Alternatives for Implementing Methods for Finding Agents in Multi-Agent Systems

Mariano Tucat
mt@cs.uns.edu.ar

Alejandro J. García
ajg@cs.uns.edu.ar

Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering, Universidad Nacional del Sur,
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

ABSTRACT

Interaction is an essential characteristic of Multi-Agent Systems (MASs). Agents that are part of an MAS usually interact by exchanging messages according to some conversation policy. Therefore, the ability to find other agents and exchange messages with them are features that need to be implemented in agents that are part of an MAS. Finding an agent means being aware of its existence and obtaining the information needed to send it a message. There exist different alternatives of implementing the capability of finding other agents. In this work, we will analyze the alternatives of implementing a discovery mechanism, a centralized Agent Name Server (ANS), a distributed ANS, or several ANSs.

1 INTRODUCTION

Interaction is an essential characteristic of Multi-Agent Systems (MASs). Agents that are part of an MAS usually interact by exchanging messages according to some conversation policy. Therefore, the abilities to find other agents and exchange messages with them are features that need to be implemented in agents that are part of an open MAS. Finding an agent means being aware of it existence and obtaining the information needed to send it a message.

There exist different alternatives of implementing MASs. One alternative is to implement the whole MAS ad-hoc. Another way is to use a MultiAgent development framework such as JACK [5], JADEX [7] or 3APL [1]. Finally, another alternative is to implement an MAS using a Programming Language extension that provides the capabilities of finding other agents and exchanging messages.

The alternative of implementing the whole MAS ad-hoc means that the developer of the system is allowed to choose the architecture of each agent, the way they interact and also the way they locate each other. Thus, this alternative has the advantage of a great flexibility in the design and implementation of the system. However, the main disadvantage is that the developer may have to implement everything, including the mechanisms used to locate the agents and also the primitives for exchanging messages.

The alternative of using a MultiAgent development framework such as JACK, JADEX or 3APL, will probably constrain the developer of the systems to use a specific agent architecture and may also determine the way in which the agents should exchange messages. This alternative has the advantage of reducing the amount of work needed to implement the system.

Finally, the alternative of implementing the MAS using a Programming Language extension or a framework that provides the capabilities of finding other agents and exchanging messages (such as [4], JADE [6] or MadKit [8]) allows a flexible design of the system. This way of implementing the MAS will try to maintain the advantages of the alternatives mentioned before also avoiding their disadvantages.

In [4], we have proposed an extension of the Logic Programming Language Prolog that provides the agents the capability of finding other agents and also facilitates the way of exchanging messages. The proposed extension includes some primitives that were motivated by the implementation of multi-agent systems for dynamic and distributed environments, where intelligent agents communicate and collaborate. One of the main goals of these primitives was to hide as much of the implementation details as possible, in order to provide a transparent way of both finding agents and programming agent interaction.

These primitives allow the creation of several independent MASs where agents communicate with others by just knowing the other agents'

Partially supported by CONICET (PIP 5050) and Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096)

names. These MASs are groups of agents and any agent may join and leave these groups dynamically. To be member of an MAS means that the agent knows which other agents belong to the MAS. Any agent is allowed to be member of several MASs simultaneously, and also to join other existing MASs, create new ones and leave the ones it is connected to.

There exist different alternatives of implementing the capability of finding other agents. In this work, we will analyze the following four alternatives:

1. a proper discovery mechanism in the agent code,
2. a centralized Agent Name Server (ANS),
3. a distributed ANS, or
4. several ANSs.

2 FINDING AGENTS

As stated before, finding an agent means being aware of its existence and obtaining the information needed to send it a message. There exist different alternatives of implementing the capability of finding other agents. One alternative is to provide the agents the ability of searching for the existence of agents. That is, to provide a mechanism in which each agent looks for agents that are also trying to interact with other agents.

Another alternative is to allow the agents to find each other by using one or more special agents in which the agents register themselves and obtain information of other registered agents. This kind of agents is often called Agent Name Server (ANS) and they store the information of the registered agents and answer any query made by them.

The ANS is similar to the Agent Management System (AMS) defined by FIPA [3] since both allow the agents to register themselves and also to obtain information of other registered agents. However, the AMS is also responsible for the creation and deletion of agents and represents the managing authority of the Agent Platforms, *i. e.* the physical infrastructure in which agents are deployed.

