]; (iii) since Antares operates in a distributed computing environment, agents must limit the number of P2P hops in order to reduce traffic and computing load.

The ant-inspired agents of Antares replicate and move descriptors, and tend to place descriptors with similar (or equal) keys into the same host or neighbor hosts. This is achieved by ants' *pick* and *drop* operations which are driven by corresponding *pick* and *drop probability functions*. The obtained rearrangement and spatial ordering of descriptors facilitates resource discovery and enables range queries.

The Grid information system constructed with Antares is basically *unstructured*, in the sense that descriptors are not required to be mapped onto specified hosts (for example, hosts determined by a hash function, as in most structured P2P systems), but they are freely placed by ants through their pick and drop operations. This assures valuable features such as self-organization, adaptivity, and ultimately scalability.

Nevertheless, Antares features a self-emerging organization of descriptors, which has some "structured" properties, since descriptors are aggregated and spatially sorted. Therefore, the resulting information system is referred to as *self-structured*, because it exploits the self-organizing characteristics of ant-inspired agents, and also because the association of descriptors to hosts is not pre-determined but adapts to the varying conditions of the Grid.

In actual fact, Antares retains important benefits, which are typical of *structured* systems. In particular, our approach enables the use of a *semi-informed* discovery protocol, that is executed in two phases. In the first phase, a discovery message travels the network in a *blind* fashion, but discovery becomes *informed* as soon as this message approaches a potentially interesting region. This event is detected when the hosts encountered by the message possess descriptors whose keys are very similar (within a given range) to the key(s) specified in the query. In the informed phase, the discovery algorithm exploits the spatial sorting of descriptors and drives the discovery message towards hosts that possess descriptors which are more and more similar to those specified in the query. Therefore, the goal of the blind phase is to discover a Grid region that may later been explored with the informed mechanism to find as many descriptors as possible.

In this paper, we show that the Antares algorithm succeeds in the spatially replication and sorting of descriptors. In fact, event-based simulation proves that agents successfully generate and disseminate several replicas of each resource, and at the same time that the homogeneity of descriptors located in each small Grid region is notably increased, meaning that descriptors are effectively reorganized and sorted on the Grid.

2. THE ANTARES ALGORITHM

The main purpose of the Antares algorithm is to disseminate resource descriptors over the Grid and at the same time achieve a logical organization of Grid resources by spatially sorting the corresponding descriptors according to the their keys.

The Grid system uses P2P interconnections to enable communication and exchange of descriptors among Grid hosts. This is coherent with the recent trend of adopting P2P techniques in Grid frameworks, in order to enhance efficiency and scalability features of large-scale Grids [12, 18].

The Antares information system is progressively and continuously constructed by a number of ant-inspired agents which travel the Grid through these P2P interconnections, possibly *pick* resource descriptors from a Grid host, carry these descriptors, and *drop* them into other hosts. *Pick* and *drop* operations are based on the evaluation of the corresponding probability functions. Though these operations are very simple, and agents are unaware of the significance of what they do, a sort of *swarm intelligence* emerges from their combined work, which is typical of ant systems, and of bio-inspired systems in general.

2.1 Pick Operation

Periodically, an agent performs a small number of P2P hops among Grid hosts (see the pseudo-code of the algorithm later reported in Figure 1). Whenever an agent arrives at a new Grid host, and it does not carry any descriptor, it

[1]this possibility is not discussed here, but the reader can refer to [10] to see how this issue is tackled in a similar context.

evaluates the *pick probability function* and decides whether or not to pick one or more descriptors from the current host.

Specifically, the agent checks each single descriptor maintained in the current host, and evaluates its *average similarity* with all the descriptors maintained by the hosts located in the *visibility region*. The *visibility region* includes all the hosts that are located located within the *visibility radius*, i.e., that are reachable from the current host with a given number of hops. This radius is an algorithm parameter, and is set here to 1, in order to limit the amount of information exchanged among hosts.

