ARiA: A Protocol for Dynamic Fully Distributed Grid Meta-Scheduling

Amos Brocco, Apostolos Malatras, Ye Huang, Béat Hirsbrunner
Department of Informatics
University of Fribourg, Switzerland
Email: {amos.brocco, apostolos.malatras, ye.huang, beat.hirsbrunner}@unifr.ch

***Abstract*—Critical to the successful deployment of grid systems is their ability to guarantee efficient meta-scheduling, namely optimal allocation of jobs across a pool of sites with diverse local scheduling policies. The centralized nature of current meta-scheduling solutions is not well suited for the envisioned increasing scale and dynamicity of next-generation grids, the success of which relies on the development of fully distributed, flexible and autonomic systems tailored to very large sets of highly volatile and heterogeneous resources. In this respect, we introduce a fully distributed grid meta-scheduling scheme that effectively addresses the concerns regarding the scalability and adaptability of future grid systems. Our approach employs a lightweight protocol, called ARiA, that is based on peer-to-peer communication between grid nodes, and makes use of dynamic rescheduling to consider and adapt to changes in the availability of resources. Extensive evaluation by means of an in depth simulation study highlighted the effectiveness of the proposed solution in improving the overall performance in terms of job completion time and load-balancing.**

I. Introduction

The original vision for grid computing was spurred by the increasing demand for solutions enabling large-scale resource sharing across geographically dispersed institutions [1]. From this perspective, grid computing aims at lowering the barriers of entry by facilitating the contribution of computing resources and their exploitation, in the same spirit as the web supports worldwide information sharing. Grid systems are the composition of resources, middleware and protocols belonging to distinct entities, referred to as *virtual organizations* [2], promoting collaborative task execution and problem solving. Grids have been successfully deployed in scientific scenarios [3], and have attracted a noteworthy research stream aimed at improving the underlying infrastructures in terms of accessibility [4], efficiency [5] and reliability [6].

Because of the inherent heterogeneous and distributed nature of grid systems, several challenges arise in their deployment. Firstly, when users submit jobs (or tasks) to the grid, nodes with resources matching the requested profile must be located in an efficient way by means of resource discovery mechanisms. Moreover, grid scheduling and allocation strategies must ensure that the job is executed according to users' demands (i.e. QoS agreements such as response time, cost, etc.) and resource providers' requirements (i.e. security policies, execution efficiency, resource utilization, etc.). Scheduling on the grid is further complicated by the fact that it needs to consider two operational levels: *local-scheduling* and *meta-scheduling*. Local-scheduling is concerned with managing local tasks' execution policies and resources on every computing node, whereas meta-scheduling provides coordination and orchestration between different local schedulers by assigning tasks to the appropriate computing nodes, typically within a virtual organization. Currently deployed grid infrastructures [7], [8] rely on centralized or hierarchical schemes to support resource discovery, resource and data management, meta-scheduling, as well as security services. While such an organizational model is justified by the business requirements imposed by virtual organizations, there is nonetheless a concrete demand for flexible, autonomic, and self-manageable grids, in order to reduce deployment costs, increase reliability, and meet dynamic users' needs [9].

Several network applications have already recognized and exploited the advantages of distributed and decentralized approaches, which are sustained by the advances of the underlying network technologies (i.e. ubiquity, link bandwidth, etc.). These distributed approaches overcome some of the limitations of their centralized counterparts, by improving scalability and robustness while being responsive to dynamic conditions. On the downside, non-centralized architectures raise a number of challenges such as coordination and synchronization between entities. Particular examples of distributed designs that have been successfully deployed in large scale networks include content distribution (e.g. BitTorrent), VoIP communication platforms (e.g. Skype), and distributed data storage and retrieval (e.g. Freenet). This established shift is also reflected within grid architectures, with the emergence of decentralized resource discovery mechanisms [10], [11], fully distributed load-balancing solutions [12], and decentralized meta-scheduling algorithms [13].

This paper focuses on grid task meta-scheduling, by presenting a fully distributed protocol named ARiA to achieve efficient global dynamic scheduling across multiple sites. The meta-scheduling process is performed *online*, and takes into account the availability of new resources as well as changes in actual allocation policies. Accordingly, the proposed approach aims at addressing the scalability and adaptability of grids, to optimally exploit dynamically changing grid resources. Scalability concerns both the size of the grid and the actual load. On one side new grid nodes must seamlessly merge into the grid system; on the other side, jobs must be distributed over all suitable nodes to avoid hot spots, as long as requirements

are met. We refer to adaptability as the ability of scheduling and rescheduling tasks according to global or local scheduling policy changes. In this respect, a balance between adaptability and stability is required to avoid coupling situations that have an adverse effect on performance.

The remaining of this paper is organized as follows: Section II discusses related work regarding meta-scheduling with a focus on decentralized approaches. Section III presents the proposed dynamic scheduling protocol, while Section IV and V present the considered evaluation scenarios, respectively the obtained results. Finally Section VI summarizes our conclusions on this work and provides some insights on future work.

II. Related Work

In order to meet the expectations of large-scale distributed computing, grid systems should be able to manage large sets of heterogeneous resources, and perform optimal task allocation on these resources by scheduling jobs on the most suitable machines while avoiding to repeatedly overload the most capable ones. To a large extent, these goals rely on the capabilities and performance of the scheduling mechanism. Accordingly, our research focuses on a decentralized scheduling mechanism, the aim of which is twofold, namely enabling fully distributed meta-scheduling across heterogeneous nodes, while additionally providing dynamic load-balancing support by rescheduling jobs across nodes whenever possible. Subsequently we review here related work concerning both the meta-scheduling and the load-balancing issues, by first briefly discussing centralized approaches and then shifting the focus on decentralized ones.

Traditional grid models [7], [8] rely on centralized or hierarchical meta-schedulers that have a global view of the resources shared on the grid or by their virtual organization. Research has come up with very efficient centralized meta-scheduling mechanisms [14] that can take full advantage of a global view of the grid and provide optimal allocation of tasks on resources. It should be noted that centralized scheduling does not necessarily require a corresponding central information repository, but can rely on distributed information systems [15]. Nonetheless, these approaches still contain bottlenecks for scalability of the system, as well as single points of failure that may affect the robustness of the grid as a whole.

While fully decentralized cooperative grid solutions bear advantages over their centralized counterparts, interoperability of the diverse systems involved is often hindered by infrastructural or organizational problems, such as lack of standardization [16]. As discussed in [17], to alleviate these issues, collaborative scheduling solutions should avoid enforcing control over local resources by establishing a clear separation between global and local resource management. Moreover, resource management and scheduling should rely on adaptive decision-making in order to cope with unprecedented situations.