We will explain next the alternatives mentioned before for implementing the capability of finding other agents. In this paper, we will assume that an agent is implemented by a process and we will use the term node as any environment capable of executing a process, such as a desktop PC or a Laptop.

2.1 A Discovery Mechanism

The agents may broadcast messages to find each other (see [2] for details). This way of implementing agent discovery allows a great flexibility in the initialization of the system and also allows the agents to discover different MASs dynamically. However, note that using broadcast, some messages may be lost or delivered out of order. Also, losses and misordering may vary by agent, which means that some agents may receive the message as intended while others may experience various losses. Thus, in a domain in which there exist several agents, this mechanism is not allowed to ensure that any agent will discover any other existent agent in the system. However, this alternative may be useful for other purposes.

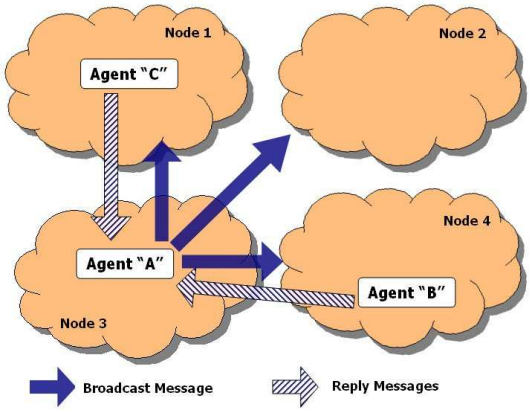

Figure 1: The mechanism of discovery implemented with broadcast messages.

As an example, in Figure 1, Agent "A" broadcasts a discovery message (dark arrows) and the existing agents ("B" and "C") answers this message (dashed arrows). There exist different alternatives of the way of using this kind of messages in order to allow the agents to find each other. One alternative is to use them to locate any existing agent in the system and then, the agent may decide with which other ones it wants to interact. Another alternative is to specify in these messages certain information that allows the agents receiving the message to decide whether they want to interact with this agent or not. While the first alternative may be useful in specific domains with a small number of agents, the second alternative allows an efficient but difficult way of implementing MultiAgent Systems of large number of agents.

The first alternative is implemented by every agent replying any broadcast message received. Thus, in a system with several agents, this behavior may overload the network. Also note that, in this kind of systems, it is unlikely that the agents need to interact with every other existing agent.

That is, the agents should only know the existence of the agents they may interact with.

The second alternative allows this by specifying in the broadcast message certain information that the agents use to detect common interests, thus creating independent MASs. Hence, only the agents that share the same interests will be members of the same MAS, replying the messages specifying common interests and achieving an efficient use of the network. However, this alternative requires a more detailed implementation and also a standard or shared way of expressing agents interests.

2.2 A Centralized Agent Name Server

This centralized approach has the advantage of an easy development of the ANS and, in the case of any other agent, the easy way of obtaining the information of the other registered agents. Note that the ANS is responsible for maintaining the data entries consistent, specially the agent identifiers, in which duplicates are not allowed. Thus, a centralized implementation of this agent reduces it complexity.

However, this approach has some disadvantages. One disadvantage is that the ANS may represent a performance bottleneck when the system has several agents. Also note that this way of implementing the ANS represents a single point of failure in the system. This means that a failure in the ANS may disable the services provided and thus, the agents will not be able to find each other.

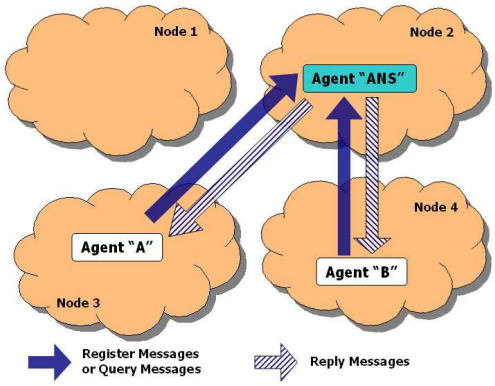

Figure 2: A Centralized ANS

In this alternative, the problem of knowing the location of other agents is solved with the existence of the ANS that maintains this information. Thus, the problem of finding other agents is reduced to finding only one agent: the ANS. Any agent should know the location of the ANS in order to register itself and also to obtain the location of other registered agents. For example, in Figure 2, Agent "A" (in Node 3) registers itself by exchanging messages with the ANS (in Node 2) and it may also query for obtaining the other registered agents. The same happens for Agent "B" in Node 4.

