Multiagent and Grid Systems – An International Journal 7 (2011) 1–20
DOI 10.3233/MGS-2011-0165
IOS Press

Teaching about Madrid: A collaborative agents-based distributed learning course

José Luis Bosque[a], Pilar Herrero[b], Susana Mata[c] and Gonzalo Méndez[d,*]
[a]*Dpto. de Electrónica y Computadores, Universidad de Cantabria, Spain*
[b]*Facultad de Informática, Universidad Politécnica de Madrid, Spain*
[c]*ETSII, Universidad Rey Juan Carlos, Spain*
[d]*I. Tecnología del Conocimiento, Universidad Complutense de Madrid, Spain*

Abstract. Interactive art courses usually require a huge amount of computational resources to run in real time. These computational demands can significantly grow whenever the application is designed to run within a Virtual Environment. This paper studies the viability of combining two previously developed approaches: a Collaborative Awareness Model for Task-Balancing-Delivery (CAMT) in clusters and the "Teaching about Madrid" course, which provides a cultural interactive background of the capital of Spain. The integration of both systems can improve the response times of the interactive course by distributing the rendering tasks among the set of available nodes at a given time. We present some experimental results that show how CAMT efficiently manages the rendering process of the "Teaching about Madrid" course.

Keywords: Task balancing, multi-agent systems, educational virtual environments

1. Introduction

The "Teaching about Madrid" course has been developed with the aim of creating a virtual tour around Madrid. This course is composed of a set of interactive scenarios (see Fig. 1) that are presented in real time by a tour-guide. Students can interact with the scenario, if needed, to get more specific information about a monument, such as the year in which it was built or its historical background. Students can also collaborate with each other to learn together from the projected environment.

Each of the scenarios of the course is projected on a CAVE (see Fig. 2) governed by a cluster of PCs. Each of the projectors of the CAVE has a PC assigned to control it, and all of these PCs are also connected to a high-performance cluster to carry out the complex computational operations that are needed to compute the images to be projected on the CAVE. A high speed myrinet network allows processing all these operations in real time, in order to provide the user with a realistic, immersive experience.

The CAMT model (Collaborative Awareness Model for Task-Balancing-Delivery) is the result of a previous research project and has been used to perform load balancing operations within the Teaching about Madrid course, with the aim of achieving real-time computation.

The design of CAMT is based on the extension and reinterpretation of one of the most successful models of awareness in Computer Supported Cooperative Work (CSCW), called the Spatial Model of

*Corresponding author. Tel.: +34 913947560; Fax: +34 913947547; E-mail: gmendez@fdi.ucm.es.

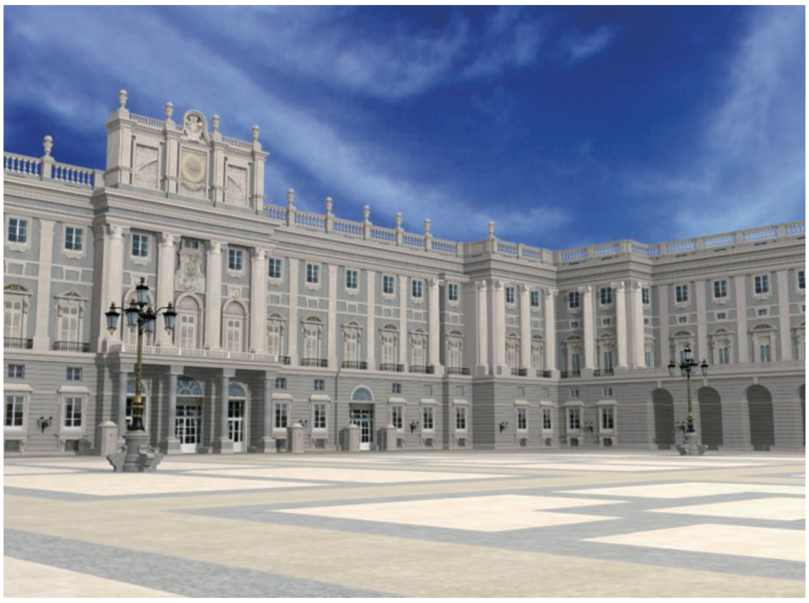


Fig. 1. Palacio Real.

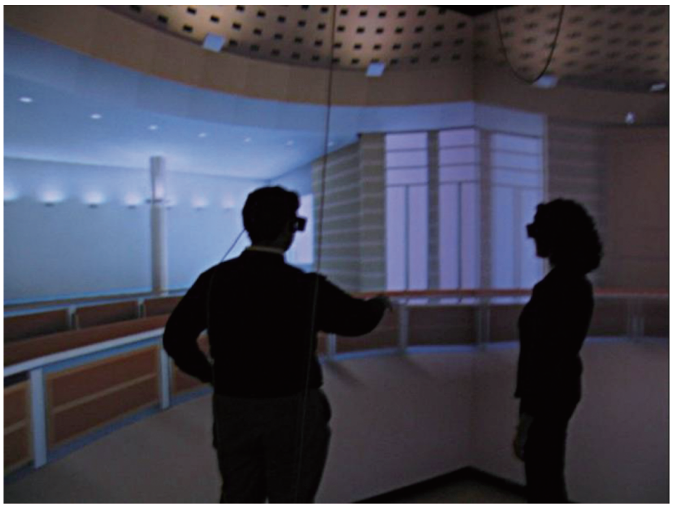


Fig. 2. CAVE.

Interaction (SMI), which manages awareness of interaction in collaborative distributed systems, through a multi-agent architecture, to create a collaborative and cooperative environment. CAMT manages the interaction in the environment, allowing autonomous, efficient and independent task allocation in it.

In addition, the "Teaching about Madrid" course is supported by an Intelligent Tutoring System (ITS), a module that provides tutoring capabilities to the system so that a human tutor needn't be present all the time while the users still get assistance for their learning. The structure of the ITS is also based on a multi-agent architecture, which allows an easier integration of the system's components.

This work presents a novel approach by proposing the combination of agent-based load balancing techniques and intelligent tutoring systems in virtual environments. The CAMT model's algorithms achieve significant improvements with respect to the response time and speedup in comparison with previous approaches to this issue.

2. Related work

In this section we provide an overview of related approaches both to agent-based load balancing and agent-based intelligent tutoring in VEs. As far as our study of related work has shown, there are no significant works that combine agent-based load balancing and intelligent tutoring in virtual environments.

2.1. Load balancing

A taxonomy of load balancing methods has been defined in [3], taking into account different aspects. Three important criteria for this classification are:

- *Time in which workload distribution is performed*: According to this aspect, load balancing algorithms can be classified as *static* [6,15] or *dynamic* [11,12].
- *Control*: With respect to this feature the algorithm can be labelled as *centralized* [10] or *distributed* [6].
- *System state view*: Depending on the view of the system state, load balancing algorithms can be labelled as *global* [6] or *local* [4].

Depending on the kind of clusters, e.g., homogeneous or heterogeneous, these rules can be applied in a different way. Another alternative is presented in [19], which defines a generic and scalable architecture for the efficient use of resources in a cluster based on CORBA.

DASH (Dynamic Agent System for Heterogeneous) [17] is an agent-based architecture for load balancing in heterogeneous clusters. Three types of agents are defined: (i) monitoring agent, which is responsible for characterizing the local node and implementing the information policy; (ii) process execution agent, which is responsible for the task execution, both local and remote and (iii) process scheduler, which is responsible for making decisions about the load balancing and scheduling policies. The most noticeable characteristic of this proposal is the definition of a collaborative awareness model, used for providing global information that helps establish a suitable load balance. Unlike this work, our proposal (CAMT) extends and reinterprets one of the most successful models of awareness, the Spatial Model of Interaction (SMI [2]), which manages awareness of interaction through a set of key concepts. Most of the agent-based load balancing systems use mobile agents, which makes the migration of tasks easier [7,16,18].

