Optimization of Distributed Systems Using Multi-Agent Systems with Virtual Time

Ioana Alexandra Pandele
Department of Mathematics and Computer Science,
Faculty of Sciences, "Vasile Alecsandri" University of Bacău
Calea Mărăşeşti, 157, 600115, Bacău, Romania
ioanapandele@yahoo.com

Alina Mihaela Patriciu
Department of Mathematics and Computer Science,
Faculty of Sciences, "Vasile Alecsandri" University of Bacău
Calea Mărăşeşti, 157, 600115, Bacău, Romania
alina_patriciu1980@yahoo.co.uk

Abstract

The fusion of Artificial Intelligence and Real-Time areas has proven to be quite a clever movement, similar to the movement associated to check in chess (lets not go that far as checkmate, considering that the area of informatics is quite slippery when it comes to updates). Artificial Intelligence provides new possibilities to the Real-Time systems. However, this approach has shown important difficulties[2]. Mainly, the Real-Time systems have temporal requirements (they usually require predictable response times) that are not usual in Artificial Intelligence techniques.

One of the ways to solve this problem is the development of software architectures. These software architectures are used to design intelligent agents that work in real-time environments. These architectures have several mechanisms to allow the agents to work in real time environments offering reactive behavior (to fulfil the temporal requirements) and deliberative behavior [2].

However, in Distributed Systems, although the notion of *global time* plays an important role, it is hard to realize; at first sight even the definition of the very term itself is not clear at all. This paper is an extended work of [6].

Keywords: intelligent agents, virtual time, multi-agent system, distributed systems

1. Introduction

The measure of time is a very important issue in *distributed systems*. The same issue (i.e. *time*) is an important characteristic in centralized systems too, but it is not considered a critical one because of the fact that there is no problem not getting to synchronize all the processes, meaning there is not a *must have* for the existence of a unique reference: a shared clock. Instead, in the case of *distributed systems* each computer has its own clock and it is not an easy process to synchronize each and every clock of the computers that arrange the entire system.

As we have mentioned in the beginning, the issue of time is very important/vital, therefore, it is very important to know the precise hour an event happens at for each component computer. The required precision, concerning the *time* component, varies from case to case, the *real time systems* being the ones that would admonish a major precision. Nevertheless, it necessary to make arrangements so that time synchronization exists, in general decision systems as well.

Considering the fact that the measuring of time is an important factor in our way of thinking, and the effect of not having synchronized clocks could be a dramatic one, we would like to particularize upon several methods of clocks synchronization of all the components of the distributed system (i.e. computers) or, which is almost the same thing, that makes all the computers have the same hour simultaneously. This is known as *the synchronization of physical clocks*.

2. Clocks and Synchronization

Properties of Distributed Systems

- The relevant information is distributed among several machines.
- The processes take the decisions only on the basis of the local available information.

- It is a must to avoid a fail point in the system.
- There is no common clock or any other different kind of certain global time resource.

Each computer has its own physical clock. This clock is an electronic device based on a quartz crystal that oscillates at a certain frequency and, further more, can organize itself so that it could generate interruptions at certain points in time (very certain number of oscillations). Knowing the precise time when these interruptions occur (called clock interruptions), we could make use of them so that the time could be called well administrated. For this purpose, the operative systems use a counter that increases with the clock interruptions.

The clocks based on a crystal of quartz are liable to a "by-pass"; in other words, they measure time at certain speeds, thereby the "distances" between *each* (*particular*) measured time keep on widening. No matter how small the oscillation period of each clock is, the accumulated differences in several oscillations lead to a significant/considerable difference. The leeway speed for this kind of clocks (i.e. clocks with quartz crystal) maintains around the value of10^{-6}, which represents a difference of a second every 11, 6 days.

The most precise clocks are based on atomic oscillators, whose precision reach 10^{14} (a second in 1.400.000 years). Certain laboratories (with atomic clocks) from all over the world periodically indicate the number of ticks (clock interruptions) to the *Bureau Internationale de l'Heure* (BIH) that calculates the medium value producing *The International Atomic Time* (IAT), also taking into consideration the fact that time started being counted on the 1st of January 1958, once the second has been defined as the necessary time for a Cesium atom to realize 9.192.631.770 transitions. Nevertheless, time units that we use have an astronomic origin (universal time), in other words, they are defined as functions of rotation periods and translation periods of Terra around the Sun. This means that the atomic time and the universal one do not synchronize.