Actually, the agent evaluates the similarity of the binary key of the descriptor under consideration with the keys of the *centroids* of the current host and of the neighbor hosts, and then takes the average. A *centroid* of a host is a virtual descriptor whose key is representative for the descriptors maintained in the local host. Similarity is evaluated against centroids (instead of against all single descriptors) in order to reduce the information exchanged among hosts. In fact each host must only know the keys of the centroids of the neighbor peers, instead of the keys of all the descriptors.

The probability of picking a descriptor must be inversely proportional to the average similarity of this descriptor with those located in the visibility region, thus obtaining the effect of averting a descriptor from co-located dissimilar descriptors. As soon as the possible initial equilibrium is broken (i.e., descriptors having different keys begin to be accumulated in different Grid regions), a further reorganization of descriptors is increasingly driven, because the probability of picking an "outlier" descriptor increases.

The pick probability function *Ppick*, is defined in formula (1) whereas f, defined in formula (2), measures the average similarity of a generic descriptor d with the other descriptors located in the visibility region R; the value of f assumes values ranging between 0 and 1, and so does *Ppick*. [2].

In more detail, in formula (1) the parameter k_p, whose value is comprised between 0 and 1, can be tuned to modulate the degree of similarity. In fact, the pick probability is equal to 0.25 when f and k_p are comparable, while it approaches 1 when f is much lower than k_p (i.e., when the descriptor in question is extremely dissimilar from the others) and 0 when f is much larger than k_p (i.e., when the descriptor in question is very similar to the others). Here k_p is set to 0.1.

$$Ppick = \left(\frac{k_p}{k_p + f}\right)^2 \tag{1}$$

In formula (2), the weight of each term is equal to the number of descriptors N_p maintained in each peer p, while N is the overall number of descriptors maintained in the region R, i.e., $N = \sum_{(p \epsilon R)} N_p$. Note that the similarity between d and the descriptor of the centroid of the peer p, C_p, is defined as the cosine of the angle between the corresponding key vectors. The parameter α defines the similarity scale [13]; here it is set to 0.5.

[2]actually, with the adopted values of α and k_p, the value of f can range between -1 and 1, but negative values are truncated to 0: this corresponds to have, in formula (1), a *Ppick* value of 1 in the case that the evaluated descriptor is very dissimilar from the other descriptors.

$$f(d, R) = \frac{1}{N} \cdot \sum_{p \epsilon R} N_p \cdot \left(1 - \frac{1 - cos(d, C_p)}{\alpha}\right) \tag{2}$$

After evaluating the pick probability function, the agent computes a random real number comprised between 0 and 1, then it executes the pick operation if this number is lower than the value of the pick function. As the local region accumulates descriptors having similar keys, it becomes more and more likely that "outlier" descriptors will be picked by an agent.

The *pick* operation can be performed with two different modes, *copy* and *move*. If the *copy* mode is used, the agent, when executing a pick operation, leaves the descriptor on the current host, *generates a replica* of it, and carries the new descriptor until it drops it into another host. Conversely, with the *move* mode, an agent picks the descriptor and *removes* it from the current host, thus preventing an excessive proliferation of replicas. These two modes and their impact are better discussed in Section 2.3.

2.2 Drop Operation

As well as the pick function, the *drop probability function Pdrop* is first used to break the initial equilibrium and then to strengthen the spatial sorting of descriptors. Whenever an agent gets to a new Grid host, it must decide, if it is carrying some descriptors, whether or not to drop these descriptors in the current host.

For each carried descriptor, the agent separately evaluates the drop probability function, which, as opposed to the pick probability, is directly proportional to the similarity function f defined in formula (2), i.e., to the average similarity of this descriptor with the descriptors maintained in the current visibility region.

In (3), the parameter k_d is set to a higher value than k_p, specifically to 0.5, 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.

As for the pick operation, the agent first evaluates *Pdrop*, then extracts a random real number between 0 and 1, and if the latter number is lower than *Pdrop*, the agent drops the descriptor in question into the current host.

$$Pdrop = \left(\frac{f}{k_d + f}\right)^2 \tag{3}$$

To summarize the behavior of agents, a high-level description of the Antares algorithm is given in Figure 1. Note that an agent must drop all the descriptors that it maintains before it can try to pick other descriptors from a new host.

