Abstract

In this paper is presented an overview of grid types. An addition is presented to the ambient intelligence model. Middleware approaches are studied in regard to this model.

I. INTRODUCTION

According to Bell’s Law of Computer Classes [1], a new generation of computers emerges every 10 years. This computer classes are grouped into waves. In the first wave, the meaning was centered on a single device, like mainframes; while in the 1990s, the second wave meaning was in the connection. Computation and information access in a distributed environment were the central point of interest. The third wave brought the collection of devices and their reactive interface into focus.

Following this scheme, we shall take a look at networks and their composition.

I.1. Sensor networks

A sensor network is a group of specialized transducers with a communications infrastructure intended to monitor and record conditions at diverse locations.

A sensor network [2] consists of multiple detection stations called sensor nodes, each of which is small, lightweight and portable. Every sensor node is equipped with a transducer, microcomputer, transceiver and power source. The transducer generates electrical signals based on sensed physical effects and phenomena. The microcomputer processes and stores the sensor output. The transceiver, which can be hard-wired or wireless, receives commands from another computer and transmits data to that computer. The power for each sensor node is derived from the electric utility or from a battery.

I.2. Multimedia networks

Multimedia content is playing an increasing part in business and private communications. The term ‘multimedia’ refers to a spectrum of media classes used to represent information.

Multimedia traffic represents the transmission of data acting as different media over communication networks. These networks offer a best-effort service and should be able to meet the delivery requirements of interactive applications. The quality of service metric is of critical importance, and should be tailored to the delivery of multimedia content.

I.3. Computational grids

The average computing environment is still inadequate for computationally sophisticated purposes such as predicting the outcome of complex actions or selecting from among many choices. This is the reason for which supercomputers have continued to evolve. The term computational grid was adopted for the infrastructure that will enable the increases in computation discussed above.

The classical definition of computational grid is given in [3], as being a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.

The term infrastructure is used because computational grid is concerned with large-scale pooling of resources, whether compute cycles, data, sensors, or people.

The pooling process implies important significant hardware infrastructure in order to achieve the necessary interconnections and software infrastructure to monitor and control the resulting ensemble.

Grid computing appears to be a promising trend for three main reasons [3]:

1) its ability to make more cost-effective use of a given amount of computer resources,

2) as a way to offer solutions to problems that can’t be solved without an enormous amount of computing power,

3) it suggests that the resources of different computers can be cooperatively harnessed and managed as collaboration in order to accomplish a common goal.

I.4. Sensor Grids

In the last decade there has been a growth of the data-centric nature of grid computing applications. With the growth of sensor networking technologies as a different research area, scientists have worked on integrating [4] the sensors and actuators with general purpose computing systems.

Integrating sensor networks with computer grids is like giving “eyes” and “ears” to the computational grid [5]. The processing and modeling of real-time data with the power of a computational grid permits almost instantaneous response and decision on a large scale. Standard applications range from environment
monitoring in order to be able to warn at the apparition of a natural disaster and to missile detection, tracking and interception.

The idea of Service-Oriented Architecture (SOA) on the computational grid is nothing new, but in the context of sensor grids, this approach allows not only the discovery, access and sharing of the services but also the sharing of the sensor-actuator infrastructure among a number of different applications and users.

*  

**1.5. Multimedia Grids**

In [6] is presented Multimedia Grid (mmGrid) as an extensible middleware architecture supporting multimedia applications in a grid computing environment. The idea of this project is to provide support for applications from domains like graphics, visualization, streaming media and tele-immersion and also to offer a mechanism for provisioning computing resources. The scheduling system is intended to be flexible, while the interactive and batch jobs have the capability to use grid-computing paradigm. Since mmGrid is focusing on graphics, visualization and streaming media applications and, in future, tele-immersion, the advance reservation of network bandwidth is a key requirement for good performance. Besides this, other basic requirements are the ability to submit batch jobs and to reserve a workstation for interactive applications.

**1.6. Ambient Intelligence**

A new paradigm – shifting technology [7] started to emerge in the late years, as a third wave of computing and it represents a new way for the interaction between the electronics and the human individuals. As previously said, the third wave brought the collection of devices and their reactive interface into focus. Some interesting features implied by these are the implicit resistance to failure (if a component fails, the goal can still be accomplished) and the inclusion of goals and constraints into the interface.

The name that designates this third wave is “ambient intelligence” and is described [8] as a sensitive, adaptive, and responsive to the presence of people and objects environment where technology is embedded, hidden in the background and augments activities through smart non-explicit assistance. This environment should also preserve the security, privacy and trustworthiness while utilizing information when needed and appropriate. The common idea is that the human is in the center, the electronics invisibly in the background.