Nevertheless, the study published in [14] concludes that the task migration only obtains moderate benefits for long duration tasks. And even in this case, the average performance of load balance algorithms does not improve by using task migration.

2.2. Agent-based intelligent tutoring

There are several projects aiming at the use of VR for education and training supported by intelligent agents. The first ones were developed over a decade ago, and the most representative among them are Adele [22], Steve [23], Herman the Bug [24], Cosmo [25] and Vincent [26]. What all of them have in common is the fact that the primary objective in all of them was to develop an embodied pedagogical agent to support education and training. Each of them tried to solve some of the problems that this emerging discipline posed.

None of these systems are structured as multi-agent systems, but as a single agent that inhabits a particular virtual world, and each of them exhibits its own internal architecture. Even so, they have been the key to identify some of the issues that researches are still trying to solve in a satisfactory way.

There are some examples of multi-agent systems that support education and training without using VEs. That is the case of FILIP, a multi-agent system for training based on simulations [27] to provide training for air controllers. The system is composed of seven agents that cover the modules of an ITS: one for the student, one for the expert, three for the tutor (skill development, curriculum and instructor agents) and two other agents related to the communication with the learning environment and the user.

Baghera is another example of multi-agent system used to teach geometry [28]. The aim of this system is to study emergent behaviours in multi-agent systems. What makes this system more interesting is the fact that agents are organized in two levels, and the number of agents is not fixed, but varies according to the number of students connected to the system. Each student is assisted by three agents: the personal interface agent, which monitors the student's actions, the tutor agent and the mediator agent. In addition, the tutor is assisted by two agents: the personal interface agent and the assistant agent. All these agents are supported by second level agents of four different kinds, which are in charge of evaluating the student's actions through a voting mechanism.

The systems that are closer to the one described in this paper are those that are based on multi-agent systems and make use of VEs to support training. A good example is MASCARET (Multi-Agent Systems to simulate Collaborative, Adaptive and Realistic Environments for Training) [29], an agent-based system that has been used to train firemen in operation management. In this system, agents are divided in organizations, each of which controls different aspects of the organization: physical, social, pedagogical, mediation, and human interaction. The agents that integrate the pedagogical organization cover the four modules of an ITS, plus a fifth module that is in charge of controlling the mistakes an student may make. The expert agent communicates with the social and physical organizations to be able to know what to do and what objects and agents are involved in an action.

Lahystotrain is an application developed to train surgeons in laparoscopy and hysteroscopy interventions [30]. This system contains five agents which help the student in the training process. One of them is the tutor, which supervises the student and registers his actions. An assistant agent provides explanations and interrupts him when he makes a mistake. These agents have an ad-hoc architecture tailored to suit their responsibilities. The other three agents take the role of an auxiliary surgeon, a nurse and an anaesthetist that play their role within the team. The architecture is the same for all three, and it is a kind of BDI architecture with a perception module, a reasoning engine and an action control module. The student must learn what the role of these three agents is and how to coordinate them.

What most of these systems have in common is the fact that they have been developed to solve a specific problem, but only a few of them have been designed to be reusable (i.e. [23]). The ITS architecture described in this paper follows this design objective, so that it can be adapted to different domains.

3. Reinterpreting the SMI key concepts

The Spatial Model of Interaction (SMI) [2] is based on a set of key concepts which are abstract and open enough to be reinterpreted in many other contexts with very different meanings [9]. The model itself defines five linked concepts: medium, focus, nimbus, aura and awareness.

3.1. Medium

A prerequisite for useful communication is that two objects have a compatible medium in which both objects can communicate.

3.2. Aura

The sub-space which effectively bounds the presence of an object within a given medium and which acts as an enabler of potential interaction [5].

In each particular medium, it is possible to delimit the observing object's interest. This idea was introduced by S. Benford in 1993 [2], and it was called *Focus*. In the same way, it is possible to represent the observed object's projection in a particular medium, called *Nimbus*.

3.3. Awareness

It quantifies the degree, nature or quality of interaction between two objects. Awareness between objects in a given medium is manipulated via *Focus* and *Nimbus*, requiring a negotiation process. Considering, for example, A's awareness of B, the negotiation process combines the observer's (A's) focus and the observed's (B's) nimbus.

For a simple discrete model of focus and nimbus, there are three possible classifications of awareness values when two objects are negotiating unidirectional awareness [9]: *Full awareness, Peripheral awareness and No awareness.*

Let's consider a system containing a set of nodes $\{\mathrm{n}_i\}$ and a task T that requires a set of processes to be solved in the system. Each of these processes necessitates some specific requirements being r_i the set of requirements associated to the process p_i, and therefore each of the processes will be identified by the tuple $(\mathrm{p}_i, \mathrm{r}_i)$ and T could be described as $\mathrm{T} = \sum_i \{(\mathrm{p}_i,\mathrm{r}_i)\}$

The CAMT model reinterprets the SMI key concepts as follow:

3.4. Focus

It is interpreted as the subset of the space on which the user has focused his attention with the aim of interacting with.

3.5. Nimbus

It is defined as a tuple (Nimbus = (*NimbusState, NimbusSpace*)) containing information about: (a) the load of the system in a given time (*NimbusState*); (b) the subset of the space in which a given node projects its presence (*NimbusSpace*). As for the *NimbusState*, this concept will depend on the processor characteristics as well as on the load of the system in a given time. In this way, the *NimbusState* could have three possible values: *Null, Medium* or *Maximum* (see Section 4).

3.6. Awareness of Interaction (*AwareInt*)

This concept will quantify the degree, nature or quality of asynchronous interaction between distributed resources. Following the awareness classification introduced by Greenhalgh in [8], this awareness could be *Full*, *Peripheral* or *Null.*

$$AwareInt(n_i, n_j) = Full \text{ if } n_j \in Focus(\{\mathrm{n}_i\}) \quad \wedge \quad n_i \in Nimbus(n_j)$$

Peripheral aware of interaction if

$$AwareInt(n_i, n_j) = Peripheral \text{ if } \begin{cases} n_j \in Focus(\{n_i\}) \wedge n_i \notin Nimbus(n_j) \\ or \\ n_j \notin Focus(\{n_i\}) \wedge n_i \in Nimbus(n_j) \end{cases}$$

The CAMT model is more than a reinterpretation of the SMI, it extends the SMI to introduce some new concepts such us:

3.7. Interactive pool

This function returns the set of nodes $\{n_j\}$ interacting with the n_i node in a given moment. If $AwareInt\ (n_i, n_j) = Full$ then $n_j \in InteractivePool(n_i)$

3.8. Task resolution

This function determines if there is a service in the node n_i, being NimbusState(n_i) $\neq$ Null, such that could be useful to execute the task T (or at least one of its processes).

The Task Resolution concept also complements the Nimbus concept, because the *NimbusSpace* will determine those machines that can be taken into account in the task assignment process because they are not overloaded yet.

3.9. Collaborative organization

This function takes into account the set of nodes determined by the *Interactive Pool* function and will return the nodes of the system in which it is more suitable to execute the task T (or at least one of its processes p$_i$). This selection will be made by means of the *TaskResolution* function.

4. The load balancing algorithm in CAMT

In this section we present the load balancing algorithm as it has been introduced in the CAMT awareness model, and how it has been applied to our distributed and collaborative multi-agent architecture in the cluster. The main characteristics of this algorithm are that it is dynamic, distributed, global and takes into account the system heterogeneity. This algorithm contains the policies described below [13].

