HAL
open science

An efficient algorithm based on weak synchronization for distributed in virtuo biological experiments

Vincent Rodin, Gireg Desmeulles, Pascal Ballet, Pascal Redou, Christophe Le Gal

To cite this version:

Vincent Rodin, Gireg Desmeulles, Pascal Ballet, Pascal Redou, Christophe Le Gal. An efficient algorithm based on weak synchronization for distributed in virtuo biological experiments. Multiagent and Grid Systems - An International Journal of Cloud Computing , 2011, 7 (4-5), pp.159-182. 10.3233/MGS-2011-0173 . hal-00668699

HAL Id: hal-00668699

https://hal.univ-brest.fr/hal-00668699v1

Submitted on 10 Feb 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An Efficient Algorithm Based on Weak Synchronization for Distributed In Virtuo Biological Experiments

Vincent Rodin[*,1,2], Gireg Desmeulles[1,3], Pascal Ballet[1,2],
Pascal Redou[1,3], Christophe Le Gal[1,3,4]

[1] European University of Brittany - UEB
[2] UBO, EA 3883 LISyC, F-29200 Brest, France
[3] ENIB, EA 3883 LISyC, CERV F-29280 Plouzané, France
[4] Cervval, 25 rue Claude Chappe, 29280 Plouzané, France
* corresponding author: vincent.rodin@univ-brest.fr

Abstract

Virtual Reality is becoming increasingly necessary to study complex systems such as biological systems. Thanks to Virtual Reality, the user is placed at the heart of biological simulations and can carry out experiments as if he were under the same experimental conditions as in vivo *or* in vitro. *We usually call this kind of experiments* in virtuo *experiments.*

In order to rapidly develop Virtual Reality applications related to biology, we have already proposed the RéISCOP meta model which makes it possible to easily design biological simulations and undertake in virtuo *experiments. This meta model allows to describe a biological system as a composition of its sub-systems and the interactions between the constituents of these sub-systems.*

Unfortunately, when using a single computer, the number of simulated entities is far from what is needed in biological simulations. It seemed thus necessary to extend the RéISCOP meta model so that it allows distributed computing on a grid. We made this choice because the structure of the RéISCOP meta model is well adapted to a distribution on a grid where the sub-systems which compose a system can be dispatched on different nodes, the synchronization and the coherence of the system being ensured by a Peer-to-Peer architecture.

Unlike traditional approaches which propose a spatial distribution, the method we describe in this paper is based on an "organizational" distribution linked to the RéISCOP meta model. This "organizational" distribution is mainly ensured by using two efficient algorithms based on a dead reckoning method, one for a data consistency between nodes and one for a weak synchronization of the nodes involved. These two algorithms are integrated into the behaviors of agents (DIVAs) which are located on each node of the grid. These agents are able to communicate by using a Peer-to-Peer architecture upon the grid.

In order to validate our approach, we implement three distributed simulations with increasing complexities and we compare the results with the results obtained in the non-distributed simulations. We get very similar results for the distributed and the non-distributed simulations.

Keywords: Distributed Simulation, Multi-Agent System, Multi-Interaction System, Peer-to-Peer, Virtual Reality.

1 Introduction

Research in the field of biology has long been using *in vivo* and *in vitro* experiments. Since the end of the 20th century, the *in silico* "experiment" has also been carried out in order to model and simulate biological phenomena. Unfortunately, during an *in silico* simulation the user does not have any more interaction with the simulated model. The user loses the interactions which are inherent in the *in vivo* and *in vitro* experiments. In order to overcome this problem, Virtual Reality tools can be used to make the interactions possible during the simulation. Thanks to Virtual Reality, the user is now placed at the heart of biological simulations and can carry out experiments as if he were under the same experimental conditions as *in vivo* or *in vitro*. We usually call this kind of experiments *in virtuo* experiments [47].

In order to rapidly develop Virtual Reality applications related to biology, we have already proposed the RéISCOP meta model which makes it possible to easily design biological simulations and undertake *in virtuo* experiments [14]. This meta model, based on the reification of interactions, allows to describe a biological system as a composition of its sub-systems and the interactions between the constituents of these sub-systems. The interactions which are the autonomous elementary units of the modelled system are implemented using real computer objects. The constituents of the sub-systems are modified by these interactions.

Unfortunately, when using a single computer, the number of simulated entities (i.e. interactions) is far from what is needed in biological simulations. It seemed thus necessary to extend the RéISCOP meta model so that it allows distributed computing on a grid. There are many frameworks (e.g. OGSA-compliant frameworks) supporting distributed computing on a grid but they are not well adapted to our problem statement, because we need efficient algorithms to increase the size of our simulations developped with our RéISCOP meta model. In the related work section, we will explain why most of previous work dealing with Distributed Virtual Reality over a Grid (DVR-G) are not suitable for our *in virtuo* biological simulations based on the RéISCOP meta model.

Existing DVR-G systems often use a spatial partitioning scheme to distribute a virtual environment [26, 51]. In this paper, we describe an "organizational" partitioning in order to be close to the RéISCOP meta model architecture during the distribution of a virtual biological environment. We made this choice because the structure of the RéISCOP meta model is well adapted to a distribution on a grid where the sub-systems which compose a system can be dispatched on different nodes, the synchronization and the coherence of the system being ensured by a Peer-to-Peer architecture. This "organizational" distribution is mainly ensured by using two algorithms originally designed for spatial partitioning [4], to which we add a dead reckoning part. One of these algorithms concerns data consistency between nodes and the other performs a weak synchronization of the nodes involved. These two algorithms are included into the behaviors of agents (DIVAs) which are located on each node of the grid. These agents are able to communicate by using a Peer-to-Peer architecture upon the grid.

In order to validate our approach, we implement three distributed simulations with increasing complexities. The first one shows a very simple simulation in which the main distribution mechanisms are involved. The second one is the well known Mitogen-Activated Protein Kinase (MAPK) cascade simulation [30]. Thanks to this second example, we are able to compare, on a real biological simulation, the results obtained using a distributed simulation and a non-distributed one. We get very similar results in both cases. Finally, we introduce a complete application using the RéISCOP meta model distribution.

The rest of the paper is organized as follows. In section 2, we describe, on one hand, the concept of *in virtuo* experiments and, on the other hand, the RéISCOP meta model. In section 3, we present briefly two

areas on which our contribution is based: Distributed Virtual Reality and Grid Computing architectures. In Section 4, we present our DIVA Agents used for the distribution of biological simulations. Section 5 mainly describes the "organizational" distribution of RéISCOP and the consistency algorithm we use. In Section 6, we describe three applications that validate our approach. Section 7 presents some related work in the specific area of Grid Computing applied to Distributed Virtual Reality. Section 8 concludes the paper.

2 *In virtuo* experiments and RéISCOP meta model

The study of biological systems cannot be limited to the sole experiments made on laboratory benches. More and more, biologists use computer science tools. Particularly, in the genetic field, computer science is used to analyze nucleobase sequences and other DNA and protein elements. Computers allow to process a lot of data produced by the experiments. This field of computer science is called bioinformatics. Moreover, computer science forms a good alliance with mathematics when it comes to analyzing, understanding and simulating biochemical reactions.

Another less common approach in computer science for the study of biological systems is virtual reality. The main goal of the research taking place in our laboratory is the study of complex systems using virtual reality tools and methods. Part of this research applies to biology. For instance, simulations of blood coagulation [29], myeloma [41], allergic urticaria [14] and arteries vasorelaxation [4] are being fulfilled.

The use of virtual reality tools offers more than the solving of an equation of a biological model, like *in silico* simulations; it also enables the user to enter a virtual environment that contains his model with which he can interact at any time: the human is part of the model's simulation loop. This way of experimenting a model is called *in virtuo* experiment [47], in reference to *in vivo* and *in vitro* experiments that respectively take place on living systems and test-tubes. The *in virtuo* experiments allow the user to make experiments as in reality, as if he were under the same experimental conditions as *in vivo* or *in vitro* experiments.

A recent study [14] on the autonomy in biological systems models (focused on interactions and coupling between different autonomous parts of a biological system) led to the RéISCOP meta model and the associated API (Application Programming Interface). This API allows to easily specify or reuse autonomous entities and other objects characterizing *in virtuo* experiments (e.g. cell, chemical reaction, chemical diffusion, etc.) and also utilities like SBML parser.

The RéISCOP meta model (for ***I**nteractions* **Re**ification ; ***S**tructure* ; ***C**onstituent* ; ***O**rganization* ; ***P**henomenon*) allows to describe a system as a composition of its sub-systems. Each sub-system (called *Organization*) is composed of active elements (called *Interactions*) that act on passive elements (called *Constituents*). The set of passive elements can be handled by *Interactions* taking place in different *Organizations*. In this case, *Organizations* are structurally coupled (see figure 1).

The fact that *Interactions* are real computer objects and that they perform actions is another essential point of this model. These *Interactions* are not the result of the action of one *Constituent* on another. The *Interactions* are objects which have attributes and activities (periodic actions). And this is called *Interaction reification*. These *Interactions* act on specific *Constituents*. *Phenomena* instantiate *Interactions* taking place between the designated *Constituents* when required conditions are found. Therefore, this modelling is an interaction based modelling. Figure 2 shows the UML class diagram of the RéISCOP meta model. Several specific models in the chemistry, mechanics or biology fields can be derived from this RéISCOP meta model.

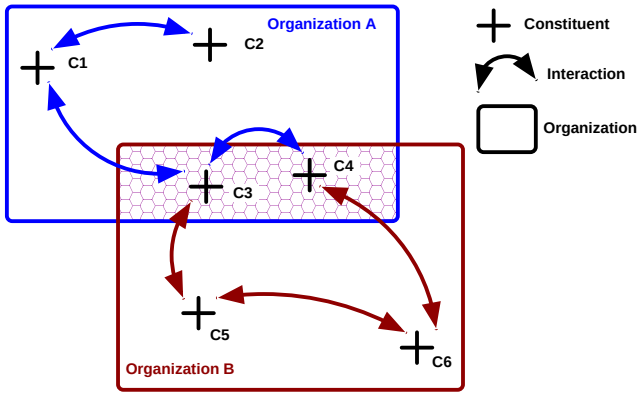

Figure 1. Structural coupling in the RéISCOP meta model: the figure shows two Organizations (A and B) with a structural coupling. The Structure of Organization A is composed by the Constituents set {C1; C2; C3; C4}, while for Organization B it is {C3; C4; C5; C6}. The set {C3; C4} represents the structural coupling between A and B. The two systems (according to the two Organizations) have mutual influences through the modifications of C3 and C4 states.