According to [7, 9, 10] there is a certain potential in merging sensor and actuator networks with multimedia networks, the result being a state-of-the-art ambient intelligence. Being such a new paradigm it poses problems starting with ultra-low-cost, ultra low power electronics for sensing and ending with ultrahigh-speed signal processing for enhanced visual experiences. From an economic point of view, the concept is highly alluring, because the demand is increasing in all the sub-markets computing, communications and consumer electronics.

## II. Non-Traditional Parallel Model

As previously presented, the tendency is evolving from mixing all kinds of equipments into mixing all kinds of networks, and extending functionality and reliability.

Ambient intelligence is designed to be used into an environment suitable to be controlled; therefore its primary target is the “smart” home. [7] Into such an environment, besides the sensor, actuators and multimedia processing another participant might appear: the PCs. This participant is not taken into consideration in the ambient intelligence approach.

On the other hand, today the multimedia ambient concept is centered on a PC approach (e.g. Microsoft Media Center, Apple FrontRow, and HP Digital Entertainment Center). This PC controls the network. Providing the case this center fails, the whole network is down. The user is the system and configuration manager making this concept still connection and device oriented.

The idea behind our approach is that the PC is just another node in the network. It can move (spatially), it can disappear or appear at will, or it can sleep. When it is awake and the human user interacts with it, it can be seen as a merged sensor-actuator entity. When it is awake it provides computational power, therefore the network can use it to process operations. Doing so, other nodes in the network can sleep, therefore reducing overall power consumption.

A comparison of all the network and grids previously discussed is presented in table 1.

<table>
<thead>
<tr>
<th>Network</th>
<th>Grid</th>
<th>Sensor grid</th>
<th>Ambient intelligence</th>
<th>Multimedia grid</th>
<th>Overall grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor network</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multimedia network</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computation network</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is clearly shown that this “overall” grid includes not only the computation, but also the multimedia and sensor networks, thus being able to exhibit ambient intelligence and computation, virtually in the same time.

The levels of this proposed model are illustrated in figure 1. The pieces of equipment (sensor, actuators, controllers, PCs, DSPs) though extremely heterogeneous, are all grouped into the device level. The links between all this devices, as well as the association to the Internet are at the connection level, while the collection level is composed not only of streaming data sources but also of all the networks and their capabilities. In the same figure, the individual networks are also denoted, in order to clearly show the composition of networks into a higher, reactive network.

Studying this approach from the power, cost and size style, due to its heterogeneous nature, this model must be assessed by device: the sensors, actuators, controllers and DSPs are already very well on the way of becoming “disappearing electronics”. Many solutions [11] emerged in the last years, though lately the idea of energy
The main problem in “disappearing electronics” is that these super low-power devices should need no batteries or outside power supply, instead relying upon microgenerators. Thus, the area of nanogenerators is blossoming. Two main areas are developing in this area: nanoscale thermoelectric energy harvesting [14] and Nano-Piezotronics[15, 16].

This leaves the PCs to be analyzed from the power, cost and size point of view. The PCs are forecasted to disappear for a while now [17], and every major company (for example IBM and Microsoft [18]) is preparing for it. The computer of the future may have only an interface with the user, the main computation and tasks being located on a server, probably shared with other users.

Security and privacy [19, 20] is a thorny problem in this type of networks, due to its inherent mix and distributed nature.

Reliability [30] is another major problem, due to the fact that unreliability is inherent to the disappearing electronics concept. This is primary caused by the fact that nodes may emerge unexpectedly, may move, may fail and may finish their energy reserves (temporary or not). All this problems are forced into focus by the cost, power and size constraints. The solution used in ambient intelligence in order to achieve reliability is redundancy [10, 7]; therefore we expect it to be also a solution in this type of grid.

Portability, scalability and configurability are another problem. Changing of the application usually means changing the software, which, of course, implies everything to fall apart. A real solution is considered to be the raising of the abstraction level [10].

This topic will be addressed further into this paper.

II.1. A Middleware Discussion

During the programming of any networks, first of all is a significant need for programming abstractions that simplify tasking, and for a middleware that supports such programming abstractions [21]. However, we must take into account the fact that our grid is composed of three distinct types of network, with their specific requirements. A summary of the programming requirements for each individual type of network is presented in table 2.

<table>
<thead>
<tr>
<th>Programming requirements</th>
<th>Sensor</th>
<th>Multimedia</th>
<th>Computatio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concealed issues</td>
<td>hardware and distribution</td>
<td>hardware</td>
<td>distribution</td>
</tr>
<tr>
<td>Restricted Resources</td>
<td>Energy</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>computing power</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>communication bandwidth</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Network Dynamics</td>
<td>high</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>Scale of Deployments</td>
<td>$N^*(100\ldots100)$</td>
<td>$N^*(10\ldots10)$</td>
<td>$N^*(10\ldots10)$</td>
</tr>
<tr>
<td>Real-world Integration</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Time scale</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Location scale</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Collection and Processing of Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preprocessing</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Aggregating data</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Local processing</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

In choosing a middleware we must see if it can support all the requirements in their worse case scenario.