The design of decentralized and adaptive scheduling algorithms is considered in [18], with nodes performing load-balancing within a limited set of neighbors. Two strategies are proposed, namely transferring jobs at precise intervals or depending on their arrival time; both strategies have the goal of achieving similar total execution time on all nodes. In the direction of reducing the average response time, [16] proposes an adaptive decentralized mechanism that employs an evolutionary fuzzy algorithm to select the best site for job delegation among the set of all possible candidates.

The Organic Grid [19] introduces a novel paradigm that redefines the grid as self-organized biologically inspired complex system of agents providing decentralized scheduling for heterogeneous tasks on a large number of resources. Nodes are organized as a tree, with the root being the job originating node, and faster nodes located closer to it; nodes can push tasks down the tree depending on the actual load of their children.

Collective intelligent behavior of mobile agents has been also exploited in [12] to support grid task load-balancing in a fully distributed environment. Job requirements and resources are profiled using a performance analysis tool called PACE [20], and matched to appropriate resources by the agents. Recognizing the importance of decentralization and self-organization for the future of grid systems, [21] presents a distributed grid scheduling framework where nodes group into communities according to resource similarities and disseminate their actual state. The scheduling process is decentralized and makes use of information about remote nodes in order to find the best resources to fulfill a request.

The distributed meta-scheduling model presented in [13] operates on the principle of submitting a job to the least loaded sites and subsequently revoking it on all but the one that has commenced its execution. An evident drawback of this model is the overloading of a large number of schedulers with jobs that are frequently cancelled. Another decentralized scheduling and load-balancing technique is detailed in [22], which depends on nodes retaining jobs or submitting them to their neighbors according to a heuristic on local load. A different approach is taken in [23], where the selection of a target neighbor for job delegation is driven by the available bandwidth; this is made possible by the adoption of a simplistic model that considers all tasks as identical and focuses on the time required to transfer data.

The potential of applying peer-to-peer technologies to support decentralized grid scheduling is highlighted in [24], with a fully distributed solution where nodes perform a gossip-based exploration of the network for the purpose of generating an optimal schedule on the discovered resources. Peer-to-peer gossiping protocols are also employed in [25], but with the goal of disseminating the state of the available resources across the grid; this information is cached by remote nodes and used to optimally allocate incoming jobs.

The GridIS [26] scheduling algorithm employs a peer-to-peer communication model that enables resource providers to bid for the delegation of a job. Job requests are submitted to the grid through a portal that broadcasts them in an unstructured peer-to-peer overlay network. The objective of GridIS is to satisfy both resource consumers and providers, by ensuring high successful execution rates, respectively fair allocation of

benefits. Similarly to GridIS, the work presented in [27] makes use of a structured peer-to-peer overlay network to discover nodes wishing to carry out a job; furthermore, rescheduling is exploited to avoid starvation of jobs failing to be executed.

The protocol proposed in this paper differs from the aforementioned research approaches in that it supports fully distributed task meta-scheduling across heterogeneous resources. All meta-scheduling decisions are made locally, based on task-related information and the node's scheduling policy, without requiring detailed scheduling information from other nodes. Our solution promotes peer-to-peer interaction between nodes and self-organization, thus contributing to the previously mentioned drive towards flexible and autonomic grids.

III. ARiA PROTOCOL

The ARiA protocol aims at providing fully distributed task meta-scheduling across a heterogeneous grid. The name ARiA (*air* in Italian, denoting the aim to be lightweight) comes from the initials of the different message types defined in our protocol, namely REQUEST, ACCEPT, INFORM, and ASSIGN (Table I). This section details the operational phases of the protocol, as well as the information exchanged between nodes by means of messages.

A. Assumptions

The fundamental design principle of the protocol is that it does not enforce any particular local scheduling policy. We assume that all nodes are connected through some sort of peer-to-peer overlay network enabling communication between any pair of nodes (for evaluation our previous work on a self-organized overlay [28] was exploited).

The underlying idea is to make use of the time-to-execution to perform dynamic rescheduling of jobs across grid nodes, thus achieving better global throughput and load-balancing. Job delegation may occur to any node whose resource profile matches the job requirements, while nodes are not allowed to decline incoming jobs that have been already accepted. Jobs are assumed to be independent, and while every node may hold several jobs within its scheduling queue, only one job at a time can be executed. Furthermore, to avoid checkpointing issues, preemption and migration of running jobs are not considered.

The protocol itself does not specify neither the resource profiles and job submission formats, nor the matching logic determining whether a task can be executed on a specific node (because of resource needs, security, or accounting policies). Actual implementations may choose to use one of the available job description schemas such as JSDL [29]. Finally, execution of tasks and transmission of task-related data between nodes are out of the scope of this research.

B. Job Submission Phase

The first phase of the protocol covers the submission of jobs and their initial handling by the node that each job was submitted to. Because the protocol aims at achieving optimal grid-level meta-scheduling, submitting a job to a particular node does not ensure that execution will take place locally, unless such a requirement is specified in the job description.

TABLE I
PROTOCOL MESSAGES AND FIELDS

ACCEPT	Node's address	Job UUID	Cost	
REQUEST	Initiator's address	Job UUID	Job Profile	
INFORM	Assignee's address	Job UUID	Job Profile	Cost
ASSIGN	Initiator's address	Job UUID	Job Profile	

Jobs are assigned a universal unique identifier (UUID) that enables their univocal tracking across the grid. Nodes receiving job submissions are referred to as *initiators* for these jobs. In order to find candidates for the execution of a job, initiators issue resource discovery queries across the grid peer-to-peer overlay by broadcasting REQUEST messages to a random subset of nodes of the overlay, and wait for a predefined timelapse for incoming query replies.

Besides the initiator's address and the job UUID, a REQUEST message contains the profile of the resources required to carry out the job, which also specifies an Estimated job Running Time (ERT) according to a grid-level accepted baseline regarding computing power. The estimated running time can be computed by means of a job profiling middleware such as PACE [20]. Job profiles may also define additional job execution constraints, for example to prevent execution of a job outside the boundaries of a virtual organization.

C. Job Acceptance Phase

Upon reception of a REQUEST message, a node determines whether the requirements of the job profile match its own resources. If the request cannot be satisfied, the message is further forwarded on the peer-to-peer overlay, otherwise a *cost* value for the job based on actual resources and current scheduling is computed. The cost information is sent back to the job's initiator by means of an ACCEPT message.

The cost depends on the adopted local scheduling state with lower values being used to indicate better offers. The initiator evaluates incoming ACCEPT responses, and selects the best qualified node (i.e. the node providing the lowest cost). The job is delegated to the latter, which is referred to as the current *assignee*, by sending an ASSIGN message.

Currently, two cost functions have been considered, namely *Estimated Time To Completion* (ETTC) and *Negative Accumulated Lateness* (NAL) for batch, respectively deadline schedulers. As we assume that deadline scheduling offers are not mixed with batch ones, values produced by different functions do not necessarily need to be comparable.