There exist different ways of solving the problem of finding the ANS. One alternative is to hardcode its location to a specific node, as in Figure 2, in which the ANS may be hardcoded to Node 3, and thus, any agent will know where to find it. Another alternative is to use a mechanism of agent discovery as explained in the previous subsection. Note that, using the first alternative reduces the flexibility of the system, however, this reduction in the flexibility only affects the ANS, allowing the other agents to be executed in any node. Also note that implementing a discovery mechanism to find a single known agent will avoid the problems present in this alternative.

2.3 A Distributed Agent Name Server

This alternative eliminates the single point of failure of the system and also the performance bottleneck present in the centraliced approach. However, the main problem with this alternative is the burden of implementing the ANS.

The way in which this alternative is implemented is using only one ANS created by several processes running one in each node of the system. As can be seen in Figure 3, there exists only one logical ANS, implemented by the processes executing on each node. Thus, the agents interact with the ANS through the process running in the same node, solving the problem of finding the ANS present in the alternative mentioned before.

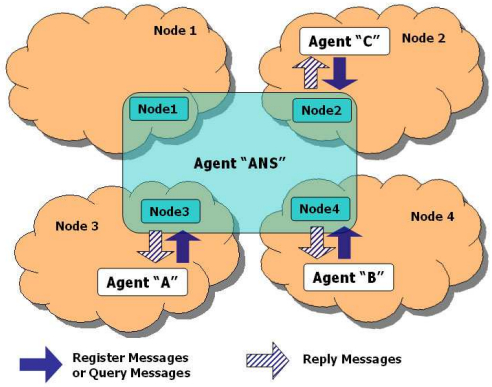

Figure 3: A Distributed ANS

One of the problems presented in the implementation of this alternative is the different ways of storing the information gathered by the ANS. One approach is to replicate all the information in

all the different processes implementing the ANS. Another way of doing this is to distribute all the information without replicating it.

While the first approach allows the agents to obtain the information of other registered agents quickly, the services that modify this information may require the exchange of several messages in order to maintain this information consistent. The other way allows easier modifications of the information stored, but may require more time to obtain information of a specific agent.

2.4 Several Agent Name Servers

Another alternative is to implement one ANS for each node, achieving most of the advantages of the distributed approach like the easy finding of the ANSs, and also avoiding the disadvantages of the centraliced one, like the performance bottleneck and the single point of failure in the system. The problems with this approach are the same as the ones in the distributed approach, with the difference that the agents are aware of the existence of several ANSs.

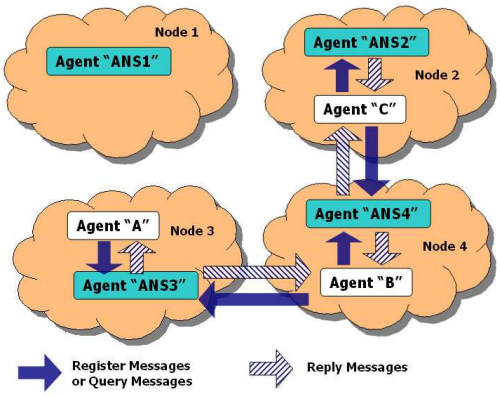

Figure 4: Several ANSs

Thus, one alternative to solve such problems is to allow the agents to create groups of interests or MASs. Each MAS must be managed by only one ANS. The agents may register themselves in different MASs, also simultaneously. This way of storing the information of the agents registered is similar to the centraliced approach, achieving the advantage of an easy implementation.

Any agent that wants to interact with other agent may first interact with the local ANS in order to register itself and to obtain information of a specific MAS. The local ANS may be the manager of the MAS that the agent is looking for, in which case it registers the agent. However, the local ANS may not be responsible of the MAS, and thus, it may inform the agent which ANS is managing the determined MAS.

