# FIPA-OS AGENTS APPLIED TO PROCESS SCHEDULING IN REAL TIME MONITORING

Angel Gómez, Diego Cantorna, Carlos Dafonte, Bernardino Arcay
*Dept. of Information and Communications Technologies, University of A Coruña, Faculty of Informatics, A Coruña, Spain*

Keywords: Intelligent agents, Telemedicine, Control systems, Hospital information systems

Abstract: This work presents a mechanism for the management of network tasks, based on the technology of Intelligent Agents applied to a project of Telemedicine in Intensive Care Units (ICUs).The telemedicine system provides the real time acquisition and analysis of physiological data of patients, the graphical visualisation of these data and their transmission to a central system charged with the collection and control of all the information concerning the patient, including knowledge based systems for medical reasoning. The system tasks are managed through the use of intelligent agents, implemented according to the FIPA standard. Each of the agents disposes of a knowledge-based system for its decision-making.

## 1 INTRODUCTION

The supervision of Intensive Care patients requires a complex real-time monitoring of their clinical evolution. In recent years, new monitoring and control systems were developed that make full use of the available human and technical resources. The progress in telemedicine systems has led to the appearance and introduction of intelligent monitoring systems that assist the physician with his diagnoses (Bird, 1975). Researchers have opted for the supervision of an ICU patient by a bedside computer, which gathers, integrates, stores and visualises in a clear way all the information that is forwarded by the medical devices. These systems can also be equipped with an intelligent component that performs tasks such as the automatic configuration of acquisition, alarm triggering, devices control, etc. The generalisation of communications networks has led to the integration of the medical systems in data networks under the supervision of a central system, following a client/server architecture (Bashshur, 1975).

The here described ICU telemedicine system uses communication networks (LAN, ADSL, ISDN, etc. over TCP/IP) to connect the bedside acquisition systems to a remote supervision system that centralises and shows all the information that is provided by the medical devices to the specialist, and provides medical reasoning based on independent knowledge based system (KBS) for each patient. At the acquisition level or bedside level, we have implemented the IEEE P-1073 Medical Information Bus (MIB) standard (Taboada, 1988).

All telemedicine systems, and especially those in ICUs, must dispose of multimedia applications that facilitate as much as possible the collaboration with the physician during the diagnosis (Bozios, 1995): complete videoconferencing systems (video, audio and collaborative whiteboard), the visualisation of graphic archives such as X-rays, or the transmission of files with any kind of relevant digital information (e.g. clinical records), etc.

One of the main problems in monitoring ICUs is the existence of strong temporal restrictions (Stankovic, 1993), due to the need to continuously update the data on vital parameters of the patient (heart rate, arterial pressure, central veined pressure, intracranial pressure, etc.). It is essential that we guarantee acceptable response times for the execution of tasks (mainly processing tasks). We therefore try to make good use of the available resources in the computers that are connected to the network. It becomes obvious that, especially in states of alarm, we will have to give transmission priority to the data that proceed from the acquisition devices, rather than to video and audio. Another problem, related to the use of WANs (Wide Area Network) on TCP/IP, is the limited capacity for the regulation of bandwidth; this inconvenience was avoided with compression techniques and an intelligent queue management scheme in the application level (Arcay, 2002).



Gómez A., Cantorna D., Dafonte C. and Arcay B. (2004).
FIPA-OS AGENTS APPLIED TO PROCESS SCHEDULING IN REAI TIME MONITORING.
In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 70-75
DOI: 10.5220/0001141000700075

This paper shows the included control mechanism that manages the tasks and their distribution between the connected systems. We have opted for a control mechanism based on Intelligent Agents technology, which has proven to be very adequate for this kind of tasks (Brenner, 1998) (Ferber, 1999). The development language, Java, allows us to create a multiplatform system that can be integrated into the Hospital Information System (HIS). Moreover, the bandwidth and the available resources will be better distributed if we incorporate in each agent a rules based knowledge system that decides which agent and which computer are responsible for a specific task (Gomez, 2000). This enables us to extend and improve the task management system according to the available resources of the network.

## 2 SYSTEM DESCRIPTION

One of the main problems with real-time monitoring systems in ICU is the management and execution of monitoring process tasks that require a great amount of CPU and memory: mainly signal processing tasks, image processing tasks, and database access tasks. These systems have very important temporal restrictions, and we need to guarantee the response times of their execution. An adequate tasks manager can assure the efficient use of the computers with free resources by making them execute simultaneously a larger number of specific tasks, or by indicating the most adequate computer for a concrete task. Efficient management of the available resources enhances the performance of the whole telemedicine system.

On the other hand, during the migration of the network monitoring system that was originally designed towards a telemedicine environment, we noticed that our previous reasoning, based on intelligent agents (Jennings, 1998), could also be a valid approach for the bandwidth regulation problem. The main work was focused on modifying the rules-based system, so that it could also efficiently manage the transmissions between all the involved subsystems (bedside computers and central system), acting on the state of the different transmission channels.

The main objectives of this development are the following:

The design of a communication mechanism, i.e. a language based on standards, that allows us to cover our message exchange needs.

The management of a platform of agents that are distributed in the various devices of which the network disposes. This platform allows us to know, at any given moment and through its local agent, all the agents that are distributed in the multiplatform network (Windows, Linux, UNIX). Every agent must know at each moment all the other agents that are dispersed over the network, and efficiently detect appearing or disappearing agents. These agents will collect information on the characteristics of all the computers that belong to the network, regularly run tests on the capacity of the available data lines and compile information on the data channels that are established in the system. In our implementation, a running agent resides in each computer.