4.1. State measurement rule

It is in charge of getting information about the computational capabilities of the node in the system. This information, quantified by a load index, provides awareness of the NimbusState of the node. Several authors have proposed different load indexes and they have studied their effects on the system performance [9]. However, as for the CPU utilization, we are especially interested in the computational capabilities of the node for the new task to be executed. Regarding this aspect, several factors have to be taken into account, because this concept depends not just on the number of tasks to be executed but also on the CPU's use for each of these tasks. In this research work the concept of CPU assignment is used to determine the load index. The CPU assignment, A$_{CPU}$, is defined as the CPU percentage that can be assigned to a new task to be executed in the node N$_i$. The calculation of this assignment is based on two dynamic parameters: the number of tasks N, which are ready to be executed in the CPU queue and the percentage of occupation of the CPU, U, and it would be calculated as:

$$If\left(U \geqslant \frac{1}{N}\right) => A_{CPU} = \frac{1}{N+1}$$

$$If\left(U < \frac{1}{N}\right) \Rightarrow A_{CPU} = 1 - Usage$$

The CPU assignment is normalized by the maximum computational power of the cluster nodes to consider the heterogeneity of the cluster.

$$I_{=}\frac{P_i \cdot A_{CPU}}{P_{MAX}}$$

The NimbusState of the node will be determined by the load index. This state determines if the node could execute more (local or remote) tasks. Its possible values are:

- *Maximum*: The load index is low and therefore this infrautilized node will execute all the local tasks, accepting all new remote execution requests coming from other nodes.
- *Medium*: The load index has an intermediate value and therefore the node will execute all the local tasks, interfering in load balancing operations only if there are not other nodes whose NimbuState would be Maximun in the system.
- *Null*: The load index has a high value and therefore the node is overloaded. In this situation, the node will not execute new tasks, the computational capabilities available are very low and for that reason it will reject any request of new remote execution.

The NimbusState of the node depends on the load index value and an increase or decrease of this index over a specific threshold will imply the corresponding modification in the NimbusSate.

4.2. *Information exchange rule*

The knowledge of the global state of the system will be determined by a policy on the information exchange. This policy should keep the information coherence without overloading the network with an excessive number of unnecessary messages.

An optimum information exchange rule for the CAMT model should be based on events [1]. This rule only collects information when a change in the Nimbus of the nodes is made. If later, the node that has modified its nimbus will be in charge of notifying this modification to the rest of the nodes in the system, avoiding thus synchronisation points. As this algorithm is global, this information has to be sent to all the nodes in the system.

4.3. *Initiation rule*

As the model implements a non user interruption algorithm, the selection of the node must be made just before sending the task execution. The decision of starting a new load balancing operation is completely local, it depends on the local information storage. When a node intends to throw the execution of a new task, the initialization rule will evaluate:

- If (NimbusState = Maximum) or (NimbusState = Medium), the task is accepted to be executed locally.

– If (NimbusState = Null), a new load balancing operation is started.

An advantage of this sketch is that it allows to control the degree of imbalance in the system, by adjusting the width of the Medium's range (NimbusState = Medium). As the load balancing operation takes place between the node in which NimbusState is Null and another in which NimbusState is Maximum, the difference in the load index of the nodes has to be, at least, equal to the width of Medium's range. If this value is very high, a bigger imbalance is allowed in the system, reducing therefore the number of load balancing operations in the system. A limit situation happens when this value is equal to zero. If later, all the nodes will be perfectly balanced.

Finally, another important issue is how to control the minimum CPU assignment. When a node gets a state such that its NimbusState is equal to Null, it can not receive more tasks to be executed. This means that the emissor threshold has a fixed value, and the minimum percentage of CPU assigned for a new task as well.

4.4. Load balancing operation

Once the node has made the decision of starting a new load balancing operation, this operation will be divided in three different rules: localization, distribution and selection.

The localization rule has to determine which nodes are involved in the *CollaborativeOrganization* of the node n_i. In order to make it possible, the CAMT model needs to determine the awareness of interaction of this node with those nodes inside its focus. To optimize the implementation, the previous awareness values are dynamically updated based on the information exchange rule. Those nodes whose awareness of interaction with n_i was Full will be part of the Interactive Pool of n_i to solve the task T, and from that pre-selection the TaskResolution method will determine those nodes that are suitable to solve efficiently the task in the environment.

This algorithm joins selection and distribution rules because it determines which nodes (among all the nodes constituting the *CollaborativeOrganization*) will be in charge of executing each of the processes making up the T task. The proposed algorithm takes into account the *NimbusState* of each of the nodes as well as the *TaskResolution* to solve any of the T's processes.

The goal of this algorithm is to find the most equilibrated processes assignment to the computational nodes, making up the cluster, based on a set of heuristics. This spread is made in an iterative way. Firstly, a complete distribution of the processes making up the T task is made in the computational nodes implicated in the *CollaborativeOrganization.* If, in this first turn, all the process were assigned to one of the nodes involved in the *CollaborativeOrganization*, the algorithm would finish. Otherwise, it would be necessary to calculate the *NimbusState* of the nodes belonging to the *CollaborativeOrganization*, repeating the complete process again.

5. The underlying architecture of CAMT

The load balancing multi-agent architecture is composed of four agents: the Load Agent (LA), the Global State Agent (GSA), the Initiation Agent (IA), the Load Balancer Agent (LBA) (see Fig. 3). All these agents are replicated for each of the nodes of the cluster.

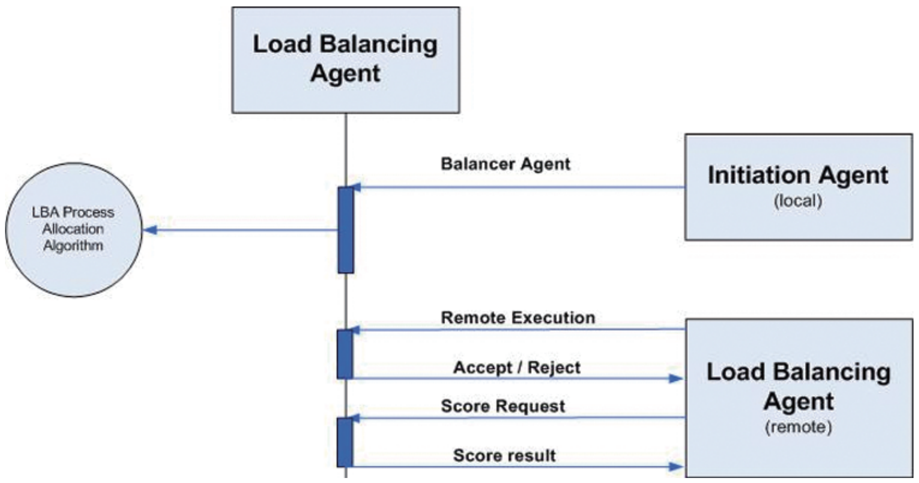


Fig. 3. CAMT underlying architecture.