As a final remark concerning pick and drop probability functions, it is worth specifying that the values of mentioned parameters (k_p, k_d, α) have an impact on the velocity and duration of the transient phase of the Antares process, but they have little influence on the performance observed under steady conditions.

```
// $N_a$ = number of agents
// $H_{max}$ = max number of P2P hops that an agent
// can perform between two successive operations
// $mod$ = mode of the algorithm (copy or move)
For each agent $a$ do forever {
    compute integer number $h$ between 1 and $H_{max}$;
    $a$ makes $h$ P2P hops in a random direction;
    if ($a$ is unloaded) { // try pick operations
        for each descriptor $d$ of the current peer {
            compute Ppick, as in formula (1);
            draw a random real number $r$ between 0 and 1;
            if ($r$<=Ppick) then {
                pick the descriptor $d$ from current host;
                if (mod == move)
                    remove descriptor $d$ from current host;
            }
        }
    }
    else { // try drop operations
        for each descriptor $d$ carried by the agent {
            compute Pdrop, as in formula (3);
            draw a random real number $r$ between 0 and 1;
            if ($r$<=Pdrop) then
                drop descriptor $d$ into the current host;
        }
    }
}
```

Figure 1: The Antares algorithm

2.3 Spatial Sorting of Descriptors

The effectiveness of Antares is evaluated through a spatial *homogeneity function H*. Specifically, for each host of the Grid, we calculate the homogeneity among all the descriptors maintained within the local visibility region, by averaging the cosine of the angle between every couple of descriptors. Afterwards, the values of the homogeneity functions calculated on all the hosts of the network are averaged. The objective is to increase the homogeneity function as much as possible, because it would mean that similar descriptors are actually mapped and aggregated into neighbor hosts, and therefore an effective sorting of descriptors is achieved.

Simulation analysis (some details about simulation are given in Section 3) has shown that the overall homogeneity function is better increased if each agent works under both its operational modes, i.e., *copy* and *move*. In the first phase of its life, an agent is required to *copy* the descriptors that it picks from a Grid host, but when it realizes from its own activeness that the sorting 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 maintain a very large number of descriptors of all types, thus weakening the efficacy of spatial reorganization. The algorithm is effective only if each agent, after replicating a number of descriptors, switches from *copy* to *move*.

A self-organization approach based on the concept of *stigmergy* [4] 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 (because descriptors are distributed randomly), but operation frequency gradually decreases as descriptors are properly reorganized. The reason of this is that the values of *Ppick* and *Pdrop* functions, defined in formulas (1) and (3), decrease as descriptors are correctly replaced and sorted on the Grid.

With a mechanism inspired by ants and other insects, each agent maintains a *pheromone base* (a real value) and increases it when its activeness tends to decrease; the agent switches to the *move* mode as soon as the pheromone level exceeds a defined threshold T_h. In particular, at given time intervals, i.e. every 2,000 seconds, each agent counts up the number of times that it has evaluated the pick and drop probability functions, and the number of times that it has actually performed pick and drop operations (see algorithm in Figure 1). At the end of each interval, the agent makes a deposit into its pheromone base, by adding an amount of pheromone equal to 1 minus the ratio between the number of operations actually performed and the total number of operation attempts. This way the value of the pheromone base results to be inversely proportional to the activeness of the agent. An evaporation mechanism is used to give a higher 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 formula (4).

$$\Phi_i = E_v \cdot \Phi_{i-1} + \phi_i \qquad (4)$$

The evaporation rate E_v is set to 0.9, whereas ϕ_i is the amount of pheromone deposited in the last time interval. 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_h (whose value must also be set between 0 and 10), the agent switches its mode from *copy* to *move*. The value of T_h can be used to tune the number of agents that work in the *copy* mode, and consequently the replication and dissemination of descriptors, since these agents are able to generate new descriptor replicas. Specifically, the number of agents in *copy* increases with the value of the threshold T_h. This phenomenon is not analyzed here but is widely discussed in [7].