The creation of independent MASs may be implicit whenever an agent requires to register itself to an inexistent MAS. In this case, the local ANS asks the other ANSs for the existence of the specified MAS obtaining no positive answers. Thus, it decides to create the MAS and informs this creation to the rest of the ANSs. Note that the process of creating an MAS should avoid the creation of MASs with the same interests managed by different ANSs.

3 COMPARING THE ALTERNATIVES

In the previous section we have explained four different alternatives of implementing the capability of finding other agents. There exist several differences between these alternatives as we will show next. These differences will allow the developer to choose between them also considering the domain in which the alternative will be used.

The alternative of using a discovery mechanism allows a great flexibility in the initialization of the system and also allows the agents to discover different MASs dynamically. Also note that this alternative avoids relying on the existence of one or more specific agents as the other approaches does, thus representing an important disadvantage.

However, note that using broadcast, some messages may be lost or delivered out of order. Also, losses and misordering may vary by agent, which means that some agents may receive the message as intended while others may experience various losses. Thus, this mechanism is not allowed to ensure the discovery of every existent agent in the system.

This problem is avoided in the other alternatives by the existence of the ANS. The centralized approach has the advantage of an easy development of this agent and, in the case of any other agent, the easy way of obtaining the information of the other registered agents. However, this approach has the disadvantage that the ANS may represent a performance bottleneck when the system has several agents. Also note that this way of implementing the ANS represents a single point of failure in the system.

The distributed approach eliminates the single point of failure of the system and also the performance bottleneck present in the centralized one. However, the main problem with this alternative is the burden of implementing the ANS. Finally, implementing several ANSs has the advantage of an easy implementation of each agent like the centraliced approach, also avoiding its disadvantages of single point of failure and performance bottleneck with the existence of several ANSs.

A discovery mechanism may be used in the implementation of these alternatives. It may be used by the agents to locate the centralized ANS. It may also be used by the distributed ANSs to discover themselves dynamically. Note that using a discovery mechanism in these particular cases will probably avoid the disadvantages of using this alternative to find every agent in the system.

Considering the domains of application of these approaches, in systems with a small number of agents, a centralized ANS may solve the problem of finding agents with good results and also allowing an easy implementation of the system. In this domains, implementing a discovery mechanism may be a good alternative whenever the developer wants to avoid the existence of centralized information that means the existence of a single point of failure.

In domains in which the number of agents is important, the centralized approach will probably result in a performance bottleneck. Thus, the alternative of implementing several ANSs and allowing the creation of independent MASs avoids this problem and also the complexity of implementing the other alternative, *i. e.* a distributed ANS. In the case of the discovery mechanism, the losses and misordering of the messages is such domain will be greater also increasing the time needed to find other existent agents, turning this alternative less feasible.

4 CONCLUSION

In this paper we have mentioned different alternatives of implementing MASs. One of the features that must be implemented in the development of any MAS is the way the agents find or locate each other. We explained four different alternatives of implementing this capability. We have also shown the differences existing between these alternatives.

We compared these alternatives, finding that the alternative of implementing a centralized ANS allows an easy development of small MAS. However, in larger domains, the approach of implementing several ANSs may allow a better performance of the whole system also avoiding the complexity of implementing a distributed ANS. A discovery mechanism may be used in the implementation of these alternatives. A discovery mechanism may by also used whenever the developer prefers to avoid the existence of centralized information, at the cost of a more time consuming location of the agents.

References

[1] 3APL. An Abstract Agent Programming Language. http://www.cs.uu.nl/3apl/.

[2] B. Langley, M. Paolucci, and K. Sycara. Discovery of Infrastructure in Multi-Agent Systems. In *Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

[3] FIPA. Foundation for intelligent physical agents. http://www.fipa.org.

[4] Alejandro J. García, Mariano Tucat, and Guillermo R. Simari. Interaction Primitives for Implementing Multi-agent Systems. In *VII Argentine Symposium on Artificial Intelligence*, Rosario, Argentina, August 2005.

[5] JACK. JACK Intelligent Agents Framework. http://www.agent-software.com/.

[6] JADE. Java Agent DEvelopment Framework. http://www.lpa.co.uk/atk.htm.

[7] JADEX. Jadex BDI Agent System. http://vsis-www.informatik.uni-hamburg.de/projects/jadex/.

[8] The MadKit Project. Multi-Agent Development Kit. http://www.madkit.org/.