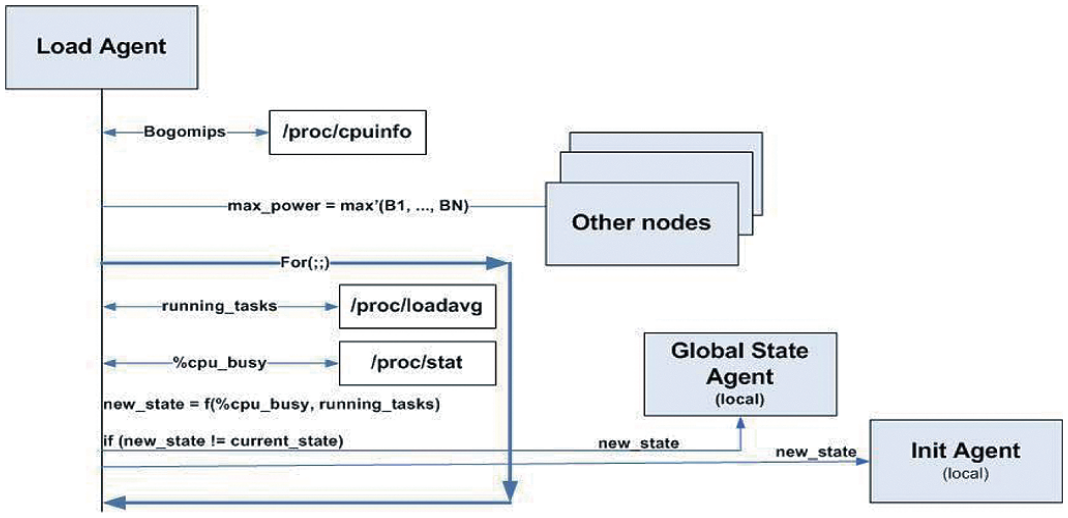


Fig. 4. The load agent.

5.1. *The load agent*

The Load Agent (see Fig. 4) has, as its main function, to calculate, periodically, the load index of the local node and evaluate the changes on its state. Moreover, it defines the thresholds determining the changes on its state for that node. When it detects a change on the state, this modification is notified to the local GSA and IA.

The first step of the LA is to obtain the static information, i.e. the node computational power. The computational power P_i is represented by a parameter of the operating system, named *Bogomips*, which is calculated by the kernel, when the system starts, as the average of the execution time of a set of instructions. This value is read from the */proc/cpuinfo* file. Then this information is communicated to the rest of the nodes through the MPI_Reduce function, which is in charge of calculating the maximum of the computational power of all the nodes composing the cluster.

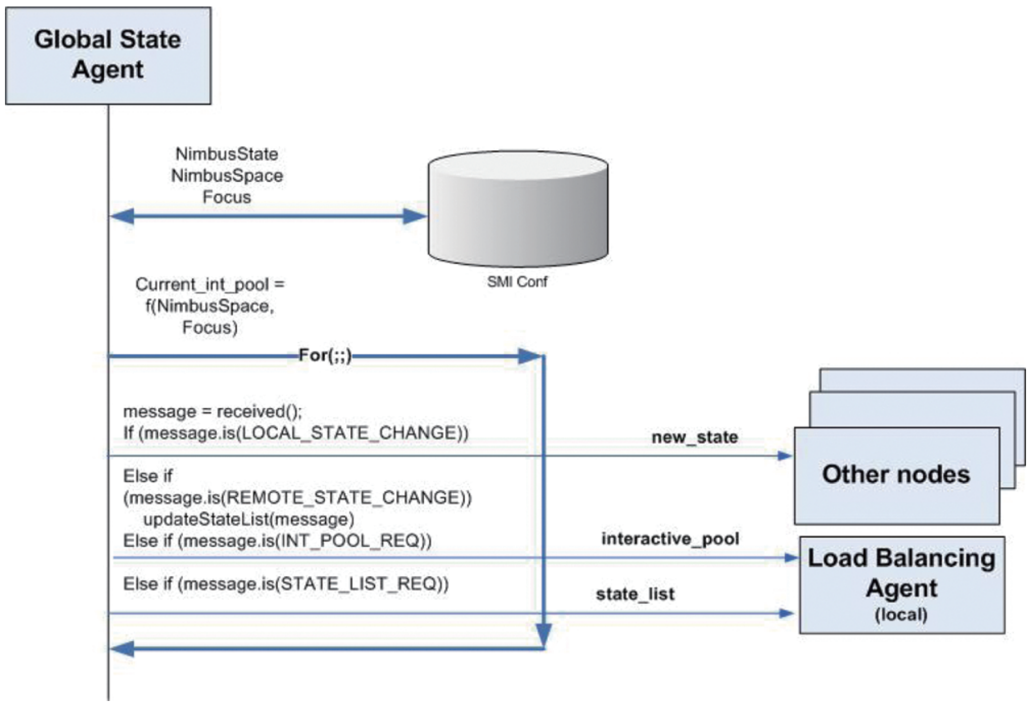

Global State Agent
NimbusState
NimbusSpace
Focus
SMI Conf
Current_int_pool = f(NimbusSpace, Focus)
For(;;)
message = received();
If (message.is(LOCAL_STATE_CHANGE))
new_state
Other nodes
Else if (message.is(REMOTE_STATE_CHANGE))
updateStateList(message)
Else if (message.is(INT_POOL_REQ))
interactive_pool
Load Balancing Agent (local)
Else if (message.is(STATE_LIST_REQ))
state_list

Fig. 5. The global state agent.

Next, the agent enters in an infinite loop until the application ends. The first step in this loop is to get dynamically information about the load of the node. This information is composed by the number of running task and the CPU usage. The number of tasks that are ready to be executed is in the */proc/loadavg* file, and the percentage of CPU free can be obtained from the file */proc/stat*.

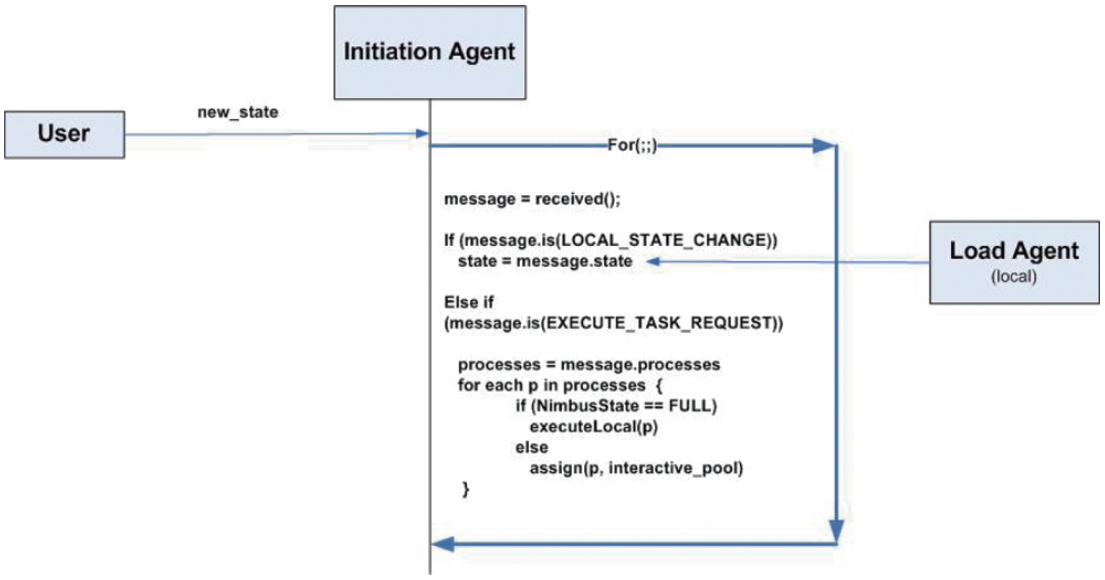
Then, the new state of the node is calculated based on the previous information. With the new state, the agent determines if a node state change has occurred. If so, the agent communicates it to the local GSA and IA. Finally, the agent sleeps a time span, defined as a parameter by the user.

5.2. The global state agent

This agent implements the exchange information rule, and therefore its main functionality is to manage the state information exchanged among the nodes of the system, and provide the LBA with this information as soon as it requires it.