The *Estimated Time To Completion* function defines the cost for a job j as:

$$ETTC_{cost}(j) = ETTC_j$$

where $ETTC_j$ corresponds to the relative time that the job is expected to finish according to the local scheduling policy and actual load of the node (determined by the scheduling queue).

The *Negative Accumulated Lateness* is targeted at deadline scheduling algorithms, and for a job j and an existing local scheduling queue Q it is computed as follows:

$$NAL_{cost}(j) = \sum_{job \in Q'} \delta_{(job,Q')} * |\gamma_{job}|$$

with

$$Q' = Q \cup \{j\}$$

$$\gamma_{job} = deadline_{job} - ETC_{job}$$

$$\delta_{(job,S)} = \begin{cases} -1 & \gamma_{job} \geq 0, \forall\ job \in S, \\ 0 & \gamma_{job} \geq 0 \wedge \exists\ w \in S\ :\ \gamma_w < 0, \\ 1 & \text{otherwise} \end{cases}$$

ETC_{job} refers to the absolute time that the job is expected to finish according to the local scheduling policy and the actual load of the node (determined by the scheduling queue Q'), while $deadline_{job}$ is the upper time limit for job completion; hence γ_{job} represents the lateness of the job.

D. Dynamic Rescheduling Phase

An important aspect of the ARiA protocol is the dynamic rescheduling of jobs. This supports the scalability and adaptability of the meta-scheduling mechanism by enabling job re-allocation to reflect possible changes in the state and availability of resources. This can typically be the result of new nodes connecting to the grid, or existing jobs terminating earlier than predicted or being cancelled.

While, at the time of job assignments, *assignees* represent the initiators' perceived optimal solutions for job execution, it should be expected that better alternatives may potentially arise in the future. Accordingly, the *assignee* attempts to find candidates for rescheduling of jobs in its queue while their execution has not yet started. For this purpose, INFORM messages are disseminated across the network using a low-overhead selective flooding protocol [28].

The structure of INFORM messages relates to that of REQUEST messages, in that they both contain a full description of the job's profile. The goal of INFORM messages is to discover nodes that might carry out the execution of the job at a lower cost than the current *assignee*. For this reason, INFORM messages also carry the actual cost value, as computed by the aforementioned cost calculation functions. Nodes will typically generate INFORM messages for a set of jobs in their queue according to a selection mechanism. For batch schedulers jobs with the largest waiting times are preferentially selected, whereas for deadline schedulers jobs with the least lateness are chosen.

The behavior of a node upon reception of INFORM messages is similar to the one concerning REQUEST messages, with the node first checking whether it can satisfy the job's requirements and then evaluating the corresponding cost for execution. Unlike REQUEST messages, an ACCEPT reply will only be sent to the current *assignee* if a lower cost can be guaranteed. Thresholds may be introduced to prevent rescheduling when the benefit does not justify the additional overhead, for example if the execution time is only reduced by a small fraction.

The rescheduling process is completed when the current *assignee* receives the ACCEPT message and accordingly reassigns the job to the new *assignee* by means of an ASSIGN message. To ease tracking of jobs, and enable failsafe mechanisms in the event of an *assignee*'s crash, rescheduling actions may be notified to the job's initiator.

IV. Evaluation

An in depth study by means of simulation was performed to evaluate the behavior of the ARiA protocol in a heterogeneous grid environment. To this extent, several scenarios were considered, taking into account the multi-faceted nature of the grid meta-scheduling problem. More specifically, we focused on the efficiency of the proposed dynamic rescheduling mechanism, its scalability and adaptability, the generated traffic overhead, as well as its ability to successfully address load-balancing. Moreover, we aim at providing a sensitivity analysis of parameters of the protocol, in order to understand their influence on the aforementioned assessment metrics. This section introduces the evaluation settings and the details of the considered scenarios.

A. Overlay Network

The overlay network is connected by means of a fully distributed algorithm named BLåTANT-S, a full analysis of which can be found in [28]. BLåTANT-S employs collaborative bio-inspired swarm intelligence techniques [30] to maintain an overlay network with bounded average path length and minimal number of links by arranging the topology in a self-organized way. The autonomic behavior of different species of ant-like agents, which are exchanged between nodes of the network, contributes to the optimization of the overlay topology: new logical links are added if required to reduce the diameter, while existing links that do not contribute to the solution are removed.

For the evaluation of ARiA, an overlay of 500 nodes with a target average path length of 9 hops was deployed in a custom simulator reproducing realistic round-trip delays. The average node's degree attained during simulations was 4, resulting in about 2000 overlay links. Aside from this baseline scenario, we also experimented with an expanding overlay growing up to a size of 700 nodes with an average of 2800 links.

B. Grid Resources

In line with the underlying assumptions of our protocol, meta-scheduling takes places over a heterogeneous pool of

resources, therefore each grid node is characterized by a different profile. Profiles are comprised of several fields that describe both hardware and software properties of the machine. In particular, we consider the implemented architecture (e.g. AMD64, POWER, etc.), available memory, available disk space, and operating system (e.g. LINUX, SOLARIS, etc.). Upon initialization, the simulator randomly assigns a profile to each node. Values for each field are chosen with different probability distributions, as follows:

- **Architectures** are chosen according to the list published on the *TOP500 Supercomputing Sites* (www.top500.org) at the time of the writing of this paper. The probability distribution is as follows: AMD64 87.2%, POWER 11%, IA-64 1.2%, SPARC 0.2%, MIPS 0.2%, NEC 0.2%;
- **Available Memory and Disk Space** are both independently and uniformly chosen as either 1, 2, 4, 8, or 16 Gigabytes;
- **Operating Systems** installed on each node are based on the aforementioned *TOP500* list, with the following distribution: LINUX 88.6%, SOLARIS 5.8%, UNIX 4.4%, WINDOWS 1%, BSD 0.2%.

Moreover, every node has an associated *performance index* that compares its computing power to the baseline hardware used to calculate the Estimated job Running Time (ERT). Each node is assigned a random *performance index* p between 1 and 2, which the simulator uses to calculate the Estimated job Running Time on that particular node (that will be referred to as ERT^p). Accordingly, the ERT^p is defined as the ERT divided by the performance index p.

C. Local Scheduling Policies

One of the primary goals of the proposed protocol is to be *local scheduler agnostic*. Different schedulers are randomly assigned to each node. In this respect, the following scheduling policies have been considered:

- **First-Come-First-Served (FCFS)**: incoming jobs are appended to the scheduling queue according to the local arrival time (i.e. reception of an ASSIGN message);
- **Shortest-Job-First (SJF)**: the scheduling order depends on the jobs' ERT, with shorter jobs being executed first;
- **Earliest-Deadline-First (EDF)**: used only for deadline scheduling, this policy prioritizes jobs with an earlier deadline (as specified in their profile).