The efficient management of the execution of the specific tasks. To this effect, we need to implant a priorities mechanism that is dynamically variable and error tolerant (replicating executions whenever possible). This *Tasks manager* uses the following information: free and busy resources of each computer; network congestion level; characteristics of the requested and running tasks (required computing power, amount of memory for the execution, size of the used data to decide if their transmission is feasible, evolution of the execution, etc.).

The management of the data transmission channels. This includes managing the priorities of all the transmission threads; establishing delays in determined transmission queues; regulating the bandwidth consumed by the collaborative systems; etc. These actions can be applied to outgoing or incoming data, in the latter case by means of agent messages. It is important for the agents to establish communication with both the knowledge based systems for the medical analysis (this will allow them to know specific data needs and alarm situations) and with the distinct execution threads dedicated to the data transmission (to reconfigure them). This *Communications Manager* uses the following information: the state of all the active transmission queues for each patient; the occupation level of the queues and introduced delays; the established quality for the collaborative system; alarm states; available bandwidth; etc.

On the basis of the information provided by the *Tasks manager* and the *Communications Manager*, and the parameters that define a device (CPU; memory; hard disk; etc.), we wrote the decision rules that assign tasks to the devices according to the resources that are available at each moment.

The graphical interface of each agent continuously shows the devices (platform of agents) and the tasks in which an agent is involved and the parameters that are linked to each device and each task. The interaction with this interface can be merely informative (to learn about the functioning of the agents platform), but it can also consist in modifying the state of the network by hand (task cancelling, disconnexion of an agent, etc.).

# 3 MATERIAL AND METHODS

The process control system is implanted through a system of distributed agents based on the specifications of the FIPA (FIPA, 2003), which establishes the basic architecture for the development of those agents.

The *Foundation for Intelligent Physical Agents* (FIPA) is an international organisation dedicated to promoting the industry of intelligent agents by openly developing specifications that support interoperability among agents and agent-based systems. The primary focus of this FIPA Abstract Architecture is to create semantically meaningful message exchanges between agents that may be using different messaging transports, different Agent Communication Languages (ACLs), or different content languages. Agent management provides the normative framework within which FIPA agents exist and operate. It establishes the logical reference model for the creation, registration, location, communication, migration and retirement of agents.

Each installation disposes of a component that registers all the agents in the network with the interface of the agents, *Agent Management System*. To this effect, it uses the *Localisation Service*, which makes it possible to locate the other agents during the initiation and to detect the possible decrease or increase of new agents by means of *triggers*. In case of an increase, the new agent is registered; in case of a decrease, it is eliminated from the Agent Management System. The agents were implemented with FIPA-OS components, and the communications are established with a subset of ACL language.

FIPA-OS (FIPA-OS, 2003) is a component-orientated toolkit for the construction of FIPA compliant Agents through mandatory components (i.e. components required by all the executable FIPA-OS Agents), components with switchable implementations, and optional components (i.e. components that a FIPA-OS Agent can optionally use). FIPA-OS tasks generally encapsulate some functionality that forms a logical work unit (i.e. search the DF, conduct a negotiation with another Agent, or wait for a specific lapse of time). Tasks can also be composed of sub-tasks so as to enable more complex “units of work” that are sub-divisible into smaller chunks of functionality. Tasks are based upon event-based processing, where events are delivered by the dynamic invocation of various methods.

Each agent includes a KBS based on the information that is collected either locally (from the computer that executes the agent) or by communicating with the other agents. The rules system that is integrated into the agents was made with the JESS expert system shell and scripting language (JESS, 2003). Jess can be used in two overlapping ways. Firstly, it can be a rule engine, a special type of program that very efficiently applies rules to data. In this case, they are said to constitute an expert system. Among the newest applications of expert systems are the reasoning part of intelligent agents. But the Jess language is also a general-purpose programming language, and furthermore, it can directly access all Java classes and libraries (Friedman-Hill, 2003).

Jess's rule engine uses an improved form of the Rete algorithm (Forgy, 1982) to match rules against the knowledge base. Jess is optimised for speed at the cost of space. Jess is different than some Rete-based systems in that it includes both a kind of backwards chaining and a construct called *defquery* which allows the user to make direct queries of the knowledge base.

The implementation of the rules associated to the task management was based on the characterisation of the tasks (resources and temporal requirements that are necessary for their execution) and on the information that is compiled by the agents network.

## 3.1 Tasks Control

The *Tasks Manager* includes a priorities scheme that is dynamically reconfigurable according to the needs of medical KBS, patient state, physician requests, etc. Possibilities like a task re-launch after the detection of a missing agent or normal disconnection, duplicated launch in extreme priority conditions, and others, were implemented for some concrete image processing tasks. The fact that each agent is linked to a rules system that determines all the executable actions, makes this implementation possible.

The election mechanism of the various candidates is very much influenced by the priority of the processes. From less to more priority, the processes were simplified and divided into 3 levels (0: low, 1: normal and 2: high), which, according to the tests, provide sufficient efficiency. We have followed the general rule that a superior priority process that surges at a given moment can cancel a lower priority process that is already being executed. The selection of the computