Agents as resource brokers in grids — forming agent teams

Wojciech Kuranowski[1], Marcin Paprzycki[2], Maria Ganzha[3], Maciej Gawinecki[3], Ivan Lirkov[4], and Svetozar Margenov[4]

[1] Software Development Department, Wirtualna Polska
ul. Traugutta 115C, 80–226 Gdansk
wkuranowski@wp-sa.pl

[2] Institute of Computer Science, Warsaw School of Social Psychology, ul. Chodakowska 19/31, 03–815 Warszawa, Poland,
marcin.paprzycki@swps.edu.pl

[3] Systems Research Institute, Polish Academy of Science,
ul. Newelska 6, 01-447 Warszawa, Poland
(maria.ganzha,maciej.gawinecki)@ibspan.waw.pl

[4] Institute for Parallel Processing, Bulgarian Academy of Sciences,
Acad. G. Bonchev, Bl. 25A, 1113 Sofia, Bulgaria
ivan@parallel.bas.bg margenov@parallel.bas.bg

Abstract. Recently we have proposed an approach to utilizing agent teams as resource brokers and managers in the Grid. Thus far we have discussed the general overview of the proposed system, how to efficiently implement matchmaking services, as well as proposed a way by which agents select a team that will execute their job. In this paper we focus our attention on processes involved in agents joining a team.

1 Introduction

In our recent work we have discussed how teams of software agents can be utilized as resource brokers and managers in the Grid. Thus far we have presented an initial overview of the proposed approach ([5]), studied the most effective way of implementing yellow-page-based matchmaking services ([4]), and considered processes involved in agents seeking teams to execute their jobs ([3]). The aim of this paper is to start addressing the question: how agent teams are formed?

To this effect, we start with an overview of the proposed system, consisting of the basic assumptions that underline our approach, followed by a UML Use Case Diagram. In the next section we discuss issues involved in agent to agent-team matchmaking. Paper is completed with UML-based formalization of the main process involved in agent joining an existing team, and report on the status of the implementation.

2 System overview

Let us start by making it explicit that in our work we follow these who claim that software agents will play an important role in design, implementation and long-term upkeep of large-scale software systems (see e.g. [8]). Second, our work assumes that software agents will be crucially involved in the future development of the Grid. While these two assumptions are not uncontroversial, arguments supporting them can be found, among others, in [7, 10]. The latter assumption is further supported by the body of research devoted to combining software agents and the Grid; summarized in [3]. Finally, we view the Grid as a global infrastructure (rather than a local / laboratory-based Grid). As a result, we deal with a situation similar to the P2P environment, where no centralized control over individual Grid nodes is exerted.

As a result of these assumptions we have functionalized the Grid as an environment in which workers (in our case *agent workers*) that want to contribute their resources (and be paid for their usage), meet and interact with users (in our case *agent users*) that want to utilize offered services to complete their tasks and (in [5]) proposed a system based on the following tenets:

– agents work in teams (groups of agents)
– each team has a single leader—*LMaster agent*
– each *LMaster* has a mirror *LMirror agent* that can take over its job
– incoming workers (*worker agents*) join teams based on individual criteria
– teams (represented by *LMaster*s) accept workers based on individual criteria
– decisions about joining and accepting involve multicriterial analysis
– each *worker agent* can (if needed) play role of an *LMaster*
– matchmaking is yellow page based [11] and facilitated by the *CIC agent* [1]

Combining these propositions resulted in the system represented in Figure 1 as a Use Case diagram. Let us now focus our attention on interactions between

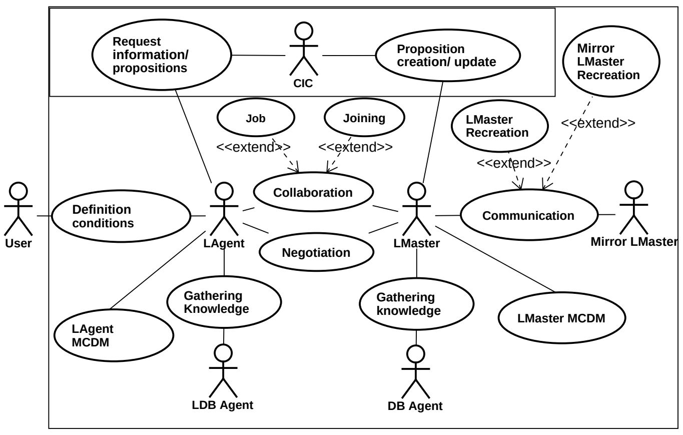


Fig. 1. Use Case diagram of the proposed system

the *User* and its representative: *LAgent* and agent teams residing in the system (remaining information can be found in [5]). Let us assume that the system is already "running for some time," so that at least some agent teams have been already formed. As a result, team "advertisements" describing: (1) what resources they offer, and (2) characteristics of workers they would like to join their team are posted with the *Client Information Center* (*CIC*). Let us also note that the *User*, can either contribute resources to the Grid, or utilize resources available there. Interestingly, both situations are "Use Case symmetric" and involve the same pattern of interactions between agents representing the *User* and the system.

User who wants to utilize resources in the Grid communicates with its local agent (*LAgent*) and formulates conditions for executing a job. The *LAgent* communicates with the *CIC* to obtain a list of agent teams that satisfy its predefined criteria. Next, the *LAgent* communicates with *LMasters* of the remaining teams and utilizes the Contract Net Protocol ([6]) and multicriterial analysis ([2]) to evaluate obtained proposals. If the *LAgent* selects a team to execute its job, a contract is formed. If no such team is found (e.g. if nobody is willing to execute a 10 hour job for 5 cents), the *LAgent* informs its *User* and awaits further instructions (for more details see [3]).

The remaining part of the text will be devoted to the situation when *User* requests that its *LAgent* joins a team and works within it (e.g. to earn extra income for the *User*).

3 Selecting team to join

The general schema of interactions involved in *LAgent* selecting the team to join is very similar to that described above. First, the *User* specifies the conditions of joining, e.g. minimum payment for job execution, times of availability etc. Then she provides its *LAgent* with the description of resources offered as a service, e.g. processor power, memory, disk space etc. The *LAgent* queries the *CIC* which agent teams seek workers with specified characteristics. Upon receiving the list of such teams, it prunes teams deemed untrustworthy (e.g. teams that did not deliver on promised payment) and contacts *LMasters* of the remaining teams (if no team is left on the list, the *LAgent* informs its *User* and awaits further instructions). Negotiations between the *LAgent* and the *LMasters* take form of the FIPA Contract Net Protocol [6]. The summary of this process is depicted as a sequence diagram in Figure 2. For clarity, this sequence diagram is simplified and does not include possible "negative responses" and/or errors. Note that registering with the *CIC* takes place only once — when a new *LAgent* joins the system (or when it wants to start anew, i.e. to erase bad reputation). All subsequent interactions between the *CIC* and a given *LAgent* involve only checking credentials. The sequence diagram includes also processes involved in "mirroring." In our system we assume that the *LMaster* has its mirror, the *LMirror* agent. The role of this agent is to become the *LMaster* in the case when the current *LMaster* "disappears." Let us note that it is only the *LMaster* that has complete information about team members, jobs that are executed (and by whom), etc. Therefore, disappearance of the *LMaster* would imemdiately "destroy the