Regarding Distributed Systems, it is impetuously necessary to synchronize the different teams that make up the main project, **than** to synchronize with the *Universal Coordinated Time*.

Two of the most important algorithms that are used in the synchronization process are: *Cristian's Algorithm* and *Berkeley's Algorithm*.

Cristian's Algorithm (figure 1) is a method for clock synchronization which can be used in many fields of distributive computer science. It suffers, **though**, in implementations using a single server, making it **unsuitable** for many **distributive applications** where redundancy may be crucial. [4]

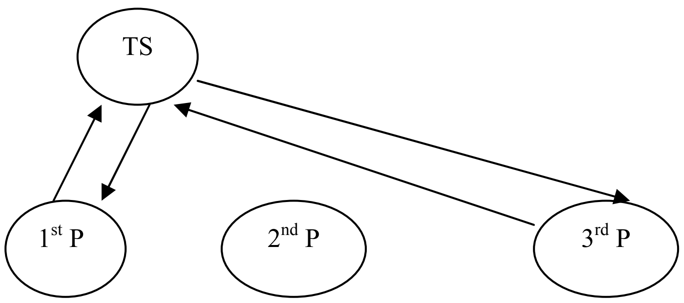

Figure 1. Cristian's Algorithm

- TS(Time Server) receives time solicitations and responds as fast as possible
- What does the solicitor do once it obtains its response/answer?

a) Set up/Adjust its clock with C_{UCT} that TS sends
b) Gradually adjust its clock with C_{UCT}

Instead of adding 1 to C_P (where C_P represents the Clock for each process P), with every tick of the clock it adds a greater value (if $C_P < C_{UCT}$) or a smaller one (if $C_P > C_{UCT}$)

Berkeley's Algorithm is more suitable for systems where a radio clock is not present; this system has no way of making sure of the actual time other than by maintaining a global average time as the global time. A time server will periodically fetch the time from all the time clients, average the results, and then report back to the clients the adjustment that needs to be made to their local clocks to achieve the average. This algorithm highlights the fact that internal clocks may vary not only in the time they contain but also in the clock rate. Any client whose clock differs by a value outside of a given tolerance is often disregarded when averaging the results. This prevents the overall system time from being drastically skewed due to one erroneous clock. [4]

- It is based on an Active Time Server that realizes a periodic "polling" to each machine, in order to ask its characteristic time
- It is useful, although there is no receiver machine for www

The time must be manually set up for the operator.

But, as we have mentioned before, in **distributed systems** the main role, concerning s**ynchronization**, is played by the **succession of events.** Thus, it is more suitable to choose *virtual time* instead of real time. The main difference between *virtual time* and *real time* seems to be that virtual time is only identifiable by the succession of events.

The concept of virtual time for distributed systems was tackled for the first time by Lamport in 1978 [3]. Lamport signalled that synchronization does not have to be absolute. First, if two processes interact, it is not necessary that their clocks be synchronized with a certain precise hour, because the lack of synchronization will not be observed and will not create any problems. But they have to agree with the order imposed by the logical succession of events.

For certain environments, where the precise synchronization with a certain referential hour (UCT) is not a must, instead of working with physical clocks *logical clocks* are chosen, whose main characteristic is given by the order of events, and not by the measure of the exact moment when the precise event happens.

It is very difficult to perfectly synchronize multiple distributed clocks; it is often not necessary to synchronize them (i.e. distributed clocks) with a UCT hour, or, furthermore, two processes that do not interact do not have to be synchronized. Therefore, in the end, it is necessary that the processes agree with the order in which the events happen/proceed, in order to solve the processes.

So, instead of working with physical clocks, it has been chosen to work with the logical ones.

In order to synchronize logical clocks, Lamport [3] defined the relationship "**happened before**" (*preceded*) which implies that the expression $a_1 \rightarrow a_2$ means "a_1 occurred before a_2", and it means that all the processes coincide in the fact that a_1 took place first, and subsequently a_2 took place. This relation can be directly observed in two situations (figure 2):

1. If two events happen during the same process, the order of the happening is indicated by the common clock;
2. When two processes communicate through a message, the event that corresponds to sending the precise message always happens before the event of receiving it (i.e. the message).

If two events, a_1 and a_2, are produced in different processes that do not exchange messages (neither directly nor indirectly), then it is not certain if $a_1 \rightarrow a_2$ or $a_2 \rightarrow a_1$. In this case it is said that these events are competitive, which means that it is not known which one happened first (and it is not a must-know thing either).

