Abstract

In multi-agent systems, blackboard communication provides an easy way for agents to communicate. However, in a distributed architecture, the blackboard is commonly implemented in a single node, that is the “blackboard server”. This approach may work well for a small number of agents. Nevertheless, when the number of agents increases, the blackboard server becomes the bottleneck for the system scalability. In this work we propose a novel agent communication system based on distributed-shared mechanisms that allows the implementation of a shared address space on a distributed system, without centralizing the blackboard. Our idea is to distribute the shared data over the nodes and use a message passing sub-system, totally transparent to the agents. We implemented this idea in a multi-agent system based on social laws, and used a distributed-shared memory system to handle the messaging. We created a conflict simulator, and showed that the distributed-shared mechanisms ease the implementation of a truly distributed blackboard.

1. Introduction

Although there are many different definitions for computational agents in the literature, it is generally agreed by researchers and developers of software agents that an agent is characterized by: acting on behalf of users; being able to perceive the environment; having a set of objectives and taking actions so as to accomplish these objectives; and being autonomous. In other words, autonomous agents are small self-contained programs that can solve simple problems in a well-defined domain [13].

Computational agents are able to interact with other agents, performing their activities in a society. A society is composed by two or more problem-solving agents, that collaborate, forming a Multi-Agent System (MAS) [22].

Communication, coordination and negotiation in the presence of conflicts are the key issues in a MAS. Communication facilitates cooperation, it enables agents to “talk” to each other helping them to reach a common goal. Coordination can be defined as the sharing of a common knowledge, including the mutual comprehension and the diffusion of the knowledge. Without coordination, a group of agents can generate a chaotic behavior. Conflicts between agents may arise from simple limited resource contention to more complex issue-based computations where the agents disagree because of discrepancies between their domains of expertise [4].

The conflicts in MAS can be solved by: i) environment monitoring [21]; ii) direct negotiation between agents [8]; iii) human intervention; or iv) social or environmental laws [17]. Therefore, a society of agents needs an architecture for agents interactions. The interactions provide the interchange of knowledge, goals or choices, since each agent
has only a partial vision of the system.

The interactions between agents are realized by means of two types of communication: direct and indirect. In a direct communication, the agents know each other a priori and the sender explicitly addresses the receiver in order to interact. In an indirect communication, the agents don’t know each other and the interaction occurs through a shared space, called blackboard.

In a society where the communication is direct, the agents must register in the network in order to know the identification of each other and send explicit messages. This requires the definition of a communication protocol. The indirect communication abstracts from the programmer all the low-level details of the communication. It is a simpler model, where the communication is implicit through reads and writes on the blackboard, that all the society has access.

Although the indirect communication is simpler, its implementation is highly dependable on the parallel architecture used by the society of agents. When the architecture has a physically shared memory between computational nodes, the implementation of indirect communication is straightforward, the blackboard lies in the shared memory, allowing reads and writes from all agents. These architectures are, however, very expensive and usually apply a small number of computing nodes.

Recently, however, the decreasing cost and high availability of commodity PCs and network technologies make cluster of PCs a low-cost alternative for running complex MAS with a good performance potential. The implementation of the blackboard in distributed architectures like this, on the other hand, is not straightforward, since there is no shared address space. The most popular solution to this problem is to implement the blackboard in a single node as a “blackboard server”. All the reads and writes to the blackboard must be replaced by explicit messages to the blackboard server. Therefore, this solution reduces the simplicity of blackboard communication and includes a potential performance and scalability problem. It may work well for a small number of agents. Nevertheless, when the number of agents increases, the blackboard server becomes overloaded, generating a performance bottleneck, reducing the scalability of the system.

In this work we propose a novel agent communication system based on distributed-shared memory (DSM) mechanisms, widely used in parallel systems [11, 16, 20], that allows the implementation of a shared address space on a distributed system, without centralizing the blackboard in a single node. Our idea is to distribute the shared data over the system nodes and use a message passing sub-system totally transparent to the programmer, in the same way a software DSM system does. The agents access the blackboard as it was shared and there is a sub-system responsible to transform these shared accesses in messages over the network.

In this way, we are proposing the implementation of a distributed and scalable blackboard that can be used in a MAS running in a low-cost architecture.