2.4 Management of Peer Disconnections

In a dynamic Grid, peers can go down and reconnect again with varying frequencies. To account for this, we define the *average connection time* of a peer, which is generated according to a Gamma probability function, with an average value set to the parameter T_{peer}. Use of the Gamma distribution assures that the Grid contains very dynamic hosts, which frequently disconnect and rejoin the network, as well as much more stable hosts.

As a consequence of this dynamic nature, two issues are to be tackled. The first is related to the management of new resources provided by new or reconnected hosts. Indeed, if all the replication agents switch to the *move* mode, it becomes impossible to replicate and disseminate descriptors of new resources; as a consequence, agents cannot be allowed to live forever, and must gradually be replaced by new agents that set off in the *copy* mode. The second issue is that the system must remove "obsolete descriptors", i.e. descriptors of resources provided by hosts that have left the system, and therefore are no longer available.

Simple mechanisms are adopted to cope with these two issues. The first is to correlate the lifecycle of agents to the lifecycle of peers. When joining the Grid, a host generates a number of agents given by a discrete Gamma stochastic function, with average N_{gen}, and sets the life-time of these new agents to the average connection time of the peer itself. This setting assures that (i) the relation between the number of peers and the number of agents is maintained with time (more specifically, the overall number of agents is approximately equal to the number of active peers times N_{gen}) and (ii) a proper turnover of agents is achieved, which allows for the dissemination of descriptors of new resources, since new agents start in the *copy* mode. A second mechanism assures that, every time a peer disconnects from the Grid, it loses all the descriptors previously deposited by agents, thus contributing to the removal of obsolete descriptors. Finally, a *soft state* mechanism [16] is adopted to avoid the accumulation of obsolete descriptors in very stable nodes. Each host periodically refreshes the descriptors corresponding to the resources owned by other hosts, by contacting these hosts and retrieving from them updated information about resources.

It is worth mentioning that the described approach for handling a dynamic Grid implicitly manages any unexpected peer fault, because this occurrence is processed in exactly the same way as a peer disconnection. Indeed, the two events are indistinguishable, since (i) a peer does not have to perform any procedure before leaving the system, and (ii) in both cases (disconnection and fault) the descriptors that the peer has accumulated so far are removed.

3. PERFORMANCE EVALUATION

The performance of the Antares algorithm was evaluated with an event-based simulator written in Java. Simulation objects are used to emulate Grid peers and Antares 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 to be delivered to other objects.

A Grid network having a number of hosts N_p equal to 2500 is considered in this work. Hosts are linked through P2P interconnections, and each host is connected to 4 peer on average. The average connection time of a peer, T_{peer} (see Section 2.4), is set to 100,000 seconds. The number of Grid resources owned and published by a single peer is obtained with a Gamma stochastic function with an average value equal to 15 (see [12]).

Resources are characterized by metadata descriptors indexed by bit strings (keys) having 4 bits, with 2^4-1 possible values [3]. These values, as mentioned in Section 1, can result from a semantic description of a resource, in which each bit represents the presence of a particular topic, or from the application of a locality preserving hash function. In any case, it is guaranteed that similar keys are given to descriptors of similar resources. Notice that if the range of possible resource types is larger than 2^4-1, it can always be assumed that a resource is characterized by several attributes, each having no more than 2^4-1 possible values. In this case, each attribute corresponds to a separate mapping on the Grid network. This assumption is made in several P2P architectures ([1, 3]); however, here we discuss the simple case of one attribute descriptors.

The mean number of agents generated by a single peer, N_{gen}, is set to 0.5; as a consequence, the average number of agents N_a that travel the Grid is equal to $N_p/2$, as explained in Section 2.4. The average time T_{mov} between two successive agent movements is set to 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.