3 Distributed Virtual Reality and Grid Computing architectures

To design and perform our Peer-to-Peer distribution software, we have been dealing with distributed virtual reality on one hand and, grid computing on the other hand. In this section, we present briefly the main architectures used in this two areas.

3.1 Distributed Virtual Reality architectures

The main issue of Distributed Virtual Reality (DVR) is the interconnection and the coordination of virtual reality simulations through a network. Many DVR systems exist and generally allow several users to share a virtual environment from separate workstations. One can quote for example DIVE [21], NPSNET-V [10] and VIPER [48]. DVR systems must ensure the consistency of the shared environment even when several users act (i.e. move, modify, etc.) on the same virtual object [43]. For example, if one of the users moves an object, the distant users need to perceive the movement of this object. Most DVR architectures which can be found in literature are based either on low level protocols or on standard protocols (i.e. DIS, HLA, etc.) and can use three types of data model: centralized, distributed, replicated [36].

Historically, the DARPA (Defense Advanced Research Projects Agency) initiated research in this field by creating SIMNET [7]. The objective was to develop single-user simulators (tank, helicopter, etc.) and connect them to a network so as to obtain a military team training platform. Each site is responsible of several virtual objects (i.e. it simulates the complete behaviors of these “Real” objects). In order to ensure the consistency of the virtual world, a site computes two states for each object under its responsibility: a real state and an approximate state (location, velocity, acceleration). When the real state and the approximate state diverge, the approximate state is updated and broadcasted to distant sites. When a distant site receives these informations, it updates the location, velocity and acceleration of a

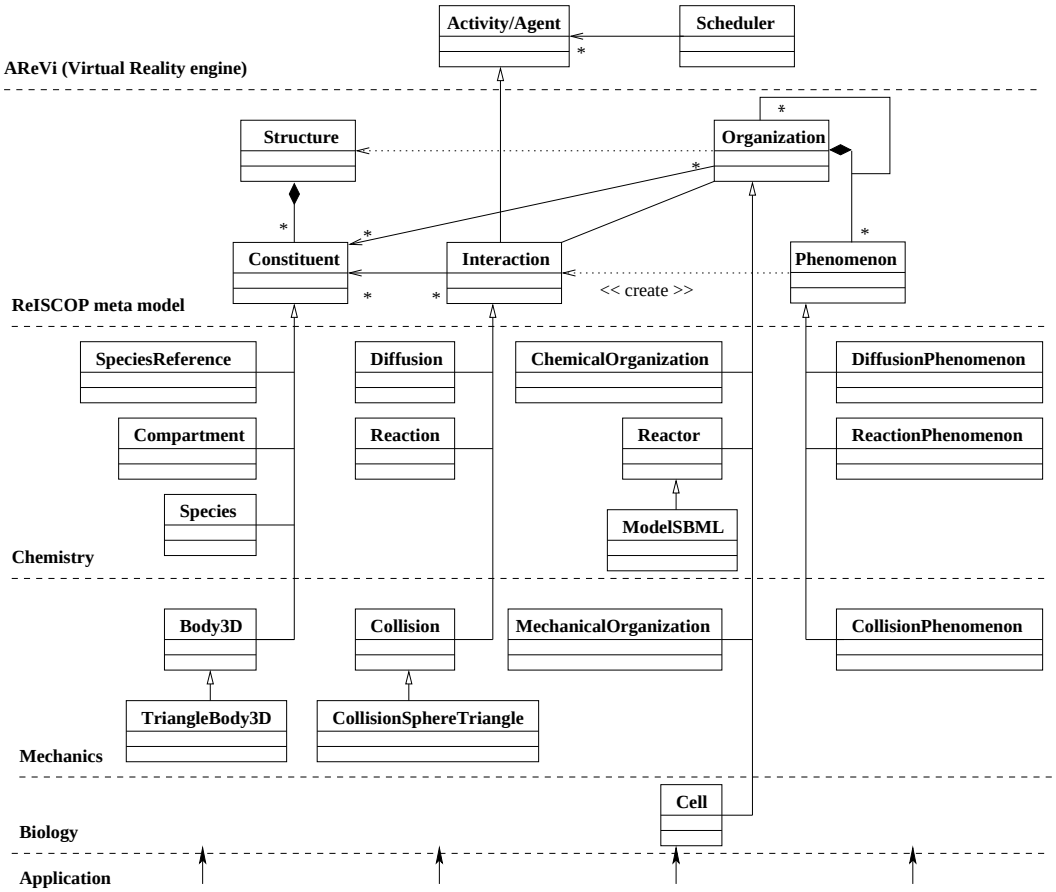

Figure 2. UML Class Diagram of RéISCOP meta model.

simple 3D object (i.e. a “Ghost” objet which correspond to a “Real” object located somewhere on the network). This method is called dead reckoning and uses the “Real/Ghost” model [38, 44].

Due to these research work, the DIS (Distributed Interactive Simulation) standard was created [24]. The DIS standard defines a data exchange protocol (PDU: Protocol Data Units) between remote workstations participating to a military simulation. One of the most known platforms using DIS is probably NPSNET [37]. Parallel to the development of DIS, the DARPA decided to create the HLA (High Level Architecture) standard [49]. HLA is a software architecture to facilitate interoperability between simulations. The RTI (Run-Time Infrastructure) is the core of the architecture. It provides common services and ensures the consistency of the distributed simulations. Developed by the Naval Postgraduate School (NPS), the Bamboo toolkit uses HLA so as to construct distributed virtual environments in a dynamic way [34].

More recently, a Grid Computing based approach appeared to design DVR systems and applications. In the related work section (section 7), we will detail some previous work dealing with Distributed Virtual Reality over a Grid (DVR-G) because our contribution is mainly guided by this recent research area. Indeed, in this paper we present a method which uses a Peer-to-Peer architecture over a grid in order to distribute *in virtuo* experiments.

3.2 Grid Computing architectures

Grid computing can be separated in two groups: global computing and metacomputing. Global computing consists in the gathering of unused processor cycles of individual connected computers. Regarding metacomputing, the principle is to use dedicated computers, what is done especially in research centers. Metacomputing is based on the following principle: a client who has a problem sends it to available and capable servers. Nowadays, many dedicated environments are used (such as DIET [12] or Globus [19] for instance) and most of research work about grid computing address load balancing (static [5] or dynamic [8]) and fault tolerance problems [15, 40].

One can find three main architectures usually used to built grid computing systems: the Open Grid Services Architecture, the Client-Server architecture and the Peer-to-Peer architecture.

Open Grid Services Architecture (OGSA): the Open Grid Services Architecture is a stardard architecture for a service-oriented grid computing [20]. This architecture is increasingly used because it allows the easy deployment of grid applications. An OGSA-compliant grid framework must have the following capabilities: infrastructure services, execute management services, data services, resource management services, security service, self-management services and information services. One of the most used OGSA toolkit is the Globus Toolkit [17], currently at Version 5. By implementing standard protocols, Globus Toolkit allows users to access remote resources transparently. This toolkit is composed of modules which can be used independently according to the concerned application. As we explained before, OGSA architecture is not well adapted to our problematic, because we need efficient algorithms to distribute our virtual biological simulations developped with the RéISCOP meta model.

Client-Server: a very common architecture for grid computing is Client-Server architecture, also called ASP model (Application Service Provider). The servers are computers that have problem solving capabilities [1]. The clients are grid users, and put forward problems to solve. More precisely, an agent-client-server model is used. The reason is: each server (that can solve one or more specific problems) registers itself to the agent. The client transmits its problem to the agent. The aim of the agent is to find the appropriate server (the latter knows how to solve the problem and has memory and processor resources to do it) and to connect it to the client. At that moment, the client transmits its problem and its data to the server [3]. The server solves it. In our work, we do not use a Client-Server approach because it is a centralized architecture, not very adjustable and not very adaptable in which roles of components are frozen.

Peer-to-Peer (P2P): another architecture exists for grids, particulary for grid database: the Peer-to-Peer [16]. Each computer of the grid has data and searches other data little by little without centralization or little (we can quote Napster - with a centralized part - and Gnutella). Each computer has the same role as the others. Consequently, this grid is open and dynamic. This architecture is attractive and a way of research tries to use it for distributed calculations [25]. Several teams are working to build API or middlewares that provide basic services for P2P network [2, 9, 22]. So, each peer has the total control of its local resources, and can get involved or withdrawn from the system at any time. However some difficulties inherent in grid computing are staying present (load balancing [50], task scheduling [32], synchronization [27]) and must be managed in an equitable way by the peers community. In our work, we decided to use a P2P approach because Peer-to-Peer systems and Multi-Agent systems share many characteristics [39]. For example, we can qualify Peer-to-Peer systems of self-organized: they have

dynamic and distributed reconfiguration properties in function of the number and the characteristics of nodes. These common characteristics between peers and agents allow us to design a dynamic and efficient architecture to distribute *in virtuo* experiments.

4 Distributed In Virtuo Agents: DIVA

In the last section, we have just seen that both Peer-to-Peer and Multi-Agent systems share very interesting characteristics. To our knowledge, only few examples combine Peer-to-Peer grid computing and Multi-Agent Systems. In [11], the authors present a Peer-to-Peer approach in order to obtain an autonomous scheduling over a grid with a decentralized nature of control. In this approach, every node has equal access and the same role than the other nodes. In [45] a grid computing architecture is proposed. It is a Peer-to-Peer architecture where each node of the grid is composed of 3 agents collaborating together. The Multi-Agent System is at the node level and not at the grid level. Another architecture is described in [54]. This architecture consists of four layers: grid middleware layer, Peer-to-Peer communication layer, agent layer and application layer. The agent layer is on top of the Peer-to-Peer layer in order to allow each peer node to provide an agent manager responsible, for example, of the creation or the destruction of peer agents.