There are already a series of middleware for every type of network, and in table 3 we present the types of approaches.

<table>
<thead>
<tr>
<th>Type of approach</th>
<th>Sensor</th>
<th>Multimedia</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Remote Procedure Call</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Object Broker Request</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Message-oriented Databases</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Mobile Agents</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

It can be noticed that the only approach for middleware that covers all the angles is mobile agents.

Among all the agent-oriented middlewares in use today the most widespread is JADE (Java Agent DEvelopment framework).

JADE is a completely distributed middleware system [28] with a flexible infrastructure allowing easy extension with add-on modules. The framework facilitates the development of complete agent-based applications by means of a run-time environment implementing the lifecycle support features required by agents, the core logic of agents themselves, and a rich suite of graphical tools.

As JADE is written completely in Java, it benefits from the huge set of language features and third-party libraries.
on offer, and thus offers a rich set of programming abstractions allowing developers to construct JADE multi-agent systems with relatively minimal expertise in agent theory.

JADE was initially developed by the Research & Development department of Telecom Italia s.p.a., but is now a community project and distributed as open source under the LGPL license.

JADE is consistent with the FIPA specifications and IEEE standards, thus using different content languages and managing of conversations through predefined interaction protocols.

**II.1. Intelligent agents**

Using intelligent agents allows us to look at the controller/DSP/PC as an agent, with all the implications that the artificial intelligence includes. The agents have several important features like autonomy, proactiveness and an ability to communicate. This allows them to execute complex, and often long-term, tasks and to initiate a task even without an explicit stimulus from a user. The communication allows an agent to interact with other agents in order to accomplish its own agenda.

An area in which concepts from the area of artificial intelligence are passed into the field of distributed systems is Agent-Oriented Programming (AOP).

The application in AOP is a collection of components called agents. The communication between agents is intrinsically peer to peer.

Agent technology has been the subject of extensive discussion and investigation [28] within the scientific community for several years, but it is perhaps only recently that it has seen any significant degree of exploitation in commercial applications.

Multi-agent systems are being used in an increasingly wide variety of applications, ranging from comparatively small systems for personal assistance to open, complex, mission-critical systems for industrial applications. Examples of industrial domains where multi-agent systems have been fruitfully employed include process control, system diagnostics, manufacturing, transportation logistics and network management.

Thus, the architecture of the overall grid as described in figure 1 on which the concept of intelligent agents is imposed is described in figure 2.

**III. Implementation**

In implementing the architecture presented in figure 2, the first stage is the simulation.

In the simulation phase we wanted to assess how JADE behaves comparing to other middleware. Because the interesting part in the “overall” grid is the computation power added to ambient intelligence we wanted to see if a typical computational middleware, MPI, is better than JADE.

In order to do this, we modelled a simple problem. There are a number of computational nodes which process a number of inputs (S). The processing mode is very simple (of polynomial complexity).

As expected, the results when implemented with MPI, depicted in figure 3 are of exponential growth.

**Figure 3. Results when implemented with MPI**

It can be seen that when the number of sensor inputs needing to be considered escalates, the overall time increases, and when the number of sensor inputs is constant, the communication among the computation nodes does increase with their number.

We can dispute the fact that this increase in time even when S is constant is due to the scalability of this
problem. This is not the purpose of this paper. The goal was to study the modality in which a computationally classical type of middleware faces a new-age middleware like JADE.

In this regard, the same problem was implemented with JADE agents who launch CFP (call for proposal) message every time they need a value from a sensor. The agent that has the sensor directly connected replies with a PROPOSE of the value and if that value is accepted the initiating agent confirms that it received the value.

The program was executed with one agent on every computational node, so that the comparison with MPI would be fair.

The comparative results between the MPI and JADE implementation, for a constant number of inputs (40) are presented in figure 4.

It can be observed that for a few computation nodes, JADE overall time of execution is clearly worse than MPI. At 25 computation nodes, though, the overhead of communication in the MPI version allows the JADE version to win.

For a simple problem, functioning on a relatively large number of computation nodes and required to intensively communicate with other nodes, the JADE version was expected to be slower than the MPI. When the communication was over heading the classical solution, JADE proved to be faster, thus proving to be at least an interesting choice in implementing future “overall” type of grids. Also, the portability of the code makes JADE an interesting option.

**CONCLUSIONS**

This paper offers an overview of the current trends in grid computing and addresses the idea of an “overall” grid.

By transferring concepts from the existing models into this new “overall” grid concept, an architecture based on intelligent agents emerges. Using such a level of abstraction many tasks, specific to this mixed type of networks, become easy to implement.

A practical solution in implementing this architecture was tested, in comparison to a classical implementation and the result prove that a JADE-based solution can be a viable option.

**REFERENCES**