Firstly, when the agent starts, it gets information about its focus, its nimbusSpace and its NimbusState. Once this information is communicated to the rest of the nodes, it determines the current InteractivePool. Next, the agent enters in an infinite loop in which it is waiting to receive messages from other agents, and therefore, from this moment, its functionality depends on the kind of message that it receives. These messages are (see Fig. 5):

– LOCAL_STATE_CHANGE: This message comes from the local LA and this information has to be notified to all the Global State Agent that are located in a different node of the cluster to update their lists.
– REMOTE_STATE_CHANGE: In this case, only the local state list should be modified to update the new state of the remote node.



```
message = received();

If (message.is(LOCAL_STATE_CHANGE))
  state = message.state

Else if
(message.is(EXECUTE_TASK_REQUEST))

  processes = message.processes
  for each p in processes {
          if (NimbusState == FULL)
            executeLocal(p)
          else
            assign(p, interactive_pool)
  }
```

Fig. 6. The initiation agent.

- INTERACTIVE_POOL_REQUEST: The local LBA request the InteractivePool to the GSA. The GSA responds to this request providing it with the required information.
- STATE_LIST_REQUEST: the local LBA request the state list that the GSA agent keeps updated with the state of all the nodes composing the cluster. The GSA responds to this request providing it with the required information.

5.3. The initiation agent

This agent is in charge of evaluating the initialisation rule. When a user intends to execute a task in a node of the cluster, this request is sent to the IA of that node. Once this agent has evaluated the initialisation rule, it determines if that can be executed locally or if a new load balancing operation has to be carried out (see Fig. 6).

As for the rest of the agents, its main structure contains an infinite loop and, for each of these iterations, the pending tasks in the execution queue are checked. If there is a pending task, a new assignment task loop starts. There are two type of messages:

- LOCAL_STATE_CHANGE: It receives a message from the local LA to notify a change on the local state.
- EXECUTE_TASK_REQUEST: It requests the execution of a new task. As a task is composed by a set of processes, the local execution of one of these processes can change the nimbusState of that node. This is the reason why, when an execution request is received, the IA starts a loop to assign all the processes of the task. For the first process, the NimbusState is checked to corroborate if its value is equal to Maximum. If so, the node is infra-utilised and therefore that process is executed locally. The same procedure is repeated for the following process. This loop finishes when all the processes have been executed or when its nimbusState has changed its value. In that moment, a new balancing operation starts and a message is sent to the local LBA.

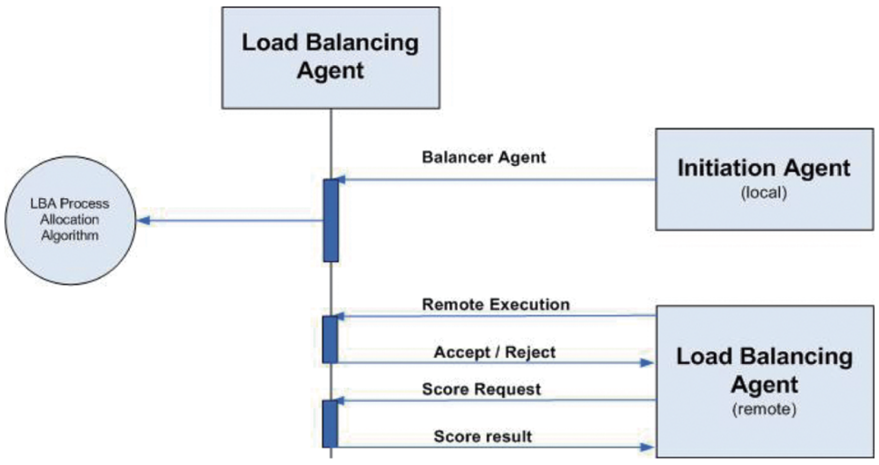


Fig. 7. The load balancing agent.

5.4. *The load balancing agent*

This agent is responsible for making the load balancing operation, strictly speaking. Its structure contains an infinite loop that is waiting to receive messages from other agents (see Fig. 7). Its functionality depends on the messages received, being the possible messages:

– BALANCER EXECUTION: This message comes from the local IA and it indicates that a new load balancing operation needs to start. The LBA executes the complete sequence of steps introduced by the CAMT model to find the most suitable nodes and to assign them the most suitable process. For the localization rule, the LBA follows the following sequence of steps:

1. Request the InteractivePool and the states list to the local GSA.
2. Determine the TaskResolution, analysing which nodes of the InteractivePool have their nimbusState different to Null.
3. Request the score of those processes composing the task to be executed to the LBA of the nodes included in the TaskResolution.
4. Taking into account the TaskResolution and the requested scores, determining the Collaborative_Organization by analysing those nodes that, belonging to the TaskResolution, can execute at least one of the processes of the task.

As for the selection and distribution rule, once the Collaborative_Organization has been made up, it is necessary to determine which processes are sent to each of the nodes of the cluster. In order to make this possible, the algorithm presented in Section 4 has been implemented. Once all the processes have been assigned, they are sent to the designated nodes. These nodes reply with an acceptance or rejection message. If the process is accepted by the node, the assignment of the process ends. Otherwise, the process is pending of assignment and it is added to the new task described in the previous paragraph.

– REMOTE_EXECUTION: The message received comes from the remote LBA, asking for the remote execution of a process. Once the LBA has checked its own state, it replies to the remote LBA with an acceptance or rejection message. If the process is accepted, the operation concludes, the LBA executes the process locally and it updates its state. The rejection can be due to a change on its nimbusState (to Null) which has not been notified yet because of the network latency.

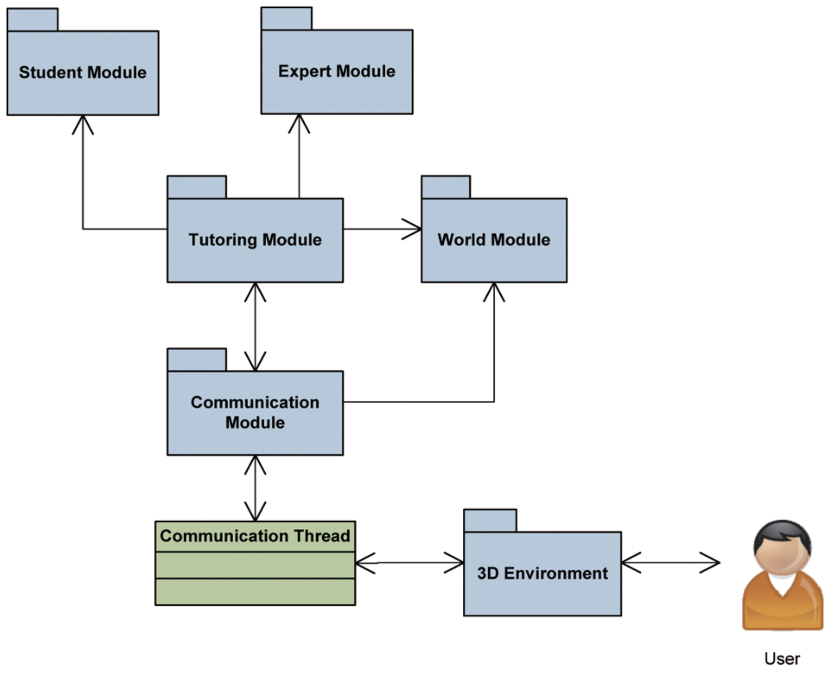


Fig. 8. Extended ITS.

– SCORE_REQUEST: This message is a request to the LBA to send the scores of a specific task. The LBA receives the task and it evaluates the scores for each of the processes belonging to that task. These scores have to be calculated in real time, as they can change dynamically. Once they have been calculated, the LBA sends the current scores to the remote LBA to execute the selection task.