We have implemented this idea over a MAS based on social laws, called Tri-Coord, and used the TreadMarks DSM system to handle the message passing. In this preliminary study, we are interested in showing how mechanisms, as the ones used in DSM systems, can be explored to remove from the agents the responsibility of orchestrating all the message exchanging applied to solve conflicts or share knowledge in a MAS. We have created a conflict simulator in order to generate the need of communication between agents in Tri-Coord and showed that the distributed-shared memory system is able to maintain the shared memory over the distributed system.

The remainder of this paper is organized as follows. In the next section we present the related work. Following we review some basic concepts on MAS. Section 4 presents the Tri-Coord MAS used as the basis of for our distributed blackboard implementation. In Section 5 we review some basic concepts on distributed-shared memory systems and describe the TreadMarks software DSM system. Section 6 describes our proposal of a blackboard implemented with DSM mechanisms in the scope of the Tri-Coord system. Finally, in Section 7 we present our conclusions and proposals for future work.

2. Related Work

Due to the lack of space to cover the vast area of multi-agent systems, we restrict our related work to blackboard implementations. Historically, the blackboard model arose from abstracting features of the HEARSAY-II speech understanding system [9]. The initial idea was to build an architecture for knowledge-based systems, where independent Knowledge Sources (KSs) communicate only through a shared blackboard. For parallel and distributed execution of KSs, there are three different design alternatives: shared memory blackboard, blackboard server and distributed blackboard.

In the shared memory blackboard approach, each KS has direct access to a shared structure. The work by Ensor and Gabble [3] is a good example of this kind of system, but relies on a multiprocessor architecture. Subsequently, many parallel blackboard systems have been built, e.g. [2, 12]. These systems, however, rely on the use of a server node to implement the blackboard, that is usually called the blackboard server. The sharing of the blackboard information are implemented by message exchanging between the agents and the blackboard server. This kind of solution, however, has two main drawbacks: it burdens the agent with the need of exchanging message with the server, and it is not scalable.

In this way, we are proposing the implementation of a distributed and scalable blackboard that can be used in a MAS running in a low-cost architecture.
In order to solve the scalability problem of the blackboard server solution, some distributed blackboard solution were proposed. The work by Jiang et al. [6] proposed the construction of a blackboard communication architecture based on graph theory. The model allows the existence of sub-blackboards and computes the communication topology among sub-blackboards based on Steiner Tree method. The work by Tnazefti et al. [19] proposed a hierarchical multi-agent system, composed of several control/monitoring agents. The communication between a parent agent and its children is done through blackboards. These works introduced the concept of sub-blackboards, avoiding the potential performance bottleneck of the blackboard server. Our work, also avoids this bottleneck, but using a straightforward communication architecture: reads and writes to the blackboard structure.

3. Multi-agent Systems

A society of agents can be classified according to: i) the type of agent; ii) the number of agents; and iii) the behavior rules [15]. When the society is classified according to the agents, it can be: homogeneous (all the agents are identical) or heterogeneous (there are different agents). When the society is classified according to the number of agents, it can be: closed (there are a fixed number of agents) or open (the number of agents may vary, agents can be included or removed on demand). When the society is classified according to the agents behavior, it can be: based on laws (there are rules that guide the agents behavior) or without law (there are no rules for the agents).

3.1. Interaction

Regardless of the society type, the social group of agents does collective operations called interactions. The interactions provide the agent the knowledge of the behavior and activities of the other agents, and help all the agents to reach a common goal.

3.2. Conflict Resolution

Considering that the agents of a society have distinct knowledge and resolution methods, conflict situations may arise [14]. The conflicts can be positive or negative. Positive conflicts occur whenever there are agents capable of performing a task or whenever there are agents with different, but complementary, results. Negative conflicts occur when there is no agent capable of performing a task or there are different results that are contradictory or inconsistent. All the conflict situations must be solved by negotiation.

3.3. Negotiation

The negotiation between two or more agents aims the agreement upon a certain cooperation, and the definition of the tasks assigned to the agents. All kind of agent negotiation, however, requires that the agents communicate to each other in order to solve a distributed problem, like resource utilization.

3.4. Communication

In a society, for agents to interact, solve the conflicts and negotiate, the system must have a well-defined architecture that allows the communication among agents [18]. The communication allows agents to interchange knowledge, goals, plans or choices, and can involve explicit message passing, or synchronized reads and writes to a blackboard.