Prior to the numerical analysis, a graphical description of the behavior of the algorithm, for the case in which the number of bits in the descriptor key is set to 3, is given in Figure 2. For the sake of clearness we show a portion of the Grid, and Grid hosts are arranged in a bi-dimensional mesh. Each peer is visualized through a color that results from the application of the RGB color model. This color results from a combination of the 3 primary colors (red, green and blue), each of which is associated to one of the three bits of the descriptor (for example red is associated to the first bit and so on) and can assume a value ranging from 0 to 255. For a given peer, the value of each primary color is set by examining the corresponding bits of the descriptors maintained in this peer. More specifically, it is set to the fraction of bits equal to 1 with respect to the total number of descriptors, and then multiplied by 255. This way the color of each peer immediately represents the descriptors that this peer maintains. Three snapshots of the network are depicted: the first is taken when the Antares process is initiated (time 0), the second is taken 50,000 seconds later, and the third snapshot is taken in a quite steady situation, 500,000 seconds after the process start. This figure shows that descriptors are initially distributed in a completely random fashion, but subsequently they are reorganized and spatially sorted by agents. In fact we note the creation of color spots that reveal the accumulation of similar descriptors in restricted regions of the Grid. The different colors of the spots correspond to the different combinations of descriptor values that are aggregated. For example red, green and blue spots reveal a massive presence of descriptor keys equal to [1,0,0], [0,1,0], [0,0,1], respectively, while yellow, cyan, violet and white spots represent the predominant presence of vectors having two or three bits equal to 1. We can also observe that colors change gradually between neighbor regions, which proves that the descriptors are not only clustered but also spatially *sorted* on the network.

A set of performance indices are defined to evaluate the performance of the Antares algorithm. The overall homogeneity function H, discussed in Section 2.3, is used to estimate the effectiveness of the algorithm in the reorganization of descriptors. The N_d index is defined as the mean number of descriptors maintained by a Grid host. Since new descriptors are only generated by agents that work in the *copy* mode, the number of such agents, N_{copy}, is another interesting index that helps understand what happens in the system. Finally, the processing load, L, is defined as the average number of agents per second that get to a Grid host, and there perform pick and drop operations.

Performance indices have been obtained by varying several parameters, for example the average number of resources published by a host and the frequency of agent movements. We found out that the qualitative behavior of Antares is not affected by these parameters, which proves the robustness of the algorithm. This robustness derives from the decentral-

[3]a key string with all bits equal to 0 is not permitted in order to correctly calculate the cosine of the angle between two vectors.

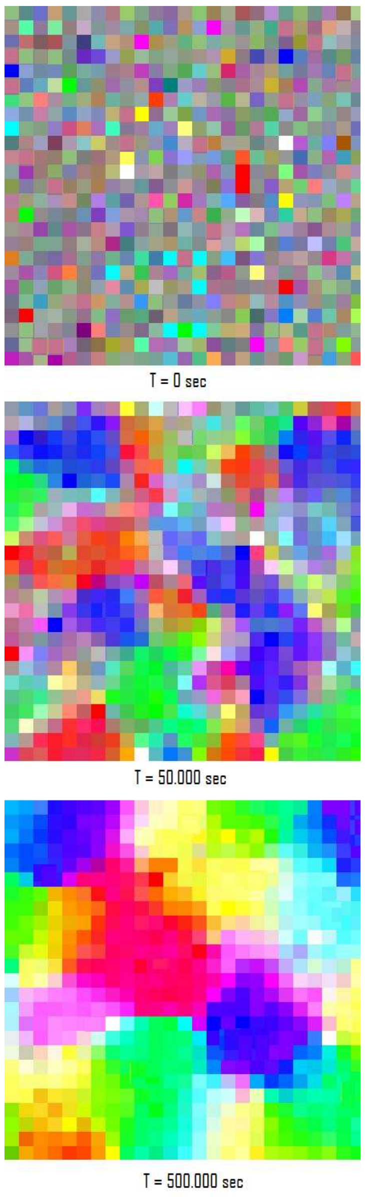

Figure 2: Accumulation and reorganization of resource descriptors.

ized, self-organizing and adaptive features of the algorithm, which are also the cause of its scalability. Indeed, since each agent operates only on the base of local information, performance is not significantly affected by the topology or the size of the network.