6. The "Teaching about Madrid" architecture

Our approach to the definition of an architecture for the *Teaching about Madrid* course is based on the one defined in [20], a multi-agent architecture for Intelligent Virtual Environments for education and training which uses the traditional structure of an Intelligent Tutoring System (ITS) [21], but extended with a new module to support the use of multi-user Virtual Environments (see Fig. 8).

The system's architecture has five agents corresponding to the five key modules of the extended ITS architecture:

– A *Communication Agent*, which is in charge of communicating the ITS with external users and software.
– A *Student Modelling Agent*, which manages the user modelling tasks.
– A *World Agent*, which is in charge of maintaining relevant information about the VE.
– An *Expert Agent*, which maintains information about the subject to be taught (in this case, about the city of Madrid).
– A *Tutoring Agent*, which controls the interaction between the user and the ITS in terms of tutoring events.

Analyzing the responsibilities of these agents, some additional roles can be identified that point to the creation of new, subordinate agents that can carry them out, subsequently giving rise to a hierarchical multi-agent architecture.

6.1. Communication agent

The *Central Communication Agent* is responsible for the communication between the Virtual Environment and the Tutoring System. It delegates part of its responsibilities to a set of individual *Communication Agents* dedicated to each student. There is also a *Connection Manager Agent*, which is responsible for coordinating the connections of the students to the system, and a set of *Device Agents* in charge of managing the data provided by the devices the students use to interact with the Virtual Environment.

6.2. Student modeling agent

This agent is in charge of maintaining a model of each student, including personal information, their actions in training sessions, and a model of the students' knowledge.

Figuring out the student's abilities and beliefs/knowledge is usually not a trivial issue. To better individualize training and appropriately understand the student's behaviour, a representation of some of its personal features (personality traits, mood or attitudes) is defined and maintained. To do this, the *Student Modeling Agent* is assisted by:

– A *Historic Agent*, which is responsible for registering the history of interactions among the students and the system.
– A *Psychological Agent*, which is responsible for building a psychological profile of each student including their learning style, attentiveness, and other personality traits, moods and emotions that may be interesting for adapting the teaching process.
– A *Knowledge Modeling Agent*, which is responsible for building a model of the student's current knowledge and its evolution.
– A *Cognitive Diagnostic Agent*, which is responsible for trying to determine the causes of the student's mistakes.

6.3. World agent

The *World Agent* is in charge of maintaining a coherent model of the VE, so that all the agents and students have the same information about the state of the world.

The *World Agent* is related to:

– The *3D Geometrical Information Agent* which has geometrical information on the objects and the inhabitants of the world. Among other responsibilities, this agent will answer questions about the location of the objects.
– The *Objects and Inhabitants Information Agent*, which has semantic knowledge about the objects and the inhabitants of the world.
– The *Path-Planning Agent*, which is capable of finding paths to reach a destination point in the environment avoiding collisions with other inhabitants and objects. With the aim of finding paths, the A* algorithm is applied to a graph model of the environment.

6.4. Expert agent

The *Expert Agent* contains the expert knowledge about the environment that is being taught, as well as the expert knowledge necessary to solve the problems posed to the student and to reach the desired goals.

The *Expert Agent* delegates some of its responsibilities to a *Simulation Agent*, that contains the knowledge about the simulated system, and a *Planning Agent*, that is able to find the best sequence of actions to solve different activities.

6.5. Tutoring agent

It is responsible for proposing activities to the students, monitoring their actions in the virtual environment, checking if they are valid or not with respect to the plan worked out by the *Expert Agent*, and making tutoring decisions. The activities that can be proposed by the *Tutoring Agent* are dependent on the particular environment that is being simulated in the VE, and they can be defined by means of an authoring tool.

The adaptation of the tutoring strategy to every particular student may also encompass how the virtual tutor behaves: a student may need a tutor with a particular character (e.g., training children may require a funny, enthusiastic tutor), or with a specific mood (e.g., if a student does not pay much attention for too long, a disgusted tutor may be effective). Poor or upsetting tutor behaviors will lead to a lack of believability, reducing the student's feeling of presence and the effectiveness of the training.

7. MAS integration

Since the presented application is supported by two different multi-agent systems, some degree of integration has been necessary in order for the Teaching about Madrid course to behave correctly.

At this stage, the integration has been made at a technological level, so that both MAS are implemented using the same platform, but they basically run as separated systems. This decision has been made in order for the ITS not to interfere with CAMT, since our first objective was to evaluate how CAMT could improve the rendering performance (see Section 8).

Once this has been tested, there are several aspects that are likely to be improved using CAMT, being the most important of them the planning task that has to be performed by the Planning Agent, since planning is a task that typically presents an exponential complexity.

Therefore, both MAS run in different nodes, and each of them has its own directory, communication channels and the like. For the moment, the rendering task performed by CAMT is only affected by the rendering effects caused by the decisions that the ITS makes, and which may have an impact on what the user sees.

8. CAMT evaluation on the "Teaching about Madrid course"

One of the goals of the "Teaching about Madrid" course is focused on the generation of realistic renderings of complex 3D models. The visualization of this type of scenarios within a CAVE environment must be accomplished at certain frame rates in order to obtain interactivity and inmmersion sensations.

The CAVE consists of 4 projectors, each of which is connected to a different PC that is in charge of the rendering of the scenario from a different point of view. The visualization is performed using an

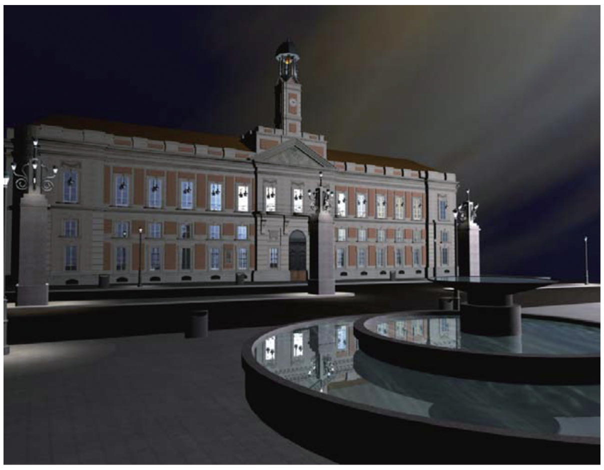


Fig. 9. Puerta del Sol of Madrid.

active stereo system with projectors that work at 100 Hz, meaning that these PCs ideally should be able to generate 100 images per second. However, the high complexity of the geometrical model and realistic rendering techniques, together with other highly demanding computations, such as collision detection, can overflow the computational capacity of these PCs. In this case, users can perceive a gap between two consecutives images and therefore the scenario's realism and the user's feeling of immersion decreases considerably.

The fact that the rendering task can be split up in several processes which can be executed independently, makes the CAMT model be a feasible approach for improving the "Teaching about Madrid" performance through the execution of the rendering task in a high-performance cluster.

The selected cluster is made up of 40 PCs (nodes) cluster connected through a 1.1 Gbps Myrinet Network. Each cluster node is a 2 GHz AMd K7 processor with 512 MB of main memory and 256 KB of cache memory. The PC operating system is Red Hat Linux 7.3. The CAMT model has been developed using GNU tools and LAM/MPI 7.1.1 Library.

The integration of the CAMT model in order to improve the performance of the rendering stage can also introduce some penalization due to the load balancing tasks. This overhead has been evaluated for three scenarios of different geometrical complexity: Teatro Real (see Fig. 10), Puerta del Sol (see Fig. 9) and Palacio Real (see Fig. 14).

Figures 11, 12 and 13 show the results obtained evaluating the overhead introduced by CAMT while it assigns the processes to the nodes of the cluster.