In the rest of this section, we will first remind the notions of agent and Multi-Agent System. We then use these notions to describe our DIVA software based on a Peer-to-Peer architecture extended with the agent paradigm.

4.1 Agent paradigm

Agent: an agent is an object (in the computer science sense) which is active and autonomous. Demazeau [13] describes an agent like a real or virtual entity which has an autonomous behavior and evolves in an environment. An agent can perceive its environment, can perform actions on it and can interact with other agents. Wooldridge [53] adds to this definition: an agent is a computing system with capabilities to make actions in an autonomous and flexible way in its environment. The flexibility means reactivity, pro-activity and possession of social capabilities. Classically, the life cycle of agents follows the sequence of actions [47]: *perception* (the agent perceives its immediate environment through specialized sensors), *decision* (it decides what it must do, taking into account its internal state, its sensor values and its intentions), *action* (it acts by modifying its internal state and its immediate environment).

Multi-Agent System: a Multi-Agent System is made up of several of these autonomous and pro-active entities with a shared environment. Demazeau [13] identifies the key-concepts of a Multi-Agent System as being:

- the Agents and especially their knowledge, their goals and their reasoning;
- agents' Environment: all that does not belong to them and on which they can act;
- the Interactions between agents: communications, etc.;
- the Organizations: the dependencies networks between agents and structures built in this way.

Thanks to the autonomy of the agents (in accordance with localization of their resources and their independent behavioral decisions), they can be physically distant to each other and form a distributed Multi-Agent System. Let us note that some previous work on grid systems address this multi-agent paradigm [18, 33, 42, 46].

4.2 DIVA overview

The distributed simulation using a Peer-to-Peer architecture is supervised by the DIVA software. This software, installed on each node of the grid, is an agent which can be viewed like a background task (daemon): this agent does nothing when it is not requested. A user can start a simulation from any node. Then, this node has additional responsibilities comparatively to the others (simulation starting, observation tools and user-simulation interactions tools), and keeps also the basic functionalities of each node. These shared functionalities (load balancing, synchronization, spatial, "organizational" and temporal consistency maintenance) allow us to consider that each node has a similar role and is a peer. In addition these peers are active and pro-active thanks to periodically executed activities (some of these will be detailed below). Their pro-activities allow them to anticipate and cooperate in order to execute as better as possible the *in virtuo* experiments. The nodes are actually agents. That justifies the name we give to this software: ***D**istributed **I**n **V**irtuo **A**gents*. This DIVA software (see figure 3) is constituted of a Central Decision Making module (CDM) and is complemented by several peripheral modules for the network interface, the observation of resources load, the load balancing, the management of the ARéVi simulation (ARéVi: a Virtual Reality engine which enables agent programming), the periodic save, etc.

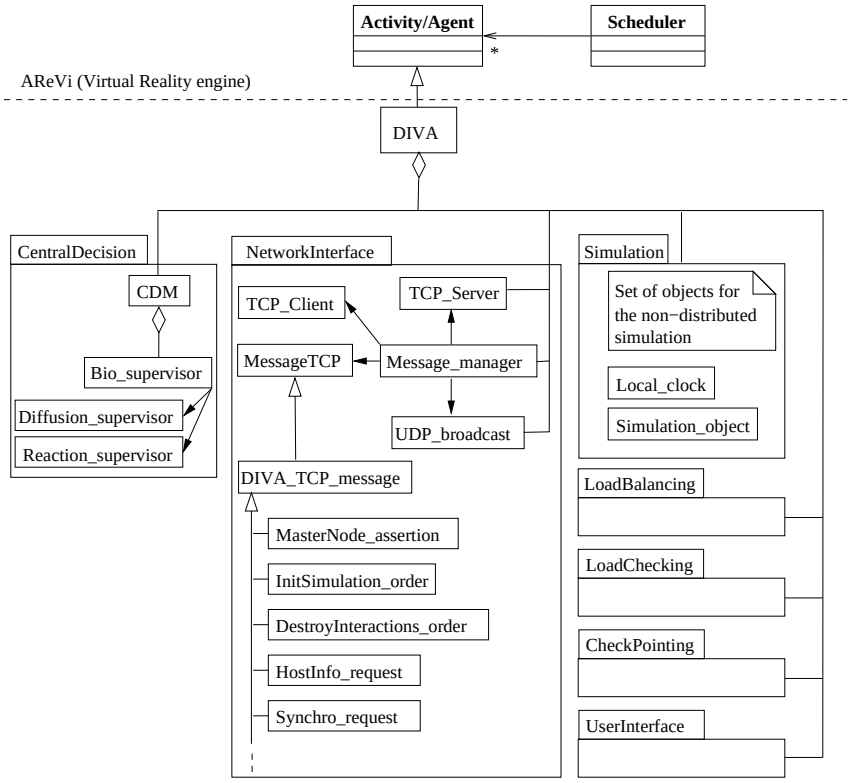

Figure 3. Simplified UML class diagram of the DIVA software.

4.3 Collaborations between DIVAs

Some tasks of the grid management are undertaken by DIVA collaborations in our Peer-to-Peer model. Like in a classical Multi-Agent System, there is no more centralized system. Let us see some examples of collaborations that underline our multi-agent view of the Peer-to-Peer computing over a grid.

Identification: when a DIVA is created on a node, the first action which it carries out is the diffusion of its birth certificate: the DIVAs already present on the network receive this message and answer it in order to notify their presence (i.e. each DIVA knows the others). Then, on the order of the user, the DIVA located on the main node (workstation on which the user is connected) proceeds to the distribution of the simulation by using load balancing.

Load balancing: before running the distributed simulation, the set of the simulation components (biological interactions, chemical species, etc.) are moved onto the set of nodes accepting this simulation. After this static load balancing carried out, the different places of calculations can be changed in order to optimize the "resources quantity/calculation load" adequacy. This optimization will be carried out dynamically by the DIVA collaboration. Several collaboration types between agents exist. We can cite the famous "Contract Net Protocol" as well as its improvements and its by-products like auctions [6].

Time of simulation: there is no global clock in our system. Each node of the grid has its own local clock, so that the whole simulation maintains consistency, these clocks must be as synchronized as possible. In order to do this, each message passing between the DIVAs has a timestamp. During a point to point communication, the receiver DIVA can take the choice to freeze its part of simulation if it is too early compared with the emitter (the threshold is adjustable). So, since all DIVAs have regular bi-directional communications between each other, the local clocks are synchronized two by two and the set of clocks have a very similar time.

Consistency of the environment: in our model, the autonomous entities of the biological simulation are not duplicated. Exclusively the parts of the environment can have several images. It is fundamental that these images of a same environment part do not cause important errors in the simulation. These images must stay in a consistency state. As mentioned by Keller [28], consistency in a virtual world occurs when a scene is identically perceived by all the entities which observe it. So, there is disconsistency when a modification is applied to one of the images and not to the others. These repercussions, or these updates of images are resynchronizations of the images. Each DIVA takes regularly the decision to resynchronize the environment parts that have an image on a distant node. In addition, between two resynchronizations, a distant node needs to estimate the state of the node. The estimation is performed thanks to the dead reckoning algorithm which permits to extrapolate the state of a node by using previous real data of this node.

5 Organizational distribution of RéISCOP using DIVAs

5.1 Consistency model

In our model, the active elements, i.e. *Interactions*, are not duplicated. Only parts of the environment can have many images or replicas. These parts, the *Constituents* for instance, are passive elements. These images of the *Constituents* must not cause substantial bias in the simulation and must be consistent.

Spatial distribution is a common idea. In fact, a simulation with a cubical environment will be split into a set of "small cubes", while a simulation with an elongated environment - like biological vessels - will be split into slices. This approach has been chosen for distributed virtual reality simulations based on

HLA or distributed video games. Commonly, this approach is called data decomposition. Furthermore, in the context of video games or virtual reality, those data are physical objects which are located in space.

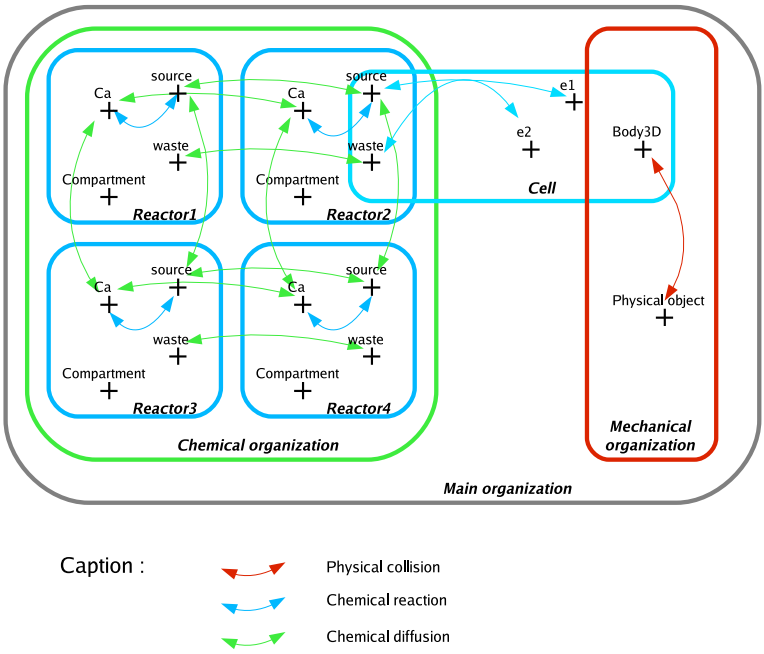

Figure 4. Example of model to be simulated (non-distributed version).