FCFS and SJF share the same cost function (as defined in Section III-C), and are thus interoperable, while EDF will be the sole scheduling policy considered in deadline scenarios.

D. Jobs

User submitted jobs are simulated by means of a random generator. Jobs are submitted to random nodes which initiate the protocol by sending REQUEST messages on the network. Each job is characterized by parameters defining the resources required to execute the job. This information is matched against grid resources profiles, and thus also includes the required architecture, memory, disk space, operating system. Job parameters are randomly chosen with the same probability distributions as for node profiles. Job descriptors also define an ERT, which is randomly assigned according to a normal distribution $\mathcal{N}(\mu, \sigma)$ with $\mu = 2h30m,\ \sigma = 1h15m$, using a lower bound of $1h$ and an upper bound of $4h$ to avoid extreme cases. For deadline scheduling, jobs' deadlines are set to an absolute time equal to the current time *plus* their ERT *plus* an additional random interval following the aforementioned distribution.

E. Scenarios

To evaluate the different aspects of the ARiA protocol, a series of 26 scenarios were devised and simulated. Each simulation run reproduces 41h 40m of grid activity, and a total of 10 runs was repeated for each scenario. For convenience, a list of the considered scenarios and a summary of their features can be found in Table II (as a naming convention, all scenarios with dynamic scheduling enabled are labelled starting with *i*).

In all scenarios a total of 1000 jobs is submitted to random nodes on the grid. Unless otherwise specified, jobs are submitted at 10 seconds intervals, starting from 20 minutes into the simulation, up to 3h 7m. Furthermore, when dynamic rescheduling is enabled, INFORM messages are sent for at most 2 scheduled jobs every 5 minutes. REQUEST messages are forwarded on the overlay for at most 9 hops; at each step, at most 4 random neighbors of the current node are contacted. Respectively, for INFORM messages a more lightweight approach is followed, with at most 8 hops and up to 2 neighbors. These values are based on the properties of the underlying peer-to-peer overlay management algorithm and the parameters set for its construction, and guarantee a near optimal operation without flooding the network.

A first set of scenarios (*FCFS*, *SJF*, *Mixed*, *iFCFS*, *iSJF*, *iMixed*) aims at assessing the impact of different local batch scheduling policies on the overall performance, and the benefits of dynamic rescheduling. In the *Mixed* and *iMixed* scenarios the ratio between FCFS and SJF schedulers is one-to-one; moreover, the *iMixed* scenario is selected as a baseline for comparison regarding subsequent evaluation of the protocol, as it best supports the concept of schedulers' heterogeneity. Additionally, the *Deadline* and *iDeadline* scenarios study the effects of dynamic rescheduling using a deadline local scheduling policy, namely EDF. In the latter scenarios, the deadline is set at an average of 7h 30m after the expected completion time, as explained in Section IV-D.

To evaluate the response of the scheduling mechanism in both low and high load situations, four scenarios were set up. To simulate low load situations (*LowLoad*, *iLowLoad*) the job submission rate was halved to one job every 20 seconds, running from 20 minutes to 5h 54m into the simulation. Respectively, for high load situations (*HighLoad*, *iHighLoad*) the rate was doubled, with one submission every 5 seconds, starting from 20 minutes up to 1h 45m. Correspondingly, the *DeadlineH* and *iDeadlineH* scenarios focus on the meta-scheduling protocol's ability to match stricter deadlines reduced to an average of 2h 30m after completion time.

TABLE II
SUMMARY OF EVALUATION SCENARIOS

Scenario	Description
FCFS	All nodes implement a FCFS batch scheduling policy without dynamic rescheduling.
SJF	All nodes implement a SJF scheduling policy without dynamic rescheduling.
Mixed	Nodes implement either a FCFS or a SJF policy (uniformly chosen at random) without dynamic rescheduling.
Deadline	All nodes implement the EDF scheduling policy.
LowLoad	Like *Mixed* but the rate of job submission is halved to 1 submission every 20 seconds, with a total of 1000 jobs between 20 minutes and 5h 54m into the simulation.
HighLoad	Like *Mixed* but the rate of job submission is doubled to 1 submission every 5 seconds, with a total of 1000 jobs between 20 minutes and 1h 45m into the simulation.
DeadlineH	Like *Deadline*, but jobs have deadlines closer to their estimated completion time (on average 2h 30m after completion instead of 7h 30m).
Expanding	Like *Mixed* but the network size is dynamically increased by connecting a new node on average every 50 seconds, starting from 1h 23m, up until a total of 700 nodes (at around 4h 10m into the simulation).
Precise	Like *Mixed* but with the actual job running time precisely matching the ERT.
Accuracy25	Like *Mixed* but the relative ERT error is $\pm 25\%$.
AccuracyBad	Like *Mixed* but the ERT is always lower than the actual running time.
iFCFS	Like *FCFS* but with dynamic rescheduling.
iSJF	Like *SJF* but with dynamic rescheduling.
iMixed	Like *Mixed* but with dynamic rescheduling.
iDeadline	Like *Deadline* but with dynamic rescheduling.
iLowLoad	Like *LowLoad* but with dynamic rescheduling.
iHighLoad	Like *HighLoad* but with dynamic rescheduling.
iDeadlineH	Like *DeadlineH* but with dynamic rescheduling.
iExpanding	Like *Expanding* but with dynamic rescheduling.
iInform1	Like *iMixed* but INFORM messages are sent only for 1 scheduled job every 5 minutes.
iInform4	Like *iMixed* but INFORM messages are sent for at most 4 scheduled jobs every 5 minutes.
iInform15m	Like *iMixed* but rescheduling is proposed only if a 15m improvement is provided.
iInform30m	Like *iMixed* but rescheduling is proposed only if a 30m improvement is provided.
iPrecise	Like *Precise* but with dynamic rescheduling.
iAccuracy25	Like *Accuracy25* but with dynamic rescheduling.
iAccuracyBad	Like *AccuracyBad* but with dynamic rescheduling.

The scalability of the proposed approach is gauged by means of a dynamically expanding network within the *Expanding* and *iExpanding* scenarios. Starting from the original network of 500 nodes, new nodes are added every 50 seconds starting from 1h 23m, increasing its size to 700 nodes at approximately 4h 10m into the simulation. These new nodes represent newly available grid resources that can take part in the scheduling and rescheduling process.

A sensitivity analysis of the dynamic scheduling mechanism is conducted by changing the number of candidate jobs and the rescheduling acceptance threshold. More specifically, in scenarios *iInform1* and *iInform4*, one scheduled job, respectively four, are considered for rescheduling, in contrast to the baseline choice of two. In the *iInform15m* and *iInform30m*, the benefit over the ETTC required for rescheduling is extended to 15 minutes, respectively 30 minutes, compared to the 3 minutes of the baseline scenario.