Here we choose to show performance indices, versus time, obtained for different values of the parameter T_{peer}, the average connection time of a peer, which has been introduced in Section 2.4. Specifically, tested values of T_{peer} range from 36,000 to 500,000 seconds and, for comparison purposes, we also tested the case in which peers never disconnect. This kind of analysis is useful because it helps understand the mechanisms through which the information system is constructed, and also because it is possible to assess the algorithm ability to adapt the mapping of descriptors to the continuous modifications of the environment. Simulations were executed with T_h equal to 9.0, which means that each agent sets off in the *copy* mode and passes to the *move* mode as soon as its pheromone, starting from 0, exceeds the threshold of 9.0 (see Section 2.3).

Figure 3 reports the trend of H, the overall homogeneity function. It appears that the work of Antares agents make this index increase from about 0.50 to much higher values. After a transient phase, the value of H becomes stable: it means that the system reaches an equilibrium state despite the fact that peers go down and reconnect, agents die and others are generated, etcetera. In other words, the algorithm adapts to the varying conditions of the network and is robust with respect to them. Note that the stable value of H decreases as the network becomes more dynamic (that is, with lower values of T_{peer}), because the reorganization of descriptors performed by agents is partly hindered by environment modifications. However, even with the lowest value of T_{peer} that we tested, 36,000, the percentage increase of H is about 30%, whereas it is more than 50% with T_{peer} equal to 500,000.

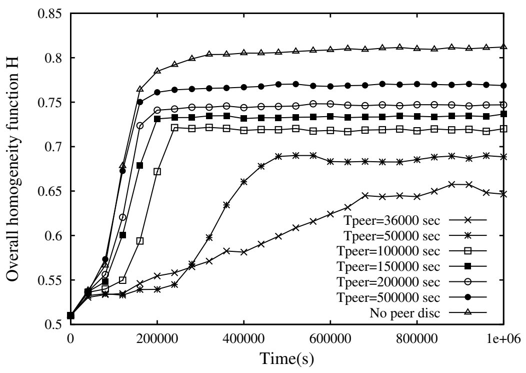

Figure 3: Overall homogeneity function, vs. time, for different values of the average connection time T_{peer}.

Figure 4 depicts N_{copy}, the number of agents that operate in the *copy* mode, also called *copy agents* in the following. This analysis is interesting because *copy* agents are responsible for the replication of descriptors, whereas agents in the *move* mode are exclusively devoted to the relocation and spatial sorting of descriptors.

When the process is initiated, all the 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_h. This corresponds to the sudden drop of curves that can be observed in the left part of Figure 4. Thereafter the number of *copy* agents gets stabilized, even with some fluctuations; this equilibrium is reached because the number of new agents which are generated by Grid hosts (these agents always set off in the *copy* mode) and the number of agents that switch from *copy* to *move* get balanced.

Figure 4 also shows that the number of *copy* agents increases as the Grid is more dynamic. Indeed, a higher turnover of agents is obtained when peers disconnect and reconnect with a higher frequency, because more agents die if more peers disconnect (because the lifetime of agents is correlated to the lifetime of peers), and at the same time more agents are generated by reconnecting peers (see Section 2.4). Since new agents set off in the *copy* mode, this leads to a larger number of *copy* agents, as appears in Figure 4. Moreover, note that in a stable network (no peer disconnections) all agents work in the *move* mode after a short transient phase. Indeed in this case no new agents are generated after the process is initiated.

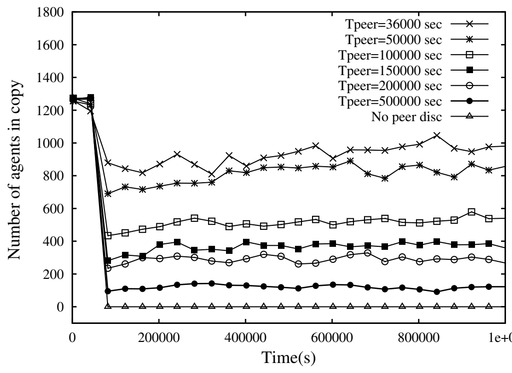

Figure 4: Number of agents that operate in the *copy* mode, vs. time, for different values of the average connection time T_{peer}.