Another choice of distribution can be done. The distribution that we propose is an "organizational" distribution which is based directly on the structure of the RéISCOP meta model. The principle is the following. First of all, one should reason only in term of *Organization-Phenomenon-Interaction-Constituent*. Then, it is important to note that an *Organization* is not divided and is not distributed on several computers. *Organizations* play here the role of "small cubes" which are present in spatial distribution (see figures 4 and 5 describing an example of "organizational" distribution). Since in RéISCOP a *Constituent* can belong to several *Organizations*, if an *Interaction* modifies a *Constituent*, it can be necessary to carry out a consistency of the value of this *Constituent*. Indeed, this *Constituent* may belong to *Organizations* which are not on the same computer.

Whether spatial or "organizational", the distribution involves a significant use of computing resources due to synchronizations. For example, if two entities interact together when simulated on different computers without shared memory, it is necessary for the computers to exchange faithfully the actions of the two entities. A great quantity of exchanges between the two simulation stations can therefore be induced. In the case of *in virtuo* experimentations, we choose to implement a weak synchronization in order to reduce the network load.

Let us explain how weak synchronization differs from strong synchronization. In the case of a strong synchronization, the data on simulation entities and their states are transmitted and updated at each change. On the opposite, the weak synchronization implies a periodical transfer of these data from time to time. More precisely, in our simulations, a scheduler runs each activity at a specific moment. A synchronization is considered as strong when it is executed at each step[1] of the scheduler. So, between

[1]one sequential operation of the scheduler, i.e., one activation of one *Interaction*

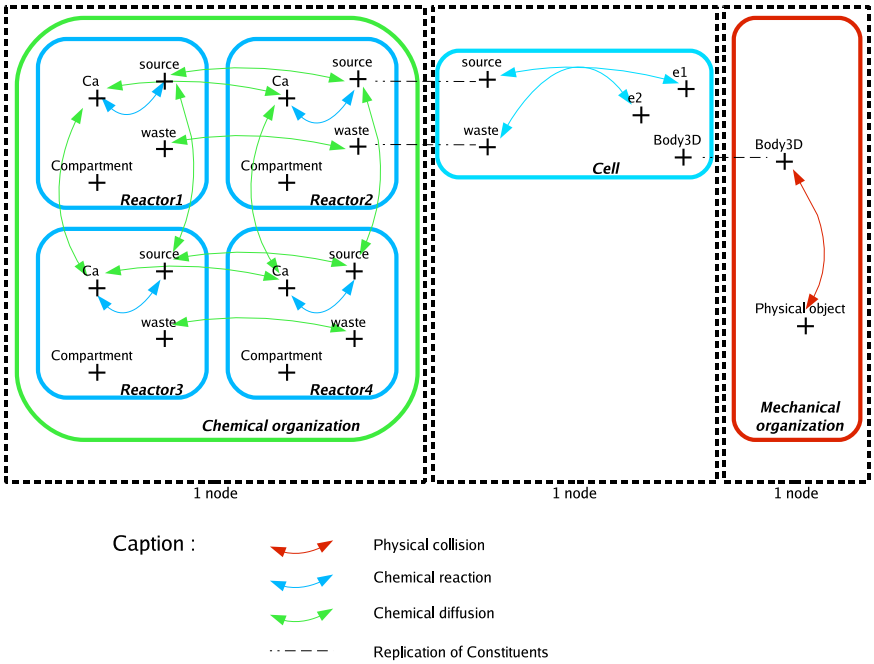

Figure 5. Same example as in figure 4, but distributed on 3 nodes. Let us note that the *Organizations* are distributed and that some *Constituents* may have several images.

two synchronizations, entities of the simulation can change their state only once. With a weak synchronization, we release this constraint in order to make a synchronization with a lower frequency than the scheduler's. This process allows to reduce network exchanges between two stations but it is not safe: consider a station, we know for a fact that the vision of the simulated entities on remote stations is false. Recall that, here, entities of the simulations are *Constituents* of RéISCOP meta model. *Constituents* are passive objects that are subject to *Interactions* activities and thereafter their state changes. As mentioned above, an *Interaction* can be on only one simulation station at once, whereas *Constituents* can have images on many stations. Therefore, we must synchronize these images from time to time.

Let us consider for instance two *Interactions* of diffusion acting on a chemical species (i.e. the *Constituent*) in a $1 * 1 * 3$ size environment (see figure 6.a). When time is $t = 0\ second$, the chemical species concentration in the middle area is initialized to 100. The activity of the diffusion *Interaction* is to balance the chemical species concentrations at each scheduler step (the processing of diffusion is based on the first Fick law). Figure 6 sums up our distribution method for this example. We are going to detail the distributed simulation with three kinds of synchronization: strong synchronization, weak synchronization and weak synchronization with our specific consistency algorithm. Figure 7 shows the results:

a/ We consider here the strong synchronization with a *Constituents* state propagation at each scheduler step: thanks to *Interactions* of diffusion, concentrations balance is reached and the matter quantity is the same.

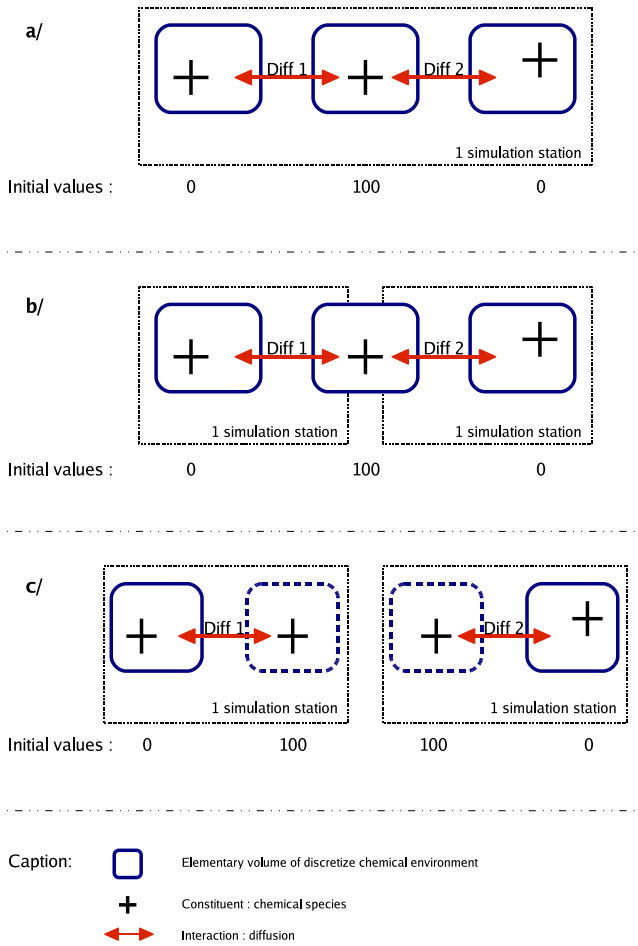

Figure 6. Example of a bio-chemical environment distribution on 2 simulation stations: a/ sequential simulation; b/ the first step of the distributed simulation: *Interactions* are moved but not replicated - what about the shared elementary volume ?-; c/ *Constituents* of shared elementary volume are replicated: 2 images for the same *Constituent* with identical initial values.

b/ Now the weak synchronization: *Constituents* synchronization (by their state transfer) is faulty when the synchronization action is not carefully adjusted on the scheduler of *Interactions*. The invariant is incorrect.

If the *Constituent* state is not transfered to all images when modifications or disruptions occur, the synchronization cannot lead to correct results. To solve this problem, we have built a mechanism called *images consistency algorithm* (algorithm 1) that reduce as much as possible the error (as shown in figure 7-c) appearing during a weak synchronization. This mechanism is not based on the *Constituent* state transfer, but on the transfer of its state variation. Unlike the Distributed Virtual Reality simulations and the Real/Ghost model, none of our images have a reference state. And the sole transfer of the state only is not sufficient to get a consistent simulation. Precisely, the *Constituent* state (see in figure 2 the UML Class Diagram of RéISCOP meta model) is a set of real values needed in order for it to be characterized in the distributed simulation. For a located chemical species in an elementary volume,

the state is the chemical concentration. For a cell body, the state is its spatial coordinates. So, the state variation is just the difference between the current state and a given previous state. This algorithm requires additional memory in order to store the *Constituent*'s previous state (see the algorithm 1).

Algorithm 1: Image consistency algorithm

```
begin
    // Comment: "concernedDIVAs" is a set of
    // DIVA sharing at least one image
    concernedDIVAs ← FindDIVAs()
    for i ∈ Images do
        LocalDelta_i ← CurrentState_i − LastConsistencyState_i
        WaitedDeltaNb_i ← card(concernedDIVAs)
        DeltaSum_i ← LocalDelta_i
        for d ∈ {concernedDIVAs} do
            send i to d
            send LocalDelta_i to d
    while Σ_{i∈Images} WaitedDeltaNb_i ≠ 0 do
        image ← receive
        remoteDelta ← receive
        DeltaSum_image ← DeltaSum_image + remoteDelta
        WaitedDeltaNb_image ← WaitedDeltaNb_image − 1
    for i ∈ Images do
        CurrentState_i ← LastConsistencyState_i + DeltaSum_i
        LastConsistencyState_i ← CurrentState_i
end
```

The consistency algorithm starts when a synchronization barrier is reached. A synchronization barrier is a date (of the simulation global virtual time) at which the group of local simulations must stop. More precisions on this algorithm are given in the next subsection. In the example, with this algorithm (figure 7-c), the two images are consistent and the simulation is not faulty. With this mechanism, we can simulate a distributed bio-chemical environment on several simulation stations. In fact, after the environment's splitting, and the balance of *Phenomena* and *Constituents* of each elementary volume on different stations, only the images of *Constituents* are to be managed. RéISCOP is responsible for each local simulation.

Notice that the consistency algorithm is the same whatever the DIVA, whatever the situation. No role (client or server, kind of the image) is specified, the similarity of the nodes is shown, and it is a real Peer-to-Peer architecture. In addition, since the consistency algorithm is only executed periodically, each DIVA has a dead reckoning module (see section 3.1) allowing him to extrapolate the current value of a *Constituent* from the last values and their variations of first and second order. The figure 8 shows the effect of the dead reckoning algorithm during a diffusion experiment between two nodes.

5.2 Implementation

To fulfill the *Constituents* consistency, it is necessary that DIVAs, sharing images, stop for a short time, while the consistency algorithm is executed and the useful data transfered. In order for DIVAs to stop together and achieve transfers at the same time without a central supervisor, each DIVA implements

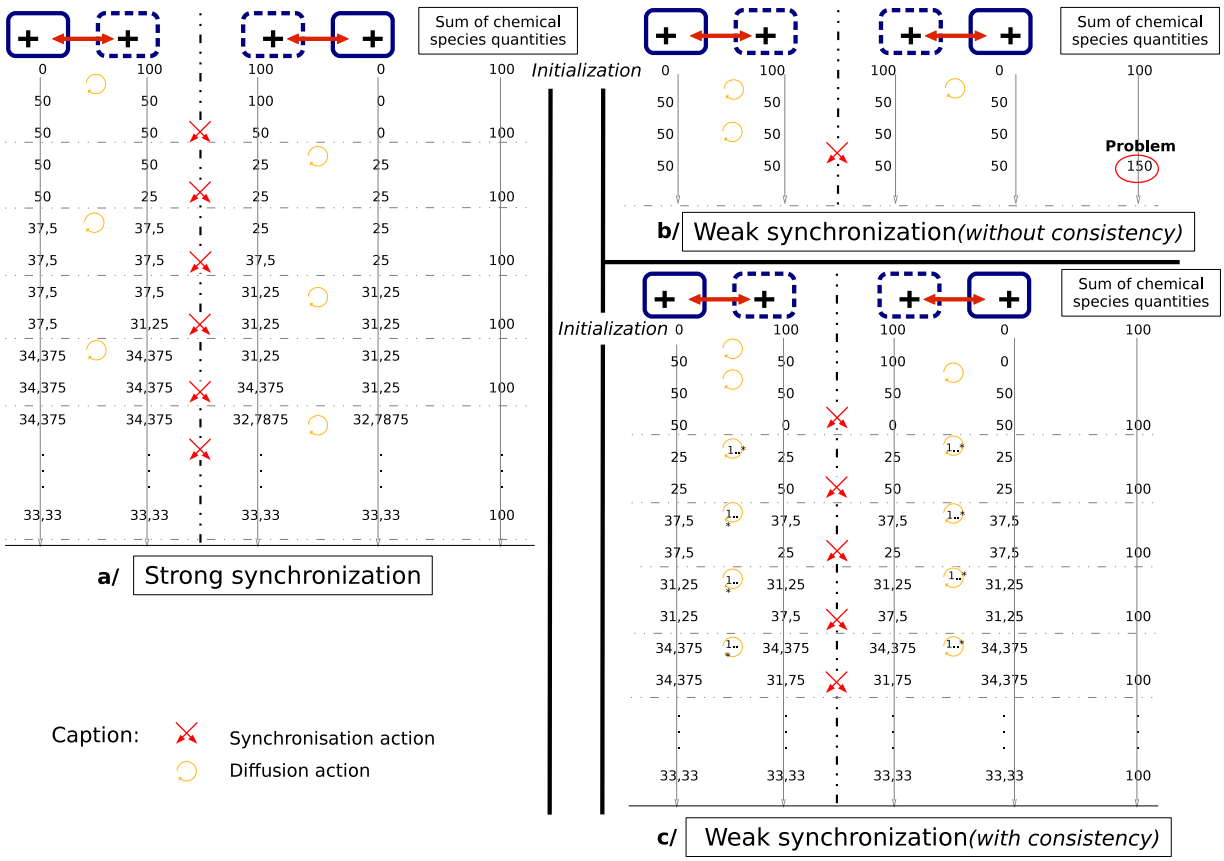

Figure 7. A synchronization example between 2 simulation stations in the framework of one chemical species and two diffusion *Interactions* distribution. a/ Strong synchronization: at each *Interactions* activity, chemical species concentration transfer occurs (the state of the *Constituent*) b/ Weak synchronization: the concentration transfer is being done regardless of *Interactions* activities. c/ Weak synchronization with consistency algorithm.

a *rendezvous* mechanism which is a part of its behavior. This *rendezvous* mechanism allows to set up a synchronization barrier.

When DIVAs reach the time of the *rendezvous*, they execute the consistency algorithm. In the current version, DIVAs share a token and send it to each other. When one DIVA receives the token, it takes the initiative to start the *rendezvous* algorithm. The others answer to it and a common date is decided. This date is the latest Local Virtual Time of the concerned DIVAs. The details of the *rendezvous* algorithm are shown in algorithm 2. This algorithm uses classical TCP/IP communications in order to send and receive data in an asynchronous way. When the *rendezvous* is finished (including the consistency), the DIVA that has the token sends it to another DIVA. Considering a Local Area Network, the token transfer is simple and does not need specific tools for its routing.

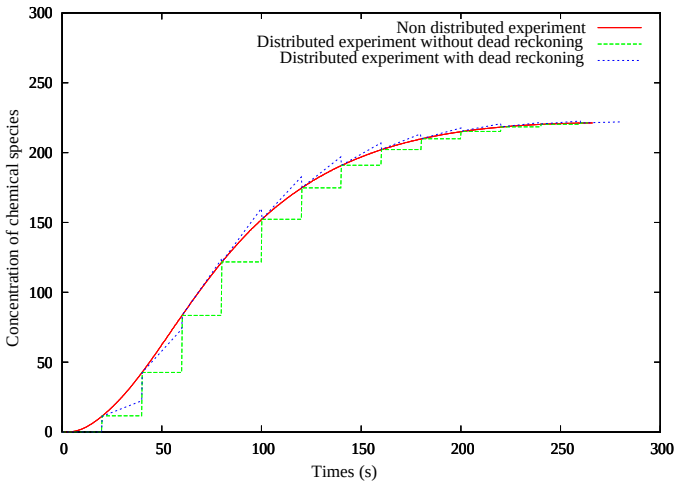

Figure 8. The dead reckoning algorithm uses a first order extrapolation. In order to see the linear extrapolation, the interval between two updates is voluntarily very high in comparison with the values used during our real simulations.

6 Applications

In the first part of this section, we show the results of a simple simulation that validates our approach. Even though this first application is quite trivial on the biological point of view, it enables us to fully control the simulation process, consequently giving us the opportunity to confirm our solution and illustrate the principle of image consistency. This example is based on the diffusion of a chemical species in a distributed chemical environment. In the second part of this section, we present the MAPK cascade distributed simulation which is also based on the diffusion of a chemical species. Then, in the last part of this section, we introduce a bigger and more complex distributed application: the artery vasorelaxation. This application involves the diffusion of chemical species but also the movement of virtual cells.

6.1 Diffusion experiment

Consider a simulation that only uses the diffusion *Interaction* in a bio-chemical environment. In a $25 * 3 * 3$ size environment, we inject a quantity of $36.0 * 10^3$ units of a chemical species in the middle part of the simulated environment at time $t = 0$ second. We let diffusion *Interactions* (acting between each mesh of the discrete environment) execute their activities.

To get the following results, we put two virtual probes in the environment: the first one is placed in the central location and the second one is outlying, at 32% of the length of the environment (see figure 9).

The simulation is executed three times: the first time with one computing node only, the second time with five nodes and the last time with five nodes but without the consistency algorithm.

Results in figure 10 show that curves are overlaid in the first and second cases. On the other hand, in the third case, when the DIVAs do not organize *rendezvous* for consistency to be fulfilled, the injected chemical species can only diffuse itself in the environment simulated by the node, which corresponds to the center of the environment. The chemical species quantity reaches 666.7 instead of 160.0 (asymptotic value when $t = +\infty$). The curve corresponding to the second probe is not drawn in the third case

Algorithm 2: Rendezvous algorithm

```
GoToSynchronizationBarrier():
begin
    UnfreezeLocalSimulation()
    runUntil(Rdv_date)
    FreezeLocalSimulation()
end
ReceiveAllAnswers():
begin
    answers_stack ← ∅
    while card(answers_stack) ≠ card(concernedDIVAs) do
        answer ← receive from emitter
        answers_stack ← answers_stack + {answer}
end

Main algorithm:
begin
    if OwnToken() = true then
        FreezeLocalSimulation()
        // Comment: "concernedDIVAs" is a set of
        // DIVA sharing at least one image
        concernedDIVAs ← FindDIVAs()
        Rdv_date ← 0
        for d ∈ {concernedDIVAs } do
            send LocalVirtualTime to d
        ReceiveAllAnswers()
        Rdv_date ← max(answers_stack ∪ LocalVirtualTime)
        for d ∈ {concernedDIVAs } do
            send Rdv_date to d
        GoToSynchronizationBarrier()
        ReceiveAllAnswers()
        for d ∈ {concernedDIVAs } do
            send StartConsistencyOrder to d
        ConsistencyAlgorithm()
    else
        date ← receive from emitter
        tokenOwner ← emitter
        FreezeLocalSimulation()
        if date > LocalVirtualTime then
            send date to tokenOwner
        else
            send LocalVirtualTime to tokenOwner
        Rdv_date ← receive from emitter
        GoToSynchronizationBarrier()
        send Rdv_date_reached to tokenOwner
        StartConsistencyOrder ← receive from emitter
        ConsistencyAlgorithm()
end
```

because the measured value is always equal to zero. In conclusion, our solution is validated by those adequate results: the weak synchronization and the consistency algorithm build a correct distributed simulation.

6.2 MAPK cascade

This example concerns the MAPK (mitogen-activated protein kinase) cascade [30]. The MAPK pathway is a series of enzymatic reactions. This pathway is present in all eukaryotic cells and is responsible for lots of control functions for the life cycle of cells. This pathway was modelled in our simulation with the Kholodenko model (see figure 11).