We should note that the expression $a_1 \rightarrow a_2$ is transitive; this means that:

if $a_1 \rightarrow a_2$ and $a_2 \rightarrow a_3$ then $a_1 \rightarrow a_3$

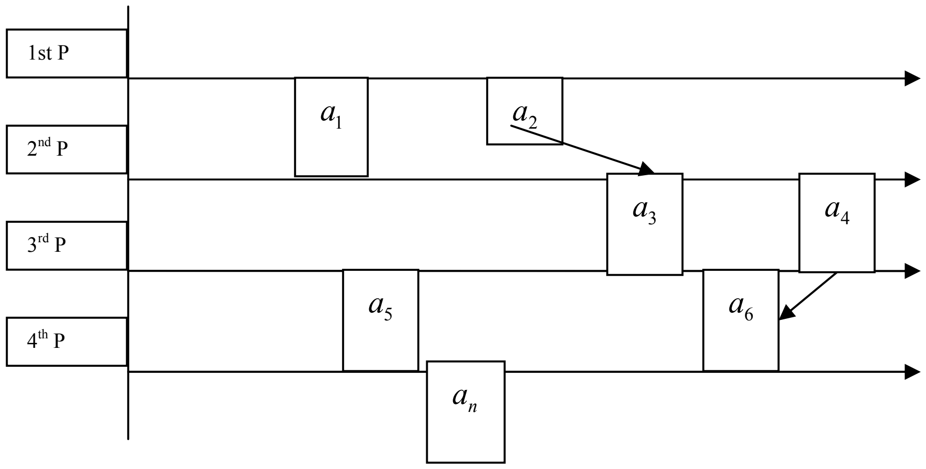

Figure 2. Lamport's algorithm

We must find a way of assigning values for the logical clocks C so that:

- If $a_1 \rightarrow a_2 \Rightarrow C(a_1) < C(a_2)$
- Clock C must always go ahead and never backwards

And at this point we would resort to *intelligent agents*.

According to Lamport, for each computer there is a logic clock C, that is used to mark up the time of each event that takes place; thus, $C_P(a_i)$ indicates the correspondent time for the event a_i in the process P.

Therefore:

1. C_p is being incremented with a unit before any event produces in a process; in other words: $C_P = C_P + 1$.
2. a) When a process *P* sends a message *m*, the value of the message becomes $t = C_P$
   b) When the process *Q* receives the message (*m, t*), $C_Q := \max(C_Q, t)$ is being calculated, and applies the first step before the event of receiving the message has been marked

3. Are Intelligent Agents a Solution?

The aim of this paper is to bring together particular elements from domains such as Artificial Intelligence, Physics and Mathematics, in order to optimize distributed systems. Distributed Artificial Intelligence represents a real challenge concerning interdisciplinary research. The development of Multi Agent Systems represents the main subject of Distributed Artificial Intelligence.

Therefore, we opt for intelligent agents to perform the events that make up each process. A schema of these intelligent agents, that perform those events, is going to be further "depicted", following that a real implementation of the script be revealed in a sequel of this article, in the near future.

According to Katia Sycara [5] a multiagent system represents a network of united agents, which interact in order to solve problems that surpass the capacities of individual agents.

In what we like to think as a future project, we tend for a micro-social organization of the agents combined with the organization based on groups of agents, knowing the fact that the main characteristic of micro-social organization is based on the establishment of a clear agent's hierarchy, and in those based on groups of agents, agents with different tasks are grouped into autonomous informatics entities. We would like to join those two organizations because there has to be a MASTER that coordinates, from the shadow sometimes, the main project (its presence would be distinguishable just in key moments) but we still have to synchronize the events, which means there has to be a certain communication among the agents. The authority is given to a single agent that is symbolically situated on the top level; this agent would be considered the root of an arborescent structure; but the stages of the process are provided by groups of agents, respectively by the relations of communication between them.

Thus, we would deal with a ZERO AGENT and several groups of agents. Such a group is made up of several agents, each and one of these agents accomplishing a certain role in the process of synchronization (figure 3).