Figure 5 reports N_d, the average number of descriptors that are maintained by a Grid host at a given time. One of the main objectives of Antares is the replication and dissemination of descriptors. This objective is achieved because the value of N_d increases from an initial value of about 15 (equal to the average number of resources published by a host) to much higher values; as for the other indices, the trend of N_d undergoes a transient phase, then it gets stabilized, even if with some fluctuations.

The value of N_d is determined by two main phenomena: on the one hand, a large number of *copy* agents tend to increase N_d, because they are responsible for the generation of new descriptor replicas. On the other hand, a frequent disconnection of peers tends to lower N_d, because a disconnecting peer loses all the descriptors that it has accumulated so far (see Section 2.4). These two phenomena work in opposite directions as the value of T_{peer} increases: in a more dynamic network there are more *copy* agents (which tends to increase N_d), but more descriptors are thrown away by disconnecting peers (which tends to decrease N_d).

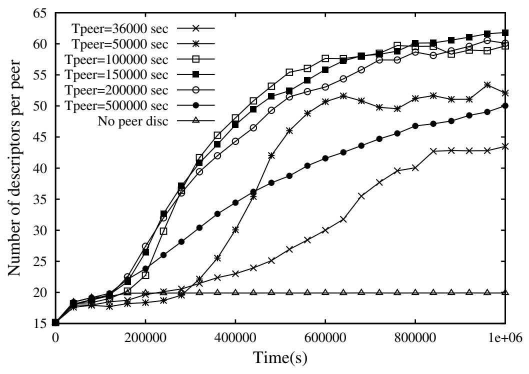

Figure 5: Average number of descriptors maintained by a Grid host, vs. time, for different values of the average connection time T_{peer}.

The result is that the stable number of N_d is relatively lower both when the disconnection frequency is very low and when it is very high or infinite (i.e., with no peer disconnections). Interestingly, a higher degree of replication can be reached for intermediate values of T_{peer}, which are more realistic on Grids. Indeed the figure shows, even if curves are more wrinkled than those examined so far (probably due to the two underlying and contrasting mechanisms discussed above), that the value of N_d first increases as T_{peer} increases from 36,000 to values comprised between 100,000 and 150,000, then it decreases again for higher values of T_{peer}, and finally it becomes very low in the (not realistic) case of no peer disconnections.

As opposed to the indices described so far, the processing load L, defined as the average number of agents per second that are processed by a peer, does not depend on the value of T_{peer}, but only on the number of agents and the frequency of their movements across the Grid. L can be obtained as follows:

$$L = \frac{N_a}{N_p \cdot T_{mov}} = \frac{N_{gen}}{T_{mov}} \tag{5}$$

In the described scenario, since the average value of T_{mov} is equal to 60 seconds, and N_{gen} is set to 0.5, each peer receives and processes about one agent every 120 seconds, which can be considered an acceptable load. Note that the processing load does not even depend on other system parameters such as the network size, the average number of resources published by a node and so on, which confirms the scalability properties of the Antares algorithm.

In the Introduction of the paper, we briefly mentioned that the reorganization and sorting of descriptors can be exploited by a semi-informed discovery algorithm. According to this algorithm, query messages travel the network randomly until they get close to a region of interest, and then they may be directly driven to peers which have a large number of useful descriptors. As an example, one can examine the last graph of Figure 2: if a discovery operation is issued to search for "red" descriptors (i.e., descriptors having keys equal to [1,0,0]), a *blind* search can become *informed* if a query message gets to a peer whose color is sufficiently similar to red. From that point, the query can be driven towards "red" peers. This can be simply done by making this query hop to the neighbor peer whose descriptors are the most similar to the target descriptor specified in the query.

Previous work [8, 9] showed that the performance of discovery operations is strictly related to a better reorganization of information, and preliminary experiments are fully confirming this observation also in the context of Antares.

Finally, the spatial sorting of descriptors enables the possibility of effectively serving *range queries*, since descriptors indexed by similar keys are likely to be located in neighbor hosts. The confirmation of this intuition is one of the main objectives of current work.