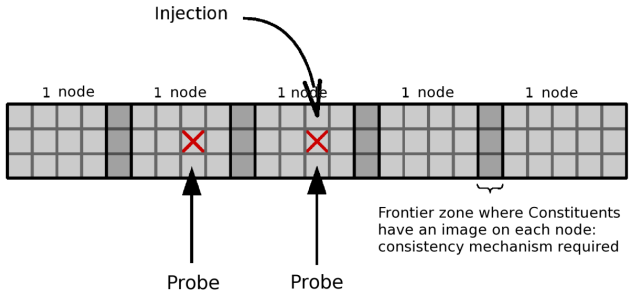

Figure 9. Layout of the diffusion experiment: a $25*3*3$ size environment and two probes. Injection of a chemical substance at the center.

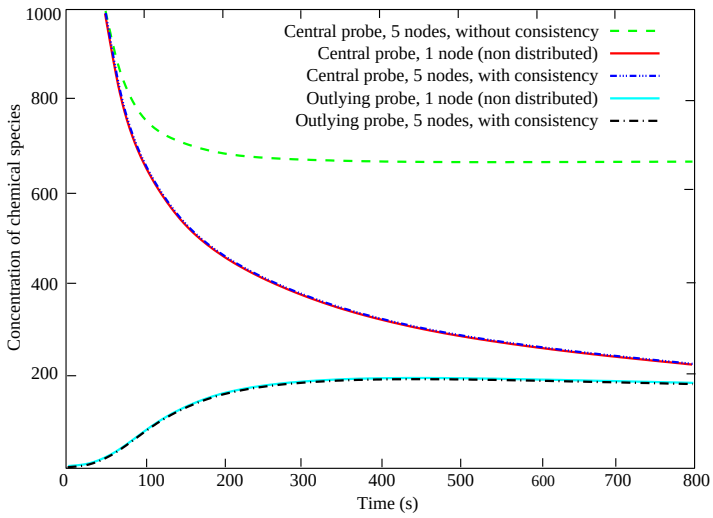

Figure 10. Set of curves representing results for the three configurations. From top to bottom: 1/ central probe measurement when consistency is not active; 2/ central probe measurement in the non-distributed simulation overlaid to central probe measurement in the distributed simulation (first and second cases); 3/ outlying probe measurements in non-distributed and distributed simulations (results are the same, first and second cases).

This model has 10 chemical reactions including a negative feedback. The usual results are the evolutions of the chemical species concentrations. These concentrations oscillate and these oscillations are not deadened during the time. These results are correct in a homogeneous environment. In the presented simulation, we build a heterogeneous environment where one chemical species of the pathway is injected in a local area. This species will be diffused in all the environment during the simulation. Figure 12 shows the result of one particular chemical species for two simulations. In the left part of the figure, it is the simulation carried out on one computer. In the right part, it is the simulation carried out on three computers. The global form of the curve is identical but the amplitude of the first peak is different. This error is caused by some imprecisions of the synchronization due to dead reckoning algorithm.

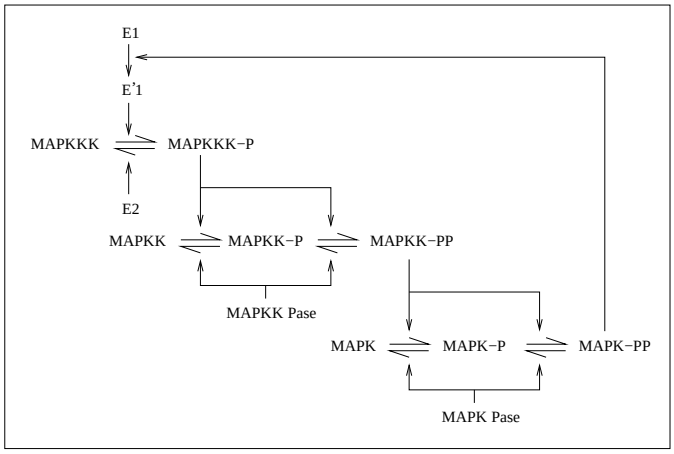

Figure 11. MAPK pathway: Kholodenko's model

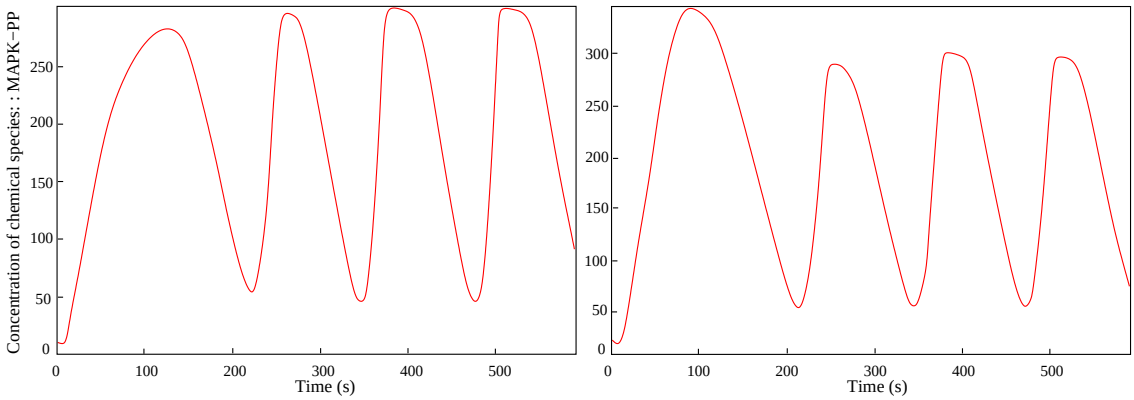

Figure 12. Simulation of the MAPK pathway in a heterogeneous environment (with one computer -left part-, and with three computers with DIVA software -right part-).

6.3 Endothelium

For this virtual reality simulation applied to biology, we work in collaboration with biologists who provide us with data. This simulation corresponds to an *in vitro* experiment in which biologists inject a substance (acetylcholin) in the blood. The substance is caught by endothelial cells. This thin layer of cells that lines the interior surface of blood vessels is called endothelium. Thereafter, a biochemical cascade fires up in the endothelial cells. At the end, endothelial cells drop out another substance (NO), which causes the relaxation of surrounding smooth muscle cells. An application example is described in [23]. In this simulation, three types of cells and blood are implemented. Among the cells, red blood cells only have a moving activity and no other internal mechanisms. On the opposite, smooth muscle cells catch the NO substance and hence expand. And, with regard to endothelial cells, they have the most complex behavior: 15 chemical reactions handling about 15 chemical components are at work inside each cell. The distributed simulation currently takes place on five nodes. We can see on figure 13 many sets of colored boxes (purple, whitish, green and black). Each color represents a different computing node. In spite of quantitative results not similar to *in vitro* experiments (our parameters are not quite

fine-tuned), qualitative results are right: smooth muscle cells are expanding according to the NO quantity produced by endothelial cell.

Hosts	% CPU	Cells	Compartments	Reactions	Diffusion
rodin (demo)	60	0	770	0	2
desmeulles	13	32	393	495	1000
ballet	21	32	393	495	950
redou	18	40	466	520	1156
legal	20	24	345	455	798
5 DIVAs	132	128	2367	1925	3906

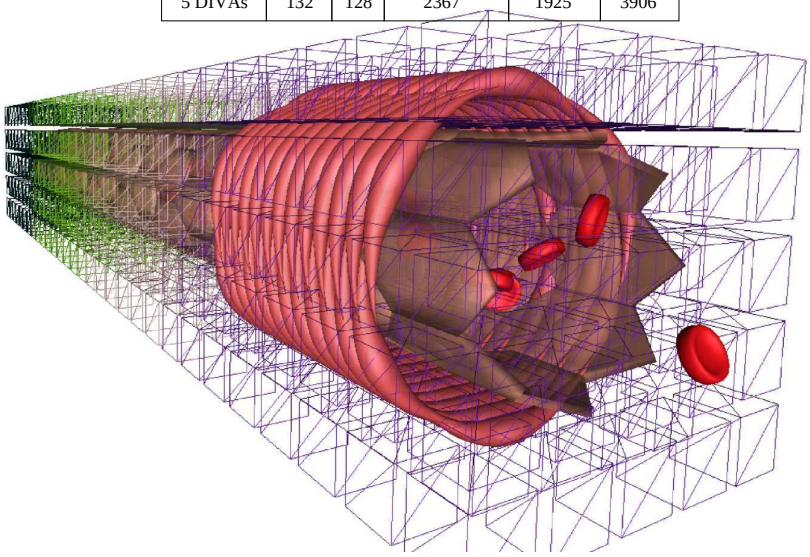

Figure 13. Screenshot of endothelium distributed simulation. The table shows that, thanks to load balancing, the endothelial cells are dispatched approximatively in a equitable way on the 5 nodes.

7 Related work

We presented in subsection 3.1 some Distributed Virtual Reality (DVR) architectures which are mainly based on either ad hoc protocols or high level protocols such as DIS or HLA. With the emergence of grid computing, new approaches arise to design DVR systems and applications. In this section, we detail some previous work dealing with DVR over a Grid (DVR-G) because our contribution uses a Peer-to-Peer architecture in order to distribute a biological virtual environment on a grid.

An a priori molecular Virtual Reality simulation on a grid is presented in [31]. The authors propose to use the computing resources of a grid to simulate 3D molecular structures. During the simulation, atoms positions are stored in a history file. At the end of the simulation, the history file is loaded in a 3D viewer allowing very simple user interactions (i.e displacement, selection of a specific atom type, etc.). Properly speaking, it is not a Virtual Reality application in which the user can interact, in real time, with the virtual world during the simulation.

Some research work concern the use of grid architectures in the field of Virtual Reality applied to Geographic Information System (GIS). In [35] the authors present a DVR-G architecture to build a Distributed Virtual Geographic Environment (DVGE) and to fulfill a 3D GIS which will be used in a collaborative way. This architecture is based on OGSA and web services. It is composed of virtual groups (i.e. Virtual Organizations in OGSA) and a main server maintaining the consistency of a database.