The meta-scheduling scheme depends on the ERT of each job, it is thus important to understand the influence of the accuracy of such an estimation. Accordingly, we evaluated the behavior of the protocol by varying the estimation error introduced in the simulation, which affects the Actual Running Time (ART). The ART for a job j (which is unknown until execution completes) on a node with performance index p is derived from ERT, ERT^p, and a relative error ϵ as follows:

$$ART_{j,\epsilon} = ERT_j^p + drift_{j,\epsilon}$$

with

$$drift_{j,\epsilon} = \mathcal{U}_{[-1,1]} * ERT_j * \epsilon$$

The baseline scenario assumes an accuracy of $\pm 10\%$ of the Estimated job Running Time ($\epsilon = 0.1$), whereas scenarios *Accuracy25* and *iAccuracy25* broaden the range to $\pm 25\%$ ($\epsilon = 0.25$). The *AccuracyBad* and *iAccuracyBad* scenarios consider an optimistic estimation where the ERT is always lower than the actual time ($\epsilon = 0.1$, and $drift_{j,\epsilon}$ is replaced with $|drift_{j,\epsilon}|$). Finally, the *Precise* and *iPrecise* scenarios refer to experiments where the estimation matches the Actual Running Time ($\epsilon = 0$).

V. RESULTS

Having detailed the parameters of the considered evaluation scenarios, we present and discuss here the corresponding results. First, a discussion on the benefits of the dynamic rescheduling mechanism of the ARiA protocol, its scalability, and its effectiveness to address the load-balancing problem is presented. This is followed by a sensitivity analysis of the different parameters of the protocol and their influence on the meta-scheduling performance. Finally, the generated network traffic is examined in order to determine the bandwidth consumption of the protocol.

A. Scheduling Policies

The first set of experiments aims at examining the effects of different local-scheduling policies on the overall efficiency of the protocol in terms of throughput, average job completion time, and load-balancing. In this respect, we compare results for the FCFS, SJF, and Mixed policies both with and without dynamic rescheduling.

Figure 1 shows the evolution of the number of completed jobs during the simulation. Vertical bars indicate the start and the end of the job submission process. The *iSJF* and *iMixed* scenarios demonstrate the benefits of dynamic rescheduling,

although it is noteworthy to highlight the comparative optimality of FCFS without the use of the proposed rescheduling mechanism. This is attributed to the fact that while the initial assignment is made to the node that provides the least time to completion in all scheduling algorithms, only FCFS preserves the optimality of the initial delegation by not modifying the scheduling order upon new submissions.

Figure 2 depicts the average job completion time, making a distinction between the waiting time and the execution time, where upon a significant observation can be made. Specifically, while dynamic rescheduling scenarios exhibit larger execution times, there is a reduction in the completion time in all but the optimal FCFS cases. These results prove the effectiveness of our approach as well as the usefulness of the rescheduling phase, and its ability to delegate jobs to nodes with shorter waiting queues rather than just to the most computationally capable ones.

The load-balancing effect of dynamic rescheduling is illustrated in Figure 3, which shows the overall utilization of grid nodes. Vertical bars indicate the start and the end of the job submission process. In both the *iSJF* and *iMixed* scenarios, the number of idle nodes (i.e. nodes with an empty scheduling queue) is reduced by approximately 100 nodes. Moreover, all dynamic rescheduling scenarios have very similar behavior as far as node utilization is concerned, which underlines the stability of our approach.

For deadline scheduling scenarios important performance metrics are the number of missed deadlines, the lateness (i.e. the time left from completion to the deadline), and the missed time (i.e. time past the deadline). As shown in Figure 4, dynamic rescheduling significantly reduces the occurrence of missed deadlines. In particular, their number decreased from an average of 187 in the *Deadline* scenario, to just 4 in the *iDeadline* one, respectively from 236 in the *DeadlineH* scenario, to 59 in *iDeadlineH*. Furthermore, while the average lateness (over successfully matched deadlines) was only slightly improved, the average missed time (over failed deadlines) was halved when employing dynamic rescheduling. These outcomes validate the benefits of the ARiA protocol even beyond simple batch scheduling.

B. Scalability

We assess here the scalability of our meta-scheduling protocol in respect to both the size of the network (i.e. number of available resources) and the frequency of job submissions. Concerning the first aspect, Figure 5 illustrates the number of idle nodes during our simulations in an expanding network. Vertical bars indicate the start and the end of the job submission process. The number of nodes was gradually increased from 500 to 700, in the interval between 1h 23m and 4h 10m. As expected, dynamic rescheduling enables better usage of the newly available resources, by reducing the number of idle nodes albeit network expansion.

Figure 6 presents the results concerning the number of idle nodes under diverse load conditions, where 1000 jobs are submitted within different time spans. Horizontal arrows mark