4. CONCLUSIONS

In this paper we introduced and evaluated Antares, an algorithm inspired on ant behavior whose aim is to build a P2P information system of a Grid. Through the evaluation of simple probability functions (*pick* and *drop*), a number of ant-inspired agents replicate and move the descriptors of Grid resources from host to host, and this way disseminate and reorganize these descriptors on the network.

Antares achieves an effective reorganization of information, since descriptors are spatially sorted on the network and, in particular, descriptors indexed by equal or similar binary keys are placed in neighbor Grid hosts. This was confirmed in the paper both with a graphical description based on the RGB model and with the analysis of performance measures, in particular of a homogeneity index based on the cosine similarity between binary vectors.

The resulting P2P information system is basically *unstructured* because there is no predetermined association between resources and hosts, but thanks to the work of agents, the spatial sorting of descriptors allows to retain important benefits of *structured* P2P systems, such as the possibility of driving discovery messages towards useful descriptors and effectively managing range queries.

Antares is scalable and robust with respect to the variation of algorithm and network parameters. In particular, the reorganization of descriptors performed by Antares spontaneously adapts to the ever changing environment, for example to the joins and departs of Grid hosts and to the changing characteristics of resources. This results from the decentralized, self-organizing and adaptive features of Antares, which are borrowed by the corresponding biological system.

5. ACKNOWLEDGMENTS

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265). This work has also been supported by the Italian MIUR FAR SFIDA project on interoperability in innovative c-businnes models for SMEs through an enabling Grid platform.

6. REFERENCES

[1] A. Andrzejak and Z. 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.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems.* Oxford University Press, New York, NY, USA, 1999.

[3] M. Cai, M. Frank, J. Chen, and P. Szekely. Maan: A multi-attribute addressable network for grid information services. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, page 184, Washington, DC, USA, 2003. IEEE Computer Society.

[4] S. Camazine, N. R. Franks, J. Sneyd, E. Bonabeau, J.-L. Deneubourg, and G. Theraula. *Self-Organization in Biological Systems.* Princeton University Press, Princeton, NJ, USA, 2001.

[5] A. Crespo and H. 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.

[6] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Compututer Systems*, 16(9):851–871, 2000.

[7] A. Forestiero, C. Mastroianni, and G. Spezzano. Construction of a peer-to-peer information system in grids. In H. Czap, R. Unland, C. Branki, and H. 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.

[8] A. Forestiero, C. Mastroianni, and G. Spezzano. An agent based semi-informed protocol for resource discovery in grids. In *Proc. of the International Conference on Computational Science(4) ICCS'06*, pages 1047–1054, 2006.

[9] A. Forestiero, C. Mastroianni, and G. Spezzano. Building a peer-to-peer information system in grids via self-organizing agents. *Journal of Grid Computing*, on line first article, Springer Netherlands, 2007.

[10] A. Forestiero, C. Mastroianni, and G. Spezzano. Qos-based dissemination of content in grids. *Future Generation Computer Systems*, on line first article, Elsevier Science, 2007.

[11] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[12] A. Iamnitchi, I. Foster, J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski. A peer-to-peer approach to resource location in grid environments. In *Grid Resource Management.* Kluwer Publishing, 2003.

[13] E. D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In *Proc. of SAB94, 3rd international conference on Simulation of adaptive behavior: from animals to animats 3*, pages 501–508, Cambridge, MA, USA, 1994. MIT Press.

[14] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Design and implementation tradeoffs for wide-area resource discovery. In *Proc. of the 14th IEEE International Symposium on High Performance Distributed Computing HPDC 2005*, Research Triangle Park, NC, USA, July 2005.

[15] C. Platzer and S. Dustdar. A vector space search engine forweb services. In *ECOWS '05: Proceedings of the Third European Conference on Web Services*, page 62, Washington, DC, USA, 2005. IEEE Computer Society.

[16] P. Sharma, D. Estrin, S. Floyd, and V. 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.

[17] TheGlobusAlliance. The web services resource framework, 2006. http://www.globus.org/wsrf/.

[18] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi. Peer-to-peer resource discovery in grids: Models and systems. *Future Generation Computer Systems*, 23(7):864–878, August 2007.