A virtual group contains a set of group clients and a group server. The group clients acquire the scene database only from their group server and the latter obtains this database from the main server. The authors propose to use Globus Toolkit to implement their model because this toolkit allows to develop applications and systems which support collaboration over a grid. We can note that the effectiveness of this semi-centralized approach is not shown. Another Distributed Virtual Geographic Environment system is presented in [55]. The aim of this paper is to provide an internet-based virtual 2D and 3D environment which allows users to share a space in a collaborative way for publishing multidimensional geodata, and for simulating and analyzing geophenomena. The authors choose web services technology to construct a DVGE system which could be deployed on a grid architecture. They designed and implemented geodata and geomodel web services that comply to OWS which is an online geographic information service. The effectiveness of the proposed architecture is shown through a 2D collaborative task using a main server for the calculation of pollutant concentration into a river.

Other applications of DVR-G concern multi-user Virtual Reality walkthrough systems. In [52] the authors propose a grid architecture to support multi-node downloading of distributed virtual environments. In this architecture based on Globus Toolkit, each grid node is a potential server which is able to provide a part of the virtual world. We can note that there is no synchronization protocol to allow multi-user interactions. Paper [26] presents a DVR-G walkthrough architecture and focuses more particularly on scene synchronization in order to keep a virtual world consistent in an efficient way. The authors offer to use multiple servers so as to obtain a scene partitioning algorithm to divide the virtual world into regions. A hierarchy of servers is proposed: ordinary servers and super servers. A super server has several ordinary servers under its responsibility and all objects in a region are managed by the same ordinary server. When an object moves from a region to another, two cases may occur. If the two regions belong to the same super server, the two ordinary servers communicate directly to move the object. Otherwise, a synchronization message is exchanged between the two super servers and the information is then propagated to the ordinary servers. A functional prototype is implemented by using Globus Toolkit and an example is described: a static virtual environment (i.e. virtual buildings) in which two cars which can be driven by remote users. In this example, the number of moving objects seems very low.

One of the closest works to our approach is detailed in [51]. The authors propose a service-oriented framework that can facilitate the design of DVR-G systems. This OGSA-compliant framework is mainly based on a service component called gamelet. The main characteristics of gamelets are mobility and load awareness (CPU, network) which make possible load balancing and migration. The gamelets are also responsible for the virtual world consistency: a gamelet is able to communicate with other gamelets to synchronize their world states. If an object is in an overlapping area managed by several gamelets, each gamelet computes its state separately and a synchronization of the object's position and velocity occurs every 100 ms. A simple two-way synchronization is performed through the Simple Object Acces Protocol (SOAP) and the exchange of XML data. According to the authors, the synchronization scheme has to to be improved for applications that need stringent consistency and response time requirements, like our virtual biological simulations.

Most of previous work on DVR-G, such as walkthrought systems or gamelets, tolerate low frequency synchonization between nodes (i.e. the accuracy of the simulation is not really impacted even if the state of an object diverges between 2 nodes). In distributed *in virtuo* experiments, we have to be as accurate as possible because small errors may generate bias in simulations. Consequently, we need efficient algorithms to maintain consistency between nodes involved in distributed simulations because, in biological simulations a change occurs very often (i.e. between 1 to 10 ms).

8 Conclusion

In the context of Peer-to-Peer distributed simulations over a grid, the consistency of replicated entities and data (with more or less correctness) is one of the main issues, because compromises must be found between the most meticulous consistency and the overload of the computing system (CPU, network, memory). Indeed, consistency mechanism between two remote data requires some network transfer.

We propose a specific implementation of the data consistency algorithm for biological simulations in virtual reality. At first, we described the RéISCOP meta model used to build our biological simulations. Our practical method for the distribution, and especially the "organizational" partitioning, is also based on this model. Let us recall that traditional approaches propose a spatial partitioning, not an "organizational" one. In our method we do not replicate the active elements of the simulation (i.e. *Interactions*), but the passive elements (i.e. *Constituents*). Accordingly, these passive elements have many images. The images are modified independently from one another. On the global simulation point of view, the images must be synchronized to keep their significance or their consistency must be restored.

We established that the sole transfer of replicated data states during strong synchronization (i.e. at each step of the *Interactions* scheduler) did not distort the global simulation. But during weak synchronization, this method was bound to fail. We chose a weak synchronization, because it gives more flexibility to computing nodes and reduces the network load. Therefore, our consistency algorithm is based on the transfer of images states variations. In order to improve the consistency of images, we add a dead reckoning algorithm to it. The aim of this algorithm is to reckon remote state variation of an image and add it to local variation state between two synchronization steps. We also wanted to validate our approach. So, we firstly presented a simulation where only these mechanisms took place. We got identical results for the distributed and non-distributed simulations on a 5-nodes grid. Secondly, we described the simulation of the MAPK cascade. The results of the distributed and non-distributed simulations of this cascade are very similar. At last, thanks to our "organizational" distribution of the RéISCOP meta model, we introduced the simulation of the artery vasorelaxation. The number of entities involved in this distributed simulation is now more important than what can be reached on a single computer. In this application the biological credibility is necessary because this study is the result of a cross-disciplinary collaboration between computer scientists and biologists. The complexity of models in our distributed virtual reality simulations and the complexity of usual mathematical models used in biological simulations (like ODE systems) are at the same level because our modelling - at the mesoscopic scale - is based on them.

Acknowledgements

This project receives a financial support from the French National Research Agency (ANR) within the program Complex Systems and Mathematical Modelling (SYSCOMM): ANR-08-SYSC-002. A special thanks to Mikaël Bourhis for his contribution to this work.

References

[1] G. Ali, N. A. Shaikh, and Z. A. Shaikh. Integration of Grid and Agent Systems to Perform Parallel Computation in a Heterogeneous and Distributed Environment. *Australian Journal of Basic and Applied Sciences, International Network for Scientific Information (INSInet) Publications*, 3(4):3857–3863, October-December 2009.

[2] J. M. Bahi, R. Couturier, and P. Vuillemin. JaceP2P: an Environment for Asynchronous Computations on Peer-to-Peer Networks. In *IEEE International Conference on Cluster Computing, Cluster 2006, 10 pages*, Barcelona (Spain), 25–28 september 2006.

[3] G. Berti, G. Lonsdale, J. G. Schmidt, S. Benkner, D. R. Hose, J. W. Fenner, D. M. Jones, S. E. Middleton, and G. Wollny. Grid simulation services for the medical community. *International Journal of Computational Methods, World Scientific Publishing Compagny*, 5(2):289–317, June 2008.

[4] M. Bourhis, G. Desmeulles, S. Bonneaud, F. Guerrero, and V. Rodin. Data Consistency in Distributed Virtual Reality Simulations applied to Biology. In *Proceedings of the 5th International Conference on Autonomic and Autonomous Systems, ICAS'09, IEEE Press*, pages 154–161, Valencia, (Spain), 20-25 April 2009.

[5] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing, Elsevier*, 61(6):810–837, June 2001.

[6] R. Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD thesis, Monash University, Melbourne (Australia), 12 April 2002. http://www.buyya.com/thesis/thesis.pdf.

[7] J. Calvin, A. Dickens, B. Gaines, P. Metzger, D. Miller, and D. Owen. The SIMNET virtual world architecture. In *Proceedings of the IEEE Virtual Reality Annual International Symposium, VRAIS'93*, pages 450–455, Seattle, WA (USA), 18-22 September 1993.

[8] J. Cao. Performance evaluation of self-organizing agents for grid computing. *Performance Evaluation of Parallel, Distributed and Emergent Systems (Volume 1 in Distributed, Cluster and Grid Computing), Nova Science Publishers*, pages 207–221, 2007.

[9] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Néri, and O. Lodygensky. Computing on Large Scale Distributed Systems: Xtrem Web Architecture, Programming Models, Security, Tests and Convergence with Grid. *Future Generation Computer Systems, Elsevier*, 21(3):417–437, March 2005.

[10] M. Capps, D. McGregor, D. Brutzmam, and M. Zyda. NPSNET: An New Beginning for Dynamically Extensible Virtual Environments. *IEEE Computer Graphics and Applications*, 20(5):12–15, September/October 2000.

[11] A. J. Chakravarti, G. Baumgartner, and M. Lauria. The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network. *IEEE Transaction on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(3):373–384, May 2005.

[12] S. Contassot-Vivier, F. Lombard, J.-M. Nicod, and L. Philippe. Evaluation of the DIET hierarchical metacomputing architecture. *Parallel and Distributed Computing Practices (special issue on Algorithms, Models and Tools for High Performance Computing on Heterogeneous Networks), Nova Science Publishers*, 5(4):64–76, December 2002.

[13] Y. Demazeau. Vowels (Invited lecture). In *1st Iberoamerican Workshop on Distributed Artificial Intelligence and Multi-Agent Systems, IWDAIMAS'96*, Xalapa (Mexico), 25–26 October 1996.

[14] G. Desmeulles, S. Bonneaud, P. Redou, V. Rodin, and J. Tisseau. In virtuo Experiments Based on the Multi-Interaction System Framework : the RéISCOP Meta-Model. *CMES: Computer Modeling in Engineering & Sciences, Tech Science Press*, 47(3):299–330, October 2009.

[15] B. Folliot and P. Sens. Load sharing and fault tolerance manager. In *High Performance Cluster Computing: Architectures and Systems, Chapter 22, Vol. 1, Rajkumar Buyya (editor), Addison-Wesley/Prentice Hall*, pages 534–552, NJ, USA, March 1999.

[16] G. Fortino and W. Russo. Using P2P, GRID and Agent technologies for the development of content distribution networks. *Future Generation Computer Systems (special section: Enhancing content networks with P2P, Grid and Agent technologies), Elsevier*, 24(3):180–190, March 2008.

[17] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *IFIP International Conference on Network and Parallel Computing (IFIP: International Federation for Information Processing), NPC 2005, Beijing (China), 30 November-3 December 2005*. Lecture Notes in Computer Science, Springer, Vol. 3779, pages 2–23, 2005.

[18] I. Foster, N. R. Jennings, and C. Kesselman. Brain Meets Brawn: Why Grid and Agent Need Each Other. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004, IEEE Press*, pages 8–15, New York, NY (USA), 19-23 July 2004.

[19] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications, SAGE Publications*, 11(2):115–128, June 1997.