- **Blackboard**: A blackboard is a central repository of shared data and knowledge [5]. There is no direct communication between agents, only the blackboard contains the current state of the problem to be solved.
- **Message Passing**: The interchange of information between agents are done through the sending and receiving of explicit messages. The messages have a well-defined common format that all the agents must be aware.

4. The Tri-Coord Model

Tri-coord is a model for conflict resolution in closed societies, with multiple agents [17]. It provides an environment where the laws or interaction rules, the behavior, and the actions are well-defined in the system.

The proposed model is based on social laws and was inspired on the theory of social contract of Jean Jaques Rosseau. The name Tri-coord came from triple coordination, because it uses the political-government organization of our society to build and auto-regulator environment. the agents behavior is implicitly controlled by the environment.

Tri-coord model is based on the tri-partition of the powers and has regular task agents that perform actions for the users, plus three special agents that monitors the environment. The special agents are: i) the executive agent, that is responsible for the conflict management and the communication between agents, it supervises the punishment of the agents that break the rules; ii) the legislative agent, that is responsible for maintaining the laws and rules and for creating new ones, when needed; and iii) the judiciary agent, that is responsible for arbitration, when the conflict persists.
Special agents have the power of balancing the tasks of judging, executing and describing the conflict. The communication base is a blackboard that contains the information to be accessed or modified. The Tri-coord blackboard stores all the dynamic interactions between agents, that are monitored by the special agents.

All the special agents have well-defined tasks in the resolution of conflicts: the executive agent has the task of applying a sanction, the judiciary agent has the task of judging pending situations, and the legislative agent has the task of updating the rules and laws.

The executive agent reads the current status of the task agents and, according to the law database, may fire a sanction. The judiciary agent also reads the current status of the task agents, and may intervene in some predefined situations. The intervention depends on the law database and on the conflict resolution strategies database. Finally, the legislative agent receives the conflict history and updates the conflict resolution strategies database.

5. Distributed-Shared Memory

Distributed-shared memory systems (DSM) provide the convenience of a shared-memory programming model, where all processes have access to a shared address space, on top of a low-cost distributed systems [10]. These systems allow processes to assume a globally shared virtual memory even though they execute on nodes that do not physically share memory. The environment is transparent to the programmer, software mechanisms transform all the shared data accesses in messages over the network.

The implementation of the virtual shared memory in a cluster requires the use of a software distributed shared memory system (SDMS). When a processor wants to access data that belongs to other processor, the SDMS system automatically brings in the data through the network, as shown in Figure 1. As clusters have become the most appealing platform for cost-effective high-performance computing, SDM systems are a low-cost alternative to provide shared data in these platforms.

5.1. The TreadMarks System

The TreadMarks system [1] is a well-known page-based commercially available SDM system. In this work, we used TreadMarks to provide a shared blackboard in a distributed MAS.

TreadMarks is implemented entirely as a user-level library on top of Unix [7]. Initially all the pages are invalid. When a processor tries to write on an invalid page, TreadMarks is called to validate the page by bringing in the necessary modifications to the local copy of the page in the form of diffs (a diff is a run-length encoding of the changes made to a single virtual memory page).

The TreadMarks API is simple but powerful. It provides facilities for process creation and destruction, synchronization, and shared memory allocation. For the agents accesses to the blackboard, synchronization primitives are essential, since when multiple processes are updating a shared data structure, concurrent accesses are not allowed. Using synchronization primitives, the programmer can express ordering constraints between the shared memory accesses of different processes. TreadMarks provides two synchronization primitives: barriers and exclusive locks. The barrier primitive stalls the calling process until all processes in the system have arrived at the same barrier. The exclusive lock primitives allows a process to acquire a lock and to release it, when the critical section finishes.

6. Case Study: Tri-coord/DSM

The idea behind Tri-coord/DSM is to use the TreadMarks system to implement a blackboard for the Tri-coord model. The blackboard is transparently distributed, which means that the agents do not need to control the messages that implement the shared structure, they are controlled by the software DSM system. In this way, each agent accesses the blackboard as it was a local structure. The TreadMarks system is responsible for the distribution of the blackboard information over the memories of the cluster processing nodes, and for the orchestration of the communication. Our proposal tackles two main issues:

- Performance: the blackboard informations are not centralized in one processing node, avoiding the commu-
Figure 2. Example of a collision in the conflict simulator in the instant \( t_2 \).

nication bottleneck.