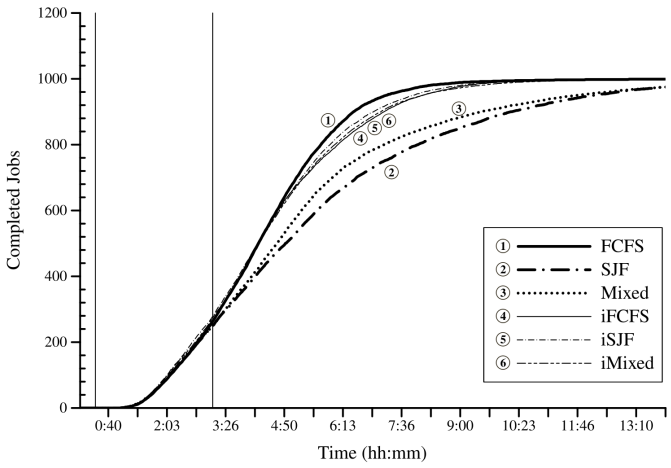

Fig. 1. Completed Jobs

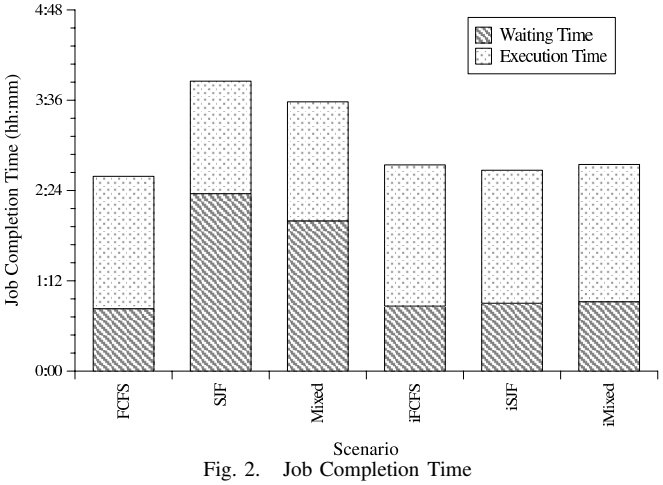

Fig. 2. Job Completion Time

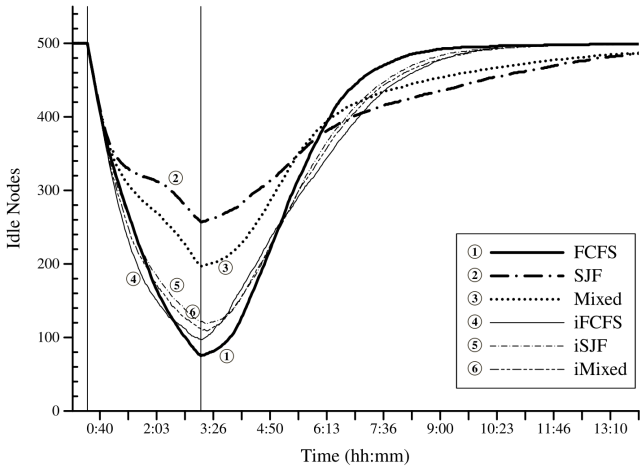

Fig. 3. Idle Nodes

the job submission intervals for the six considered scenarios. It is evident that in dynamic rescheduling scenarios it is possible to maintain a higher degree of resources' utilization than in their counterparts without rescheduling both in low and high

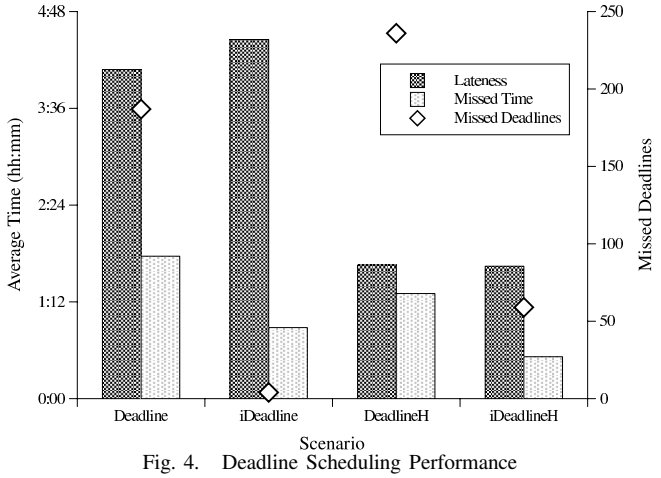

Fig. 4. Deadline Scheduling Performance

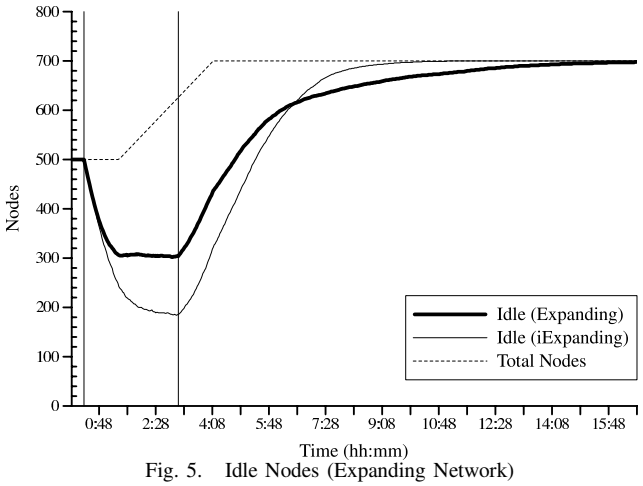

Fig. 5. Idle Nodes (Expanding Network)

load situations.

The merits of the increased jobs' load-balancing in dynamic rescheduling scenarios is reflected in the average job completion time, as illustrated in Figure 7. It interesting to notice that the performance in the *iHighLoad* scenario is comparable to the *LowLoad* one, despite the frequency of job submission having been increased by four times.

C. Rescheduling Policies

The dynamic rescheduling operation is determined by both the number of jobs that could potentially be reassigned, and by the benefit that could be provided by such an operation. In this respect, we are interested in evaluating the impact of different rescheduling policies on the average job completion time. In relation to the number of candidates for rescheduling, Figure 8 shows minimal differences between the *iInform1*, *iMixed*, and *iInform4*, with the latter achieving the lowest waiting time. Pertaining to the experiments aimed at comparing the diverse ETTC improvement thresholds required to trigger the rescheduling of a job, no particular variations in the overall performance were noticed.

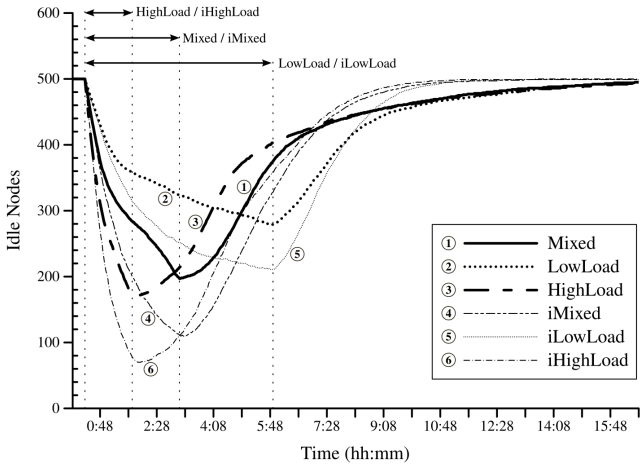

Fig. 6. Idle Nodes (Load)

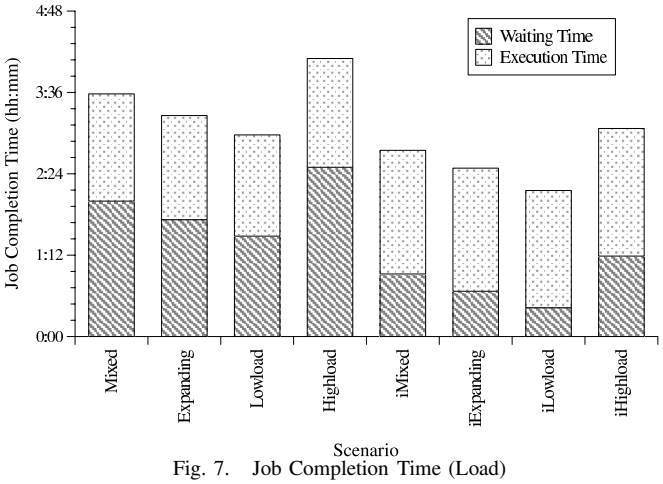

Fig. 7. Job Completion Time (Load)