As it can be observed, the overhead incurred by the algorithm to assign a process doesn't interfere with the frame rate of the CAVE's projectors. The overhead remains almost constant for all of the tasks and processes even though it increases as the geometrical complexity of the scenario – and therefore the data file size – also increases, demonstrating that the CAMT algorithm has been endowed with very strong scalability features.

9. Conclusions

This paper presents the integration of two previous research works. The first of these two projects is an agent-based guided course, named "Teaching about Madrid", which is intended to provide students with

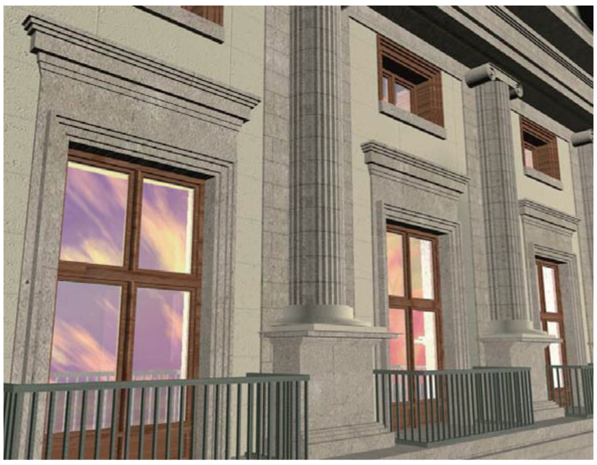


Fig. 10. Teatro Real.

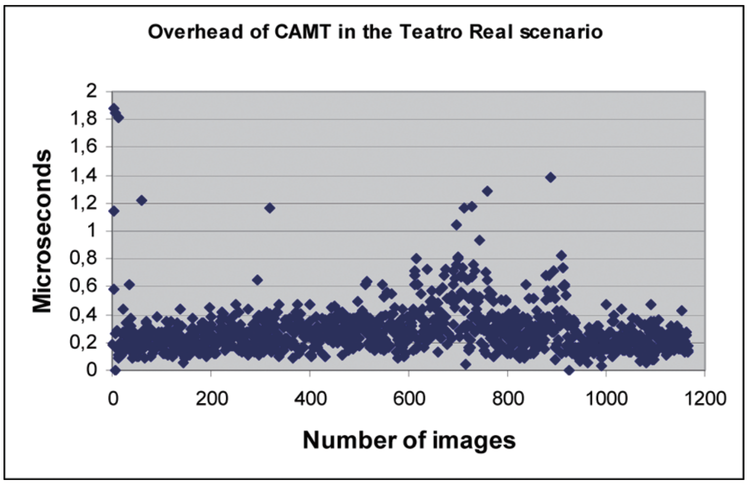


Fig. 11. CAMT overhead in the Teatro Real.

a cultural interactive background of Madrid. The second one, CAMT, manages awareness of interaction in collaborative distributed systems, through a multi-agent architecture, to allow autonomous, efficient and independent task allocation in the environment.

The CAMT model complements the "Teaching about Madrid" course as it selects the best processor to make the complex render task of each of the images of the course in the cluster. CAMT divides the render task of each of these images into a set of independent processes which are assigned to the most suitable nodes in the cluster. Thus, even though the geometrical model and the illumination algorithms are complex, practically none of the images are lost, and users don't perceive a gap between consecutives images, feeling a high degree of realism and immersion.

The integration of both systems, from a technological point of view, shows that the "Teaching about Madrid" course has greatly benefitted from such an integration, and so further integration will be carried out in order to improve the efficiency of the ITS.

Finally, the experimental results presented in this paper demonstrate that the overhead incurred by

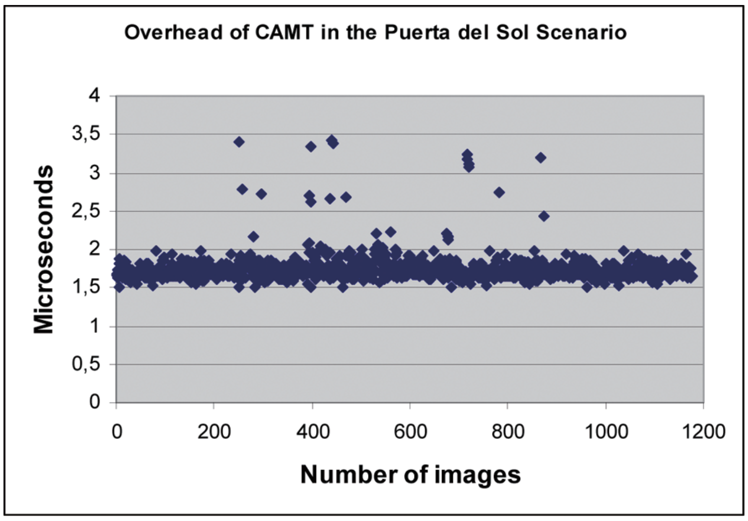


Fig. 12. CAMT overhead in the Puerta del Sol.

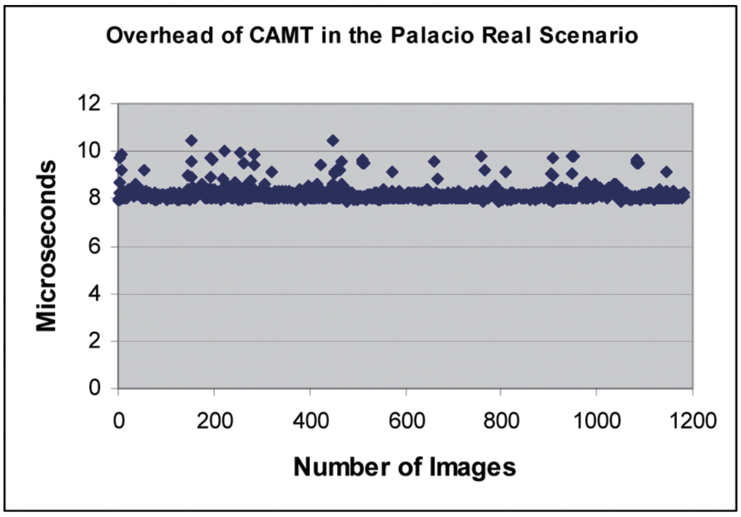


Fig. 13. CAMT overhead in the Palacio Real.

the algorithm to assign a process doesn't interfere with the frame rate of the CAVE's projectors, and therefore we can conclude that CAMT complements successfully the teaching course.

References

[1] M. Beltrán, J.L. Bosque and A. Guzmán, *Resource Disseminatioin policies on Grids*. On the Move to Meaningful Internet Systems 2004: OTM 2004. Lectures Notes in Computer Science. Springer-Verlag Heidelberg, October 25–29, 2004, pp. 135–144.

[2] S.D. Benford and L.E. Fahlén, A Spatial Model of Interaction in Large Virtual Environments. Published in Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93). Milano, Italy, Kluwer Academic Publishers, 1993, pp. 109–124.

[3] T.L. Casavant and J.G. Kuhl. "A taxonomy of scheduling in general-purpose distributed computing systems", *Readings and Distributed Computing Systems*, 1994, pp. 31–51.

[4] A. Corradi, L. Leonardi and F. Zambonelli, Diffusive load-balancing policies for dynamic applications, *IEEE Concurrency* **7**(1) (1999), 22–31.

[5] S. Darbha and D.P. Agrawal, Optimal scheduling algorithm for distributed-memory machines, *IEEE Transactions on Parallel and Distributed Systems* **9**(1) (January 1998), 87–95.

[6] S.K. Das, D.J. Harvey and R. Biswas, Parallel processing of adaptive meshes with load balancing, *IEEE Trans on Parallel and Distributed Systems* **12** (2001), 1269–1280.