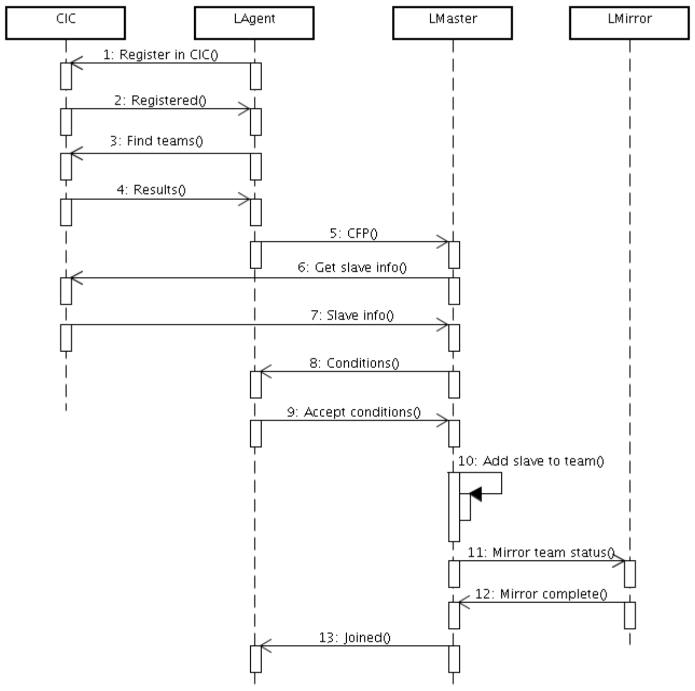


Fig. 2. Sequence diagram of interactions when an agent is seeking a team to join.

team.” In an attempt to avoid such a situation the *LMaster* shares all vital information with the *LMirror*. Obviously, it is possible that both the *LMaster* and the *LMirror* “go down” simultaneously, but our goal is only to introduce some degree of resilience (not to build a fault tolerant environment). Since this subject is out of scope of this paper it is omitted from further considerations.

3.1 Representing conditions of joining

Let us now discuss representation of (1) resources that the *LAgent* brings to the team, and (2) its conditions of joining. Before we proceed, let note that in an ideal situation, an all-agreed “ontology of the Grid” (that would include both the resources and the economical model) would exist. Unfortunately, while there exist separate and incompatible attempts at designing such an ontology, currently they are only “work in progress.” Therefore, we focused our work on designing and implementing agent system skeleton, while using simplistic ontologies (and thus all proposals presented below should be viewed with this fact in mind). Obviously, when the Grid ontology will be agreed on, our system can be easily adapted to utilize it. In [5] we presented our ontological representation of computational resources. Here, we describe parameters used to negotiate conditions of joining.

Currently we utilize three parameters of joining: (1) price per work-hour, (2) work time—specific times of the day when the resource is to be available, and (3) length of contract—time interval that a given *LAgent* is offering to be a member of a given team. While the contract holds for a limited time, we assume that if both sides are satisfied, it can be extended for subsequent (and possibly longer) time periods.

What follows is an instance of joining conditions that ontologically depict a computer: (1) with an Intel processor running at 3 GHz, (2) that offers to users 256 Mbytes of RAM and (3) 20 Gbytes of disk space, and that is offered to the team under the following conditions: (4) it is available every night between 23:50 and 8:15, and (5) wants to sign a contract for 7 days. Note that payment conditions are not specified (they are a part of the response of the *LMaster*).

```
(cfp
  :sender (agent-identifier :name proteus@bach:1099/JADE)
  :receiver (agent-identifier :name zerg@chopin:1099/JADE)
  :content
    ((action
      (agent-identifier :name zerg@chopin:1099/JADE)
      (take-me
        :configuration (hardware
          :cpu 3.0
          :memory 256
          :quota 20)
        :conditions (condition
          :availability (every-day
            :when (period
              :from 00000000T23500000
              :to 00000000T08150000))
          :contract-duration +00000007T000000000))
  :language fipa-sl0
  :ontology joining-ontology
  :protocol fipa-contract-net
)
```

This type of an information is used in two situations. First, each team looking for members advertises the resources it is looking for. Such an advertisement is an instance of an ontology, where parameters with numerical values (e.g. processor speed or available disk space) are treated as minimal requirements, while parameters that describe necessary software are hard constraints that have to be satisfied. Note that descriptions of sought workers include only resource parameters, but they do not include specific offers related to, for instance, payments for working for the team. In this way, when the *LAgent* requests list of teams that look for members, information about its own resources is used as a filter. For querying ontologically demarcated information we use SPARQL query language [9]. Therefore when the *LAgent* representing the above described computer communicated with the *CIC*, the following SPARQL query is executed.

```
PREFIX Grid: <http://Gridagents.sourceforge.net/Grid#>
SELECT ?team
WHERE {
  ?team Grid:needs ?machine .
  ?machine Grid:hasCPU ?cpu ;
           Grid:hasMemory ?mem ;
           Grid:hasQuota ?quota .
  FILTER ( ?cpu <= "3.0"^xsd:float ) .
  FILTER ( ?mem <= "256"^xsd:integer ) .
  FILTER ( ?quota <= "20480"^xsd:integer ) .
}
```