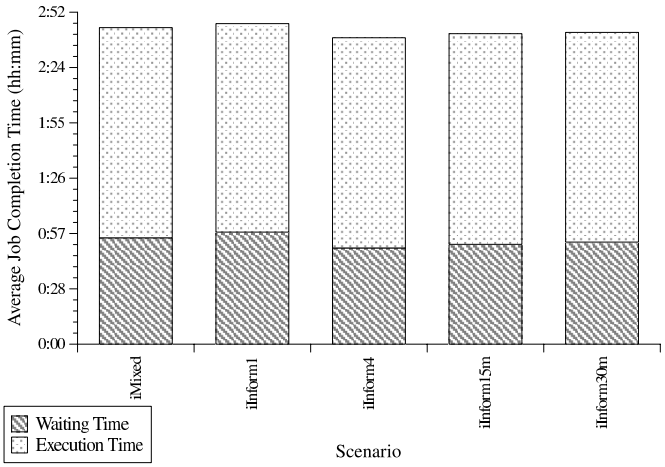

Fig. 8. Job Completion Time (Rescheduling Policies)

D. Sensitivity to ERT Accuracy

An important assumption of the ARiA protocol is the availability of a job running time estimation. We therefore experimented with different levels of accuracy to evaluate its

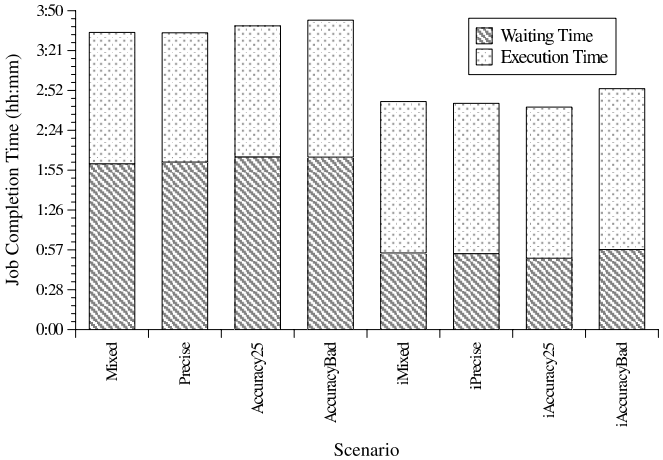

Fig. 9. Sensitivity to ERT

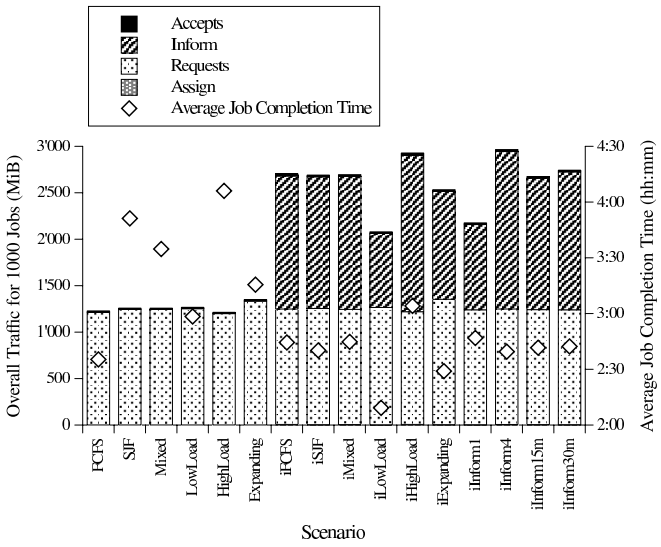

Fig. 10. Network Overhead Comparison

effects on the job completion time (Figure 9). In all but the *AccuracyBad* and *iAccuracyBad* scenarios, the balanced nature of the introduced error is to be accounted for the homogeneity of the produced results. Nonetheless, even optimistic estimations do not excessively worsen the efficiency of the system.

E. Traffic Evaluation

Figure 10 summarizes the generated network overhead for the most representative scenarios. Traffic estimations for the different types of messages employed by the protocol are detailed. Actual sizes for each message type were considered as follows: REQUEST, INFORM, and ASSIGN messages carry 1KBytes of information, whereas ACCEPT messages only 128bytes. Traffic generated by REQUEST messages represents the initial job allocation phase, and thus does not vary significantly across the different scenarios. Moreover, ASSIGN and ACCEPT messages account for a negligible part of the overall traffic.

The rescheduling phase generates additional traffic, which is nonetheless compensated by an improvement in the average job completion time. As this traffic estimation refers to a network of 500 nodes, the overhead accounts only for an average of 3MBytes per node in all scenarios over a period of about 42h, namely a bandwidth consumption of just 149bps. It is interesting to note that execution in an expanding network scenario (i.e. *iExpanding*) reduces the overall traffic generated by INFORM messages. The reason behind this is the ability of starting job execution earlier on newly available resources, hence reducing the number of candidate jobs for rescheduling. As far as dynamic rescheduling is concerned, scenario *iInform1* can be regarded as the best compromise between performance and network overhead, as it achieves an average completion time comparable to other scenarios while generating significantly less traffic.

VI. Conclusions

In this paper, we presented a fully distributed grid meta-scheduling protocol named ARiA that aims at improving the efficiency of heterogeneous grids, as well as addressing the related scalability and adaptability concerns. In this respect, our work is consistent with the vision of next-generation grids that strive to evolve into reliable, flexible, autonomic, and self-manageable systems that require minimal user intervention and reduced deployment costs.

The proposed scheme is based on simple messages exchanged between grid nodes over a peer-to-peer overlay, and does not impose any restriction on the implemented local scheduling policies, thus facilitating its integration with existing grid middlewares. The meta-scheduling process features a rescheduling phase that enables optimal job reallocation under dynamic conditions, by considering newly available resources as well as changes in current resource utilization.

An extensive experimental evaluation of the protocol's behavior yielded significant results that validate the effectiveness of our approach. More specifically, it was proved that shorter average execution times can be achieved under heterogeneous local scheduling policies. Additionally, dynamic rescheduling remarkably improved deadline scheduling performance, by reducing the number of missed deadlines. In terms of resource utilization, evaluation on both static and expanding network scenarios highlighted the ability of the dynamic rescheduling phase to enhance load-balancing amongst the nodes and to rapidly respond to the availability of new resources.

Traffic analysis of the ARiA protocol pinpointed an acceptable bandwidth consumption when compared to the acquired benefits, thus suggesting the viability of our approach in real-world deployments. Accordingly, we do recognize the need for full-scale evaluation with real grid workload traces, a task that will nonetheless be carried out in the future.

Future work will also include experiments with different types of peer-to-peer overlay networks in order to gain a better understanding of its correlation to the meta-scheduling performance. Furthermore, additional local-scheduling policies would need to be considered, such as advance reservation,

backfill or priority scheduling. Finally, the encouraging results presented in this paper serve as a motivation for our future work, which will see ARiA integrated within the SmartGRID framework [31].