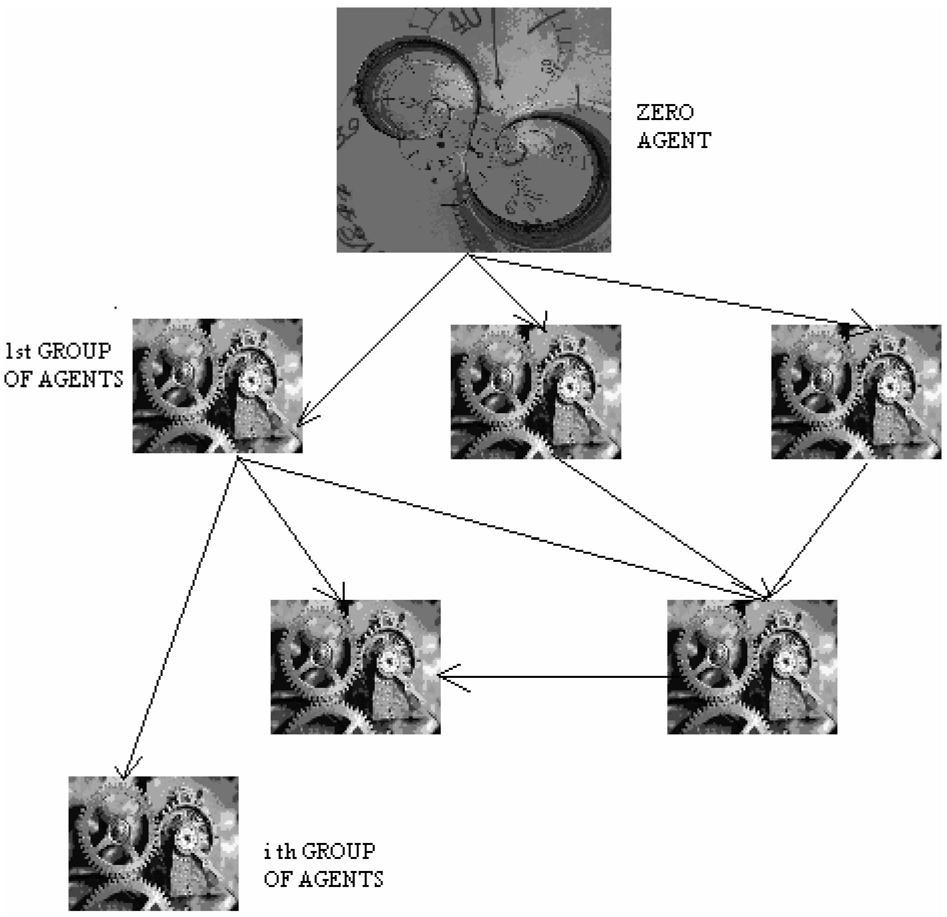

Figure 3. The multi-agent system

A crucial role in the entire process of interrelation among the groups of agents is played by the *negotiation between agents*. Negotiation appears when agents might have different goals and each agent/group of agents tends to maximize its own good, seconding global "well-being".

As far as the role of Mathematics in the optimization of distributed systems, we could appeal to *Minkowski's relativistic space-time model*. The use of this tool represents a fineness problem (it would be the final part of the ENTIRE process (i.e. optimization problem/process)). Because of the finite speed of light, Minkowski's well-known relativistic space-time model may reflect reality more accurately than the standard model of time. [1]

4. Conclusions

A fair and straight answer to the question: "Are Intelligent Agents a Solution?" would be given once a characteristic algorithm had been implemented and tested.

One thing is certain for sure: Mathematics, Physics and Computer Science can become magnificent tools in the hands of a proper master.

5. References

[1] Friedemann, Mattern (2008). Virtual Time and Global States of Distributed Systems. [Electronic version]. Department of Computer Science, University of Kaiserslautem D 6750 Kaiserslautern, Germany.
[2] Lopez, L. H. (2008). Heuristicas para el control deliberativo en una arquitectura de agentes inteligentes en tiempo real. Doctoral thesis at Universidad Politecnica De Valencia Departamento De Sistemas Informaticos y Computacion,
[3] Lamport, L. (1978). Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(7), 558-565.
[4] Cardinale, Y. (n.d.). Sincronizacion En Sistemas Distribuidos. Sistemas de operacion II [Electronic version].
[5] Sycara, K., Decker, K., Pannu, A., D., Williamson, M. & Zeng, D. (n.d.). Distributed Intelligent Agents. [Electronic version]. The Robotics Institute, Carnegie Mellon, University Pittsburgh, PA 15213, U.S.A.
[6] Pandele, I. A., Patriciu, A. M. (2009). Virtual Time – Solution To The Optimization Of Distributed Systems? In: *Knowledge Engineering: Principles And Techniques. Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques*, KEPT2009 (pp. 25–28), Cluj-Napoca, Romania, July 3–4, 2009.