Second, when the *LAgent* issues a CFP (arrow 5 in Figure 2), the complete information describing resources and conditions of joining is included in the CFP and is used by the *LMaster* to prepare an offer. Let us note that specific offers are based on: available resources, overall availability to do the job, etc. Furthermore, note that each time a given *LAgent* issues a CFP it may specify different resource as: (1) the same *LAgent* may represent *User*'s different machines, or (2) for a single machine at one time available disk space may be 5 Gbytes, while at another time 25 Gbytes (e.g. depending on the number of stored MP3 files).

3.2 Negotiations

Let us now focus our attention on negotiations. The first step is the *LAgent* sending a CFP (arrow 5 in Figure 2) containing resource description and conditions of joining (see ontology snippet above). Upon receiving the CFP each *LMaster* contacts the *CIC* to make sure that this particular *LAgent* is registered with the system (arrows 6 and 7 in Figure 2). To somewhat improve safety of the system we assume that only *LAgent*s that are registered with the *CIC* can join agent teams.

On the basis of the CFP, *LMaster*s prepare their response. First, CFPs that do not satisfy hardware / software requirement are refused (e.g. worker that does not have Maple, cannot join a team that requires Maple). Second, each *LMaster* utilizes its knowledge about past jobs to establish base price per hour and base system that matches it. Currently, this price is split between the three components that exist in our ontology (processor speed: P_b, memory: M_b, disk space: D_b). As a result we obtain processor cost P_c, memory cost M_c and disk cost D_c (such that the base cost $B_c = P_c + M_c + D_c$). This information is used to estimate the "value" of the new worker in the following way, (assume that the new worker has processors speed P, memory M and disk space D):

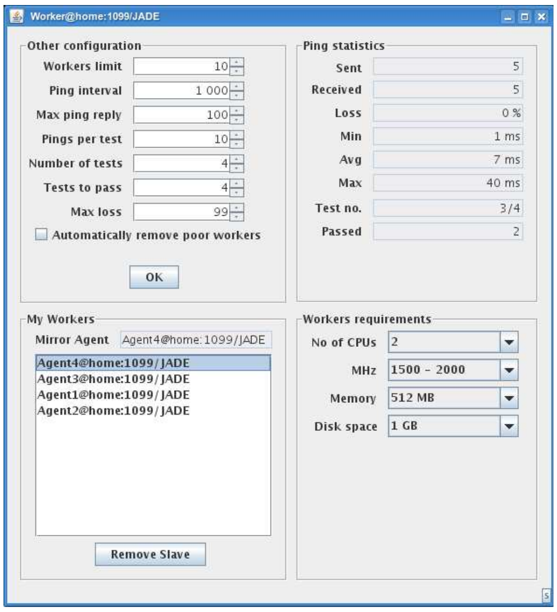
$$Cost = \alpha\Big(\frac{P}{P_b}P_c + \frac{M}{M_b}B_c + \frac{D}{D_b}D_c\Big), \tag{1}$$

Where $\alpha \in [0, 1]$ denotes the overhead charged by the *LMaster*. Obviously, this model is extremely simplistic, but our goal was not to build a complete economical model of the Grid (for this, one would need a Grid ontology), but to specify a replaceable function that can be used in our system skeleton.

Responses from *LMaster*s can have the following forms: (1) refusal (an ACL REFUSE message), (2) lack of response in a predefined by the *LAgent* time, (3) a specific offer (an ACL PROPOSE message). The *LAgent* awaits a specific time for responses and then finds the best of them (currently the response contains only the proposed price; as soon as a more complicated response is to be used a multicriterial analysis has to be applied). If the best available offer is above its own private valuation an agent team is selected to be joined (arrow 9 in Figure 2). If no acceptable offer is received, *User* is informed and *LAgent* awaits further instructions. Note that, the final confirmation is depicted as arrow number 11 in Figure 2. According to the Contract Net Protocol, since the *LAgent* was the originator of the negotiations, it has to be the receiver of the final confirmation.