- Programming Model: is much more easier to write an agent that communicates with other agents, just reading and writing in a shared structure.

The coherence of the distributed blackboard information is guaranteed by the SDSM system. Therefore, the agents perform as they were on a shared memory system.

As the main focus of our work is agents communication, we implemented a conflict generation simulator, in order to evaluate the benefits of distributing the blackboard using TreadMarks. The simulator generates conflicts situations where the agents need to communicate. As the conflict resolution is based on the Tri-coord social law model, our simulator generates situations where the special agents executive and judiciary have to act. As we are interested here only in communication issues, in our simulator, the agents are able to coordinate themselves but are not able to learn.

The basic idea of the conflict simulator is that the task of the computational agents is to move over a bidimensional plane. The agents are objects that move from a random initial coordinate to a random final coordinate. When more agents are inserted, they can collide in the same \((x,y)\) coordinate, in a certain instant of time, as shown in Figure 2. This collision defines a conflict in our simulator.

When a conflict is generated in instant \( t_1 \), the agents have to wait until the conflict is solved by the executive and judiciary agents. The executive and judiciary decide that one of the conflicting agents must remain in the same position and the other must return to the position it was in instant \( t_{i-1} \). After the conflict resolution, the agents become ready again, returning to their tasks, that are moving towards their final coordinate.

6.1. The Distributed Blackboard

The blackboard of the Tri-coord/DSM system contains information concerning to state, position and conflicts. The shared data structures are: Table of Positions; History of Positions; Status List; and Conflict Matrix. These structures store the positions of the agents in the plane; the number of conflicts that a agent has been involved and with which agent the conflict occurred; the status of the agents (active, inactive, suspended and waiting for judgement); and the state of the conflicts in terms of resolution and judgement.

TreadMarks is responsible for distributing all these structures over the cluster processing nodes. All the accesses to the shared structures must be protected by lock/unlock primitives provided by TreadMarks API (\texttt{Tmk-lock-acquire} and \texttt{Tmk-lock-release}). The use of these primitives guarantees that the shared structures are moved, transparently to the agents, from one processing node to other.

6.2. The Task Agent

The role of the task agent is to move over the plane. The agent verifies the status list, and waits until its status is active. When the agent is set active, it calculates its next position and check for the presence of another agent in that position. If there is another agent in the position, the conflict is generated. Otherwise, it moves to the next position and performs the same process again.

6.3. The Executive Agent

The role of the executive agent is to evaluate the conflicts in lower court and solve them (when they occur at first time). The agent keeps searching for unsolved conflicts. When a conflict occurs quite frequently, it will be solved by the judiciary agent.

6.4. The Judiciary Agent

The role of the judiciary agent is to evaluate the conflicts and solve them in last resort. Just like the executive agent, the judiciary agent keeps searching for not solved and not judged conflicts. When it finds one, the judiciary agent solves it in favor of one of the agents.

7. Conclusions

In this work, we proposed the use of distributed-shared memory mechanisms in the implementation of a distributed
and scalable blackboard of a MAS system. The use of these mechanisms allows that MAS systems to be easily implemented on low-cost and distributed architectures like clusters of PCs.

Our implementation used the Tri-coord MAS system as baseline. The Tri-coord system uses a blackboard for the agents communication and is based on social laws to guide the conflict resolution. We used the TreadMarks SDSM system to support the distributed blackboard and implemented a conflict simulator, in order to test our system, by artificially generating conflicts between the agents. So, the conflict simulator created the communications requirements between the agents.

The conflicts were generated in the blackboard and the communication between the processing nodes was handled by TreadMarks. We simulated the system with task agents, a judiciary agent and an executive agent (the legislative agent was not simulated because it is not involved in agents communication). We showed that the distributed memory subsystem is able to handle all the message exchanging between agents in a completely transparent way.

As future work, we intend to compare our system with an implementation of the Tri-coord using a blackboard server. We also intend to evaluate the scalability of the system for a great number of task agents.

References