VII. Acknowledgments

This research is financially supported by the Swiss Hasler Foundation in the framework of the "ManCom Initiative", project Nr. 2122. The authors would like to thank Fulvio Frapolli for his useful feedback.

References

[1] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2nd edition, December 2003.

[2] Ian Foster. The anatomy of the grid: Enabling scalable virtual organizations. *Cluster Computing and the Grid, IEEE International Symposium on*, 2001.

[3] Jonathan Chin, Matthew J. Harvey, Shantenu Jha, and Peter V. Coveney. Scientific grid computing: The first generation. *IEEE Computing in Science and Engineering*, 7(5):24–32, 2005.

[4] D.J. Lutz, P. Mandic, S. Neinert, R. del Campo, and J. Jaehnert. Harmonizing service and network provisioning for federative access in a mobile environment. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 843–846, April 2008.

[5] Cecile Germain-Renaud, Charles Loomis, Jakub T. Moscicki, and Romain Texier. Scheduling for responsive grids. *Journal of Grid Computing*, 6:15–27, 2008.

[6] Hai Jin, Yi Pan, Nong Xiao, and Jianhua Sun, editors. *Grid and Cooperative Computing - GCC 2004: Third International Conference, Wuhan, China, October 21-24, 2004. Proceedings*, volume 3251 of *Lecture Notes in Computer Science*. Springer, 2004.

[7] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[8] Mathilde Romberg. The unicore architecture: seamless access to distributed resources. In *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on*, pages 287–293, 1999.

[9] A. Andrzejak, A. Reinefeld, F. Schintke, T. Schtt, C. Mastroianni, P. Fragopoulou, D. Kondo, P. Malecot, G. Cosmin Silaghi, L. Moura Silva, P. Trunfio, D. Zeinalipour-Yazti, and E. Zimeo. Grid architectural issues: State-of-the-art and future trends. CoreGRID White Paper, May 2008.

[10] Manfred Hauswirth and Roman Schmidt. An overlay network for resource discovery in grids. In *Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on*, pages 343–348, Aug. 2005.

[11] Haiying Shen, Ze Li, Ting Li, and Yingwu Zhu. Pird: P2p-based intelligent resource discovery in internet-based distributed systems. In *Distributed Computing Systems, 2008. ICDCS '08. The 28th International Conference on*, pages 858–865, June 2008.

[12] Junwei Cao, Daniel P. Spooner, Stephen A. Jarvis, and Graham R. Nudd. Grid load balancing using intelligent agents. *Future Generation Computer Systems*, 21(1):135–149, 2005.

[13] Vijay Subramani, Rajkumar Kettimuthu, Srividya Srinivasan, and P. Sadayappan. Distributed job scheduling on computational grids using multiple simultaneous requests. In *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, Washington, DC, USA, 2002.

[14] Mona Aggarwal, Robert D. Kent, and Alioune Ngom. Genetic algorithm based scheduler for computational grids. In *HPCS '05: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications*, pages 209–215, Washington, DC, USA, 2005. IEEE Computer Society.

[15] Rajiv Ranjan, Aaron Harwood, and Rajkumar Buyya. Peer-to-peer-based resource discovery in global grids: a tutorial. *Communications Surveys & Tutorials, IEEE*, 10(2):6–33, 2008.

[16] Alexander Fölling, Christian Grimme, Joachim Lepping, and Alexander Papaspyrou. Decentralized grid scheduling with evolutionary fuzzy systems. In Eitan Frachtenberg and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 16–36. Springer Verlag, 2009. Lect. Notes Comput. Sci. vol. 5798.

[17] Christian Grimme, Joachim Lepping, and Alexander Papaspyrou. Prospects of collaboration between compute providers by means of job interchange. In *Job Scheduling Strategies for Parallel Processing*, pages 132–151, 2007.

[18] Ruchir Shah, Bhardwaj Veeravalli, and Manoj Misra. On the design of adaptive and decentralized load balancing algorithms with load estimation for computational grid environments. *IEEE Transactions on Parallel and Distributed Systems*, 18(12):1675–1686, 2007.

[19] Arjav J. Chakravarti and Gerald Baumgartner. The organic grid: Self-organizing computation on a peer-to-peer network. In *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, pages 96–103, Washington, DC, USA, 2004. IEEE Computer Society.

[20] Graham R. Nudd, Darren J. Kerbyson, Efstathios Papaefstathiou, Stewart C. Perry, John S. Harper, and Daniel V. Wilcox. Pace–a toolset for the performance prediction of parallel and distributed systems. *International Journal of High Performance Computing Applications*, 14(3):228–251, 2000.

[21] Değer Cenk Erdil and Michael J. Lewis. Supporting self-organization for hybrid grid resource scheduling. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1981–1986, New York, NY, USA, 2008. ACM.

[22] Manish Arora, Sajal K. Das, and Rupak Biswas. A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments. In *ICPPW '02: Proceedings of the 2002 International Conference on Parallel Processing Workshops*, page 499, Washington, DC, USA, 2002. IEEE Computer Society.

[23] Barbara Kreaseck, Larry Carter, Henri Casanova, and Jeanne Ferrante. Autonomous protocols for bandwidth-centric scheduling of independent-task applications. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 26.1, Washington, DC, USA, 2003. IEEE Computer Society.

[24] Marco Fiscato, Paolo Costa, and Guillaume Pierre. On the feasibility of decentralized grid scheduling. In *Proceedings of the Workshop on Decentralized Self-Management for Grids, P2P, and User Communities (Selfman)*, Venice, Italy, October 2008.

[25] Deger Cenk Erdil and Michael J. Lewis. Grid resource scheduling with gossiping protocols. In *Peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on*, pages 193–202, Sept. 2007.

[26] Lijuan Xiao, Yanmin Zhu, Lionel M. Ni, and Zhiwei Xu. Gridis: An incentive-based grid scheduling. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, Washington, DC, USA, 2005. IEEE Computer Society.

[27] Kay Dörnemann, Jörg Prenzer, and Bernd Freisleben. A peer-to-peer meta-scheduler for service-oriented grid environments. In *GridNets '07: Proceedings of the first international conference on Networks for grid applications*, pages 1–8, ICST, Brussels, Belgium, 2007.

[28] Amos Brocco and Beat Hirsbrunner. Service provisioning framework for a self-organized grid. In *Proceedings of GridPeer 2009 Workshop, International Conference on Computer Communications and Networks*, pages 1–6, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[29] Open Grid Forum. Job submission description language (jsdl), http://www.ggf.org/documents/gfd.56.pdf. November 2005.

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

[31] Ye Huang, Amos Brocco, Béat Hirsbrunner, Michèle Courant, and Pierre Kuonen. Smartgrid: A fully decentralized grid scheduling framework supported by swarm intelligence. In *7th International Conference on Grid and Cooperative Computing*, pages 160–168. GCC2008, October 2008.