3.3 Implementation

Currently we are implementing the above described processes. Note that they cannot be implemented without additional mechanisms involved in agent team management (that were omitted here due to the lack of space). To illustrate the state of our implementation, in Figure 3.3, we present the GUI of the *LMaster* agent. Most important informations, in the context of this paper, are (1) the



Worker@home:1099/JADE

Other configuration

Workers limit	10
Ping interval	1 000
Max ping reply	100
Pings per test	10
Number of tests	4
Tests to pass	4
Max loss	99

☐ Automatically remove poor workers

OK

Ping statistics

Sent	5
Received	5
Loss	0 %
Min	1 ms
Avg	7 ms
Max	40 ms
Test no.	3/4
Passed	2

My Workers

Mirror Agent: Agent4@home:1099/JADE

Agent4@home:1099/JADE
Agent3@home:1099/JADE
Agent1@home:1099/JADE
Agent2@home:1099/JADE

Remove Slave

Workers requirements

No of CPUs	2
MHz	1500 - 2000
Memory	512 MB
Disk space	1 GB

Fig. 3. GUI of the *LMaster* agent.

Workers requirements box and (2) the *My Workers* box. The first one specifies that this *LMaster* is interested in workers that have 2 processors running at between 1.5 and 2.0 GHz, minimal memory of 512 Mbytes and disk space of 1 Gbyte. At the same time we can see that this *LMaster* is currently managing a team of 4 workers.

The *Other configuration* box represents options related to agent team management. We can see there that this *LMaster* will accept no more that 10 workers, as well as a number of parameters used to monitor which worker agents are down and thus will not continue executing their jobs. Finally, the *Ping statistics* box provides statistical results of monitoring sessions. Describing these (already working) mechanisms is outside of scope of this paper.

4 Concluding remarks

The aim of this paper was to discuss processes involved in an agent joining a team, conceptualized within the framework of the proposed earlier agent-team-based Grid resource brokering and management system. Processes described in this paper, while relatively simplistic, can be easily augmented to a more robust version. Currently we are proceeding with implementation of the above described processes. This involves also development of agent team management tools that have been briefly mentioned in Section 3.3.

Acknowledgments

Work presented here is a part of the Poland-Bulgaria collaborative grant: "Parallel and distributed computing practices".

References

1. C. Bădică, A. Bădită, M. Ganzha, and M. Paprzycki. Developing a model agent-based e-commerce system. In J. L. et. al., editor, *E-Service Intelligence - Methodologies, Technologies and Applications*, pages 555–578. Springer, Berlin, 2007.
2. J. Dodgson, M. Spackman, A. Pearman, and L. Phillips. *DTLR multi-criteria analysis manual.* UK: National Economic Research Associates, 2001.
3. M. Dominiak, M. Ganzha, and M. Paprzycki. Selecting grid-agent-team to execute user-job—initial solution. In *Proceedings of the Conference on Complex, Intelligent and Software Intensive Systems*, pages 249–256, Los Alamitos, CA, 2007. IEEE CS Press.
4. M. Dominiak, W. Kuranowski, M. Gawinecki, M. Ganzha, and M. Paprzycki. Efficient matchmaking in an agent-based grid resource brokering system. In *Proceedings of the International Multiconference on Computer Science and Information Technology*, pages 327–335. PTI Press.
5. M. Dominiak, W. Kuranowski, M. Gawinecki, M. Ganzha, and M. Paprzycki. Utilizing agent teams in grid resource management—preliminary considerations. In *Proceedings of the IEEE J. V. Atanasoff Conference*, pages 46–51, Los Alamitos, CA, 2006. IEEE CS Press.
6. FIPA Contract Net Protocol specification. `http://www.fipa.org/specs/fipa00029/SC00029H.html`.
7. I. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 8–15, Los Alamitos, CA, 2004. IEEE CS Press.
8. N. R. Jennings. An agent-based approach for building complex software systems. *CACM*, 44(4):35–41.
9. SPARQL query language for RDF. `http://www.w3.org/TR/rdf-sparql-query`.
10. H. Tianfield and R. Unland. Towards self-organization in multi-agent systems and grid computing. *Multiagent and Grid Systems*, 1(2):89–95, 2005.
11. D. Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 89–98, New York, NY, USA, 2002. ACM Press.