[7] S. Desic and D. Huljenic, Agents based load balancing with component distribution capability, *In Proceedings of the 2nd International Symposium on Cluster Computing and the Grid* (*CCGRID'02*), 2002.

[8] C. Greenhalgh, *Large Scale Collaborative Virtual Environments*, Doctoral Thesis. University of Nottingham. October 1997.

[9] T. Kunz, The influence of different workload descriptions on a heuristic load balancing scheme, *IEEE Transactions on Software Engineering* **17**(7) (July 1991), 725–730.

[10] Bettina Schnor, Stefan Petri, Reinhard Oleyniczak, and Horst Langendorfer. *Scheduling of Parallel Applications on Heterogeneous Workstation Clusters*. In Proceedings of the ISCA 9th International Conference on Parallel and Distributed Computing Systems, volume 1, September 1996, pp. 330–337.

[11] J. Watts and S. Taylor, A practical approach to dynamic load balancing, *IEEE Transactions on Parallel and Distributed Systems* **9**(3) (March 1998), 235–248.

[12] L. Xiao, S. Chen and X. Zhang, Dynamic cluster resource allocations for jobs with known and unknown memory demands, *IEEE Trans. on Parallel and Distributed Systems* **13**(3) (March 2002), 223–240.

[13] C. Xu and F. Lau, Load balancing in parallel computers: theory and practice. Kluwer Academic Publishers, 1997.

[14] W. Leland and T. Ott, Load-balancing heuristics and process behavior, *ACM SIGMETRICS*, 1986, pp. 54–69.

[15] G. Maniraman and C.S.R. Murthy, An efficient dynamic scheduling algorithm for multiprocessor real-time systems, *IEEE Transactions on Parallel and Distributed Systems* **9**(3), March 1998.

[16] W. Obelöer, C. Grewe and H. Pals, Load Management with Mobile Agents, p. 21005, 24th. EUROMICRO Conference Volume 2 (EUROMICRO'98), 1998.

[17] A. Rajagopalan and S. Hariri, An Agent Based Dynamic Load Balancing System, *Proceedings of the International Workshop on Autonomous Decentralized Systems*, 2000, pp. 164–171.

[18] N. Suri, P.T. Groth and J.M. Bradshaw, While You're Away: A System for Load-Balancing and Resource Sharing Based on Mobile Agents CCGRID 2001, pp. 470, 1st International Symposium on Cluster Computing and the Grid, 2001.

[19] S. Vanhastel, F. De Turck and P. Demeester, Design of a generic platform for efficient and scalable cluster computing based on middleware technology, *In Proceedings of the CCGRID 2001*, 2001, pp. 40–47.

[20] G. Mendez and A. de Antonio, Training agents: An architecture for reusability, 5th International Working Conference on Intelligent Virtual Agents (IVA05), LNAI, vol. 3661, Springer-Verlag, 2005, pp. 1–14.

[21] E. Wenger, Artificial Intelligence and Tutoring Systems. Computational and Cognitive Approaches to the Communication of Knowledge, Morgan Kaufmann Publishers, Los Altos, California, 1987.

[22] E. Shaw, W.L. Johnson and R. Ganeshan, Pedagogical agents on the web, In Proceedings of the Third Annual Conference on Autonomous Agents, Seattle, WA, USA. ACM Press. 1999, pp. 283–290.

[23] J. Rickel and W.L. Johnson, Animated agents for procedural training in virtual reality: Perception, cognition, and motor control, *Applied Artificial Intelligence* **13** (1999), 343–382.

[24] J. Lester, B. Stone and G. Stelling, Lifelike pedagogical agents for mixed-initiative problem solving in constructivist learning environments, *User Modeling and User-Adapted Interaction* **9**(1–2) (1999), 1–44.

[25] J. Lester, J. Voerman, S. Towns and C. Callaway, Cosmo: A life-like animated pedagogical agent with deictic believability, In IJCAI97 Workshop on Animated Interface Agents: Making them Intelligent, Nagoya, Japan, 1997.

[26] A. Paiva and I. Machado, Life-long training with vincent, a web-based pedagogical agent, *International Journal of Continuing Engineering Education and Life-Long Learning* **12**(1) (2002).

[27] D. Zhang, L. Alem and K. Yacef, Using multi-agent approach for the design of an intelligent learning environment, In Proceedings of the Workshops on Commonsense Reasoning, Intelligent Agents, and Distributed Artificial Intelligence: Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications. LNCS 1441, 221–230. Springer-Verlag. 1998.

[28] C. Webber and S. Pesty, A two-level multiagent architecture for a distance learning environment, In Workshop on Architectures and Methodologies for Building Agent-based Learning Environments (ITS 2002), 2002, pp. 26–38.

[29] C. Buche, R. Querrec, P.D. Loor and P. Chevaillier, Mascaret: A pedagogical multi-agent system for virtual environments for training, *International Journal of Distance Education Technologies* **2**(4) (2004), 41–61.

[30] J.L. los Arcos, W. Muller, O. Fuente, L. Ore, E. Arroyo, I. Leaznibarrutia and J. Santander, Lahystotrain: Integration of virtual environments and its for surgery training, In Intelligent Tutoring Systems (2000), LNCS 1839, 43–52. Springer-Verlag. 2000.

Anthor's Bios

Jose L. Bosque graduated in Computer Science and Engineering from Universidad Politécnica de Madrid in 1994. He received the PhD degree in Computer Science and Engineering in 2003 and the Extraordinary Ph.D Award from the same University. He has been an associate professor at the Universidad Rey Juan Carlos in Madrid, Spain, since 1998 and from 2006 he is associate professor at the Universidad de Cantabria. His research is in the area of computer architecture, with special emphasis on high performance computers: interconnection networks, parallel and distributed processing, parallel computational models, performance and scalability evaluation and load balancing algorithms.

Pilar Herrero is an Associate Professor at the Universidad Politécnica de Madrid, in Spain. European Ph.D. in Computer Science. Extraordinary Ph.D. Award, in the last few years, she has also been involved in the organization of more than 15 international events, such as conferences and workshops with different roles: General co-Chair, Workshops General Chair, PC co-Chair, Steering Committee and Scientific Steering Committee.

Author of more than 85 international publications – some of them in prestigious international journals that are listed as Journal Citation Reports (JCRs)- and Program Committee member of more than 80 international conferences and scientific workshops, she has been involved in some networks of excellence and more than 30 research projects as both project manager and researcher.

Editor of more than 15 international publications and special issues in prestigious journals and several proceedings books, since 2005 she has been involved in the organization board of the On The Move (OTM) Federated Conferences as Workshops General Chair, and since then she has coordinated more than 70 international conferences, workshops and scientific events.

Susana Mata graduated in Computer Science and Engineering from Universidad Politécnica de Madrid in 1998. She received the PhD degree in Computer Science and Engineering in 2009 from Universidad Rey Juan Carlos. She has proffesional experience in software development in several national and international companies. Since 2002 she has been an assistant professor at Universidad Rey Juan Carlos. Her research is focused in the area of 3D graphics, more specifically in multiresolution techniques for the visualization of complex scenes and the generation of adpatively simplified polygonal meshes.

Gonzalo Méndez is an assistant professor at the Universidad Complutense de Madrid. He received a PhD. in computer science in 2008 from Universidad Politécnica de Madrid. He is co-organizer, together with Pilar Herrero, of the AWeSOMe workshop series within the On The Move (OTM) Workshops, where he also acts as Workshops Publicity Chair. He is also Program Committee member of several international conferences and workshops. His research interests are in the areas of Virtual Environments, Multi-Agent Systems, Educational Software and Software Design.