[20] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, 22 June 2002.

[21] E. Frécon and M. Stenius. DIVE: a scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering (special issue on Distributed Virtual Environments), IOPscience*, 5(3):91–100, September 1998.

[22] L. Gong. JXTA: A Network Programming Environment. *Internet Computing, IEEE*, 5(3):88–95, May/June 2001.

[23] E. Heylen, F. Guerrero, H. Berbari, M. Gilard, B. Saïag, and J. Mansourati. Correlation between reactive hyperaemia and acetylcholine induced vasodilation in rat cutaneous microcirculation. *Atherosclerosis, Elsevier*, 180(2):419–421, June 2005.

[24] IEEE. *Standard for Information Technology : protocols for Distributed Interactive Simulation applications (IEEE 1278)*. IEEE Computer Society Press, 1993.

[25] L. Ji. Computation in Peer-to-Peer Networks. In *Proceedings of the 2002-2003 Graduate Symposium Publication*, Computer Science Dept, University of Saskatchewan (Canada), 10 April 2003.

[26] C. Jiang, X. Xu, J. Wan, W. Li, and X. You. Large Scale Distributed Environments on the Grid: Design, Implementation, and Case Study. In *11th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2007, Melbourne (Australia), 26-28 April 2007*. Lecture Notes in Computer Science, Springer, Vol. 5236, pages 384–395, December 2008.

[27] T. Jinno, T. Kamiya, and M. Nagata. Coordinated Checkpointing using Vector Timestamp in Grid Computing. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2006*, pages 710–716, Las Vegas, NV (USA), 26-29 June 2006.

[28] J. Keller and G. Simon. Toward a Peer-to-Peer Shared Virtual Reality. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops, ICDCSW '02, IEEE Press*, pages 695–700, Vienna (Austria), 2-5 July 2002.

[29] S. Kerdélo, J.-F. Abgrall, M. Parenthoën, and J. Tisseau. Multi-Agent Systems: A Useful Tool For the Modelization and Simulation of the Blood Coagulation Cascade. In *Proceedings of the 1st International Workshop on Bioinformatics and Multi-Agent Systems, BIXMAS 2002, an AAMAS 2002 (1st International Joint Conference on Autonomous Agents and Multiagent Systems) Workshop*, pages 33–36, Bologna (Italy), 15 July 2002.

[30] B. N. Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascade. *European Journal of Biochemistry, Wiley*, 267(6):1583–1588, March 2000.

[31] A. Lagana and O. Gervasi. A Priori Molecular Virtual Reality on EGEE Grid. *International Journal of Quantum Chemistry, Special Issue: 7th European Conference on Computational Chemistry, EUCO-CC7, Venice (Italy), 11-15 September 2008*, 110(2):446–453, February 2010.

[32] X. Laisheng and W. Zhengxia. Multi-polar Autonomous System Grid Resource Scheduling Model and Algorithm Based on Multi-agent and GA. In *Proceedings of the 1st International Conference on Electrical and Control Engineering, ICECE 2010, IEEE Press*, pages 5868–5873, Wuhan (China), 25–27 June 2010.

[33] Z. Li and M. Parashar. Rudder: An agent-based infrastructure for autonomic composition of grid applications. *Multi-agent and Grid Systems, IOS Press*, 1(3):183–195, 2005.

[34] S. W. Liles. *Dynamically extending a networked virtual environment using Bamboo and the High Level Architecture*. Master's thesis, Naval Postgraduate School, Monterey, CA (USA), September 1998. http://handle.dtic.mil/100.2/ADA354468.

[35] T. Lu-Liang and L. Qing-quan. Research on Grid-Based Distributed Virtual Reality. In *Proceedings of the International Conference on Wireless Communications Networking and Mobile Computing, WCNM 2005, IEEE Press*, volume 2, pages 1316–1319, Wuhan (China), 23-26 September 2005.

[36] M. R. Macedonia and M. J. Zyda. A Taxonomy for Networked Virtual Environments. *IEEE Multimedia*, 4(1):48–56, January-March 1997.

[37] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz. NPSNET: A network software architecture for large-scale virtual environments. *Presence*, 3(4):265–287, Fall 1994.

[38] D. C. Miller, A. R. Pope, and R. M. Waters. Long-Haul Networking of Simulators. In *Proceedings of the 10th Interservice/Industry Training Systems Conference*, pages 577–582, Orlando, FL (USA), 29 November-1 December 1988.

[39] G. Moro, A. M. Ouksel, and C. Sartori. Agents and Peer-to-Peer Computing: A Promising Combination of Paradigms. In *1st International Workshop Agents and Peer-to-Peer Computing, AP2PC 2002, Bologna (Italy), July 15 2002*. Lecture Notes in Computer Science, Springer, Vol. 2530, pages 15–28, 2003.

[40] A. Nguyen-Tuong. *Integrating Fault-Tolerance Techniques in Grid Applications*. PhD thesis, Faculty of the School of Engineering and Applied Science at the University of Virginia, Charlottesville, Virginia, USA, August 2000. http://www.cs.virginia.edu/∼an7s/publications/thesis/thesis.pdf.

[41] G. Querrec, R. Bataille, V. Rodin, J.-F. Abgrall, and J. Tisseau. Computer Simulation of Multiple Myeloma in the Context of Systems Biology. In *Proceedings of the 10th International Myeloma Workshop, supplement of Haematologica, the hematology journal, Vol. 90, No. 1*, pages 92–93, Sydney (Australia), 10–14 April 2005.

[42] O. F. Rana and L. Moreau. Issues in Building Agent-Based Computational Grids. In *Proceedings of the 3rd Workshop of the UK Special Interest Group on Multi-Agent Systems, UKMAS'2000, 11 pages*, Oxford (UK), 15 December 2000.
[43] S. Singhal and S. Zyda. *Networked Virtual Environments.* Siggraph Series. ACM Press Books, 1999.
[44] S. K. Singhal and D. R. Cheriton. Exploiting Position History for Efficient Remote Rendering in Networked Virtual Reality. *Presence: Teleoperators and Virtual Environments*, 4(2):169–193, Spring 1995.
[45] J. Tang and M. Zhang. An Agent-based Peer-to-Peer Grid Computing Architecture: Convergence of Grid and Peer-to-Peer Computing. In *Proceedings of the 4th Australasian Symposium on Grid Computing and e-research, AusGrid 2006 (Australasian Computer Science Week), Conferences in Research and Practice in Information Technology (CRPIT) series, Australian Computer Society Inc., Vol. 54*, pages 33–39, Hobart, Tasmania (Australia), 16-19 January 2006.
[46] H. Tianfield and R. Unland. Towards self-organization in multi-agent systems and Grid computing. *Multiagent and Grid Systems, IOS Press*, 1(2):89–95, 2005.
[47] J. Tisseau. *Virtual Reality* –in virtuo *autonomy*–. Accredition to Direct Research in Computer Science, University of Rennes I, Rennes (France), 6 December 2001. http://www.enib.fr/∼tisseau/doc/hdr/hdrJTuk.pdf.
[48] P. Torguet and R. Caubet. VIPER (VIrtuality Programming EnviRonment): A virtual reality applications design platform. In *Proceedings of the 2nd Eurographics Workshop on Virtual Environments, 8 pages*, Monte Carlo (Monaco), 31 January-1 February 1995.
[49] U.S. Department of Defense. High Level Architecture Interface Specification – Version 1.3. *IEEE P1516.1, Standard for Modeling and Simulation (M&S)*, 20 April 1998.
[50] J. Wang, Q.-Y. Wu, D. Zheng, and Y. Jia. Agent based Load Balancing Model for Service based Grid Applications. In *Proceedings of the International Conference on Computational Intelligence and Security, CIS 2006, IEEE Press*, pages 486–491, Guangzhou (China), 3-6 November 2006.
[51] T. Wang, C.-L. Wang, and F. C. M. Lau. An architecture to support scalable distributed virtual environment systems on grid. *Journal of supercomputing, Special Issue on New Trends in Parallel and Distributed Computing and Networking, Springer*, 36(3):249–264, June 2006.
[52] Y. Wang, G. Lu, J. Jia, and H. Yang. An Architecture to Support Multi-node Downloading of Distributed Virtual Environment on Grid. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control, ICNSC'07*, pages 602–607, London (UK), 15-17 April 2007.
[53] M. Wooldridge and P. Ciancarini. Agent-Oriented Software Engineering: The State of the Art. In *1st International Workshop on Agent-Oriented Software Engineering, AOSE 2000, Limerick (Ireland), 10 June 2000*. Lecture Notes in Computer Science, Springer, Vol. 1957, pages 1–28, January 2001.
[54] J. W. Yin, W. Y. Zhang, Y. Li, and H. W. Chen. A peer-to-peer-based multi-agent framework for decentralized grid workflow management in collaborative design. *International Journal of Advanced Manufacturing Technology, Springer*, 41(3–4):407–420, March 2009.
[55] J. Zhang, J. Gong, H. Lin, G. Wang, J. Huang, J. Zhu, B. Xu, and J. Teng. Design and development of Distributed Virtual Geographic Environment system based on web service. *Information Sciences, Elsevier*, 177(19):3968–3980, October 2007.

Authors' biographies

Vincent RODIN was born in 1966. Professor at University of Brest (France), he is working on image processing, multi-agent systems and computer simulation of biological processes.

Gireg DESMEULLES was born in 1979. Lecturer at Brest National Engineering School (France), he is working on virtual reality, multi-interaction paradigm and in virtuo experimentation.

Pascal BALLET was born in 1971. He has studied at Brest National Engineering School (France). He is currently lecturer at University of Brest (France) and he is working on multi-agent systems and computer simulation of biological processes.

Pascal REDOU was born in 1971. Lecturer in applied mathematics at Brest National Engineering school (France), he is working on consistence and stability of multi-agent algorithms, and on coupling differential and multi-agent systems.

Christophe LE GAL was born in 1972. He is associate professor in computer sciences at Brest National Engineering School (France), and COO of CERVVAL company. His domain of interest is numerical simulation methods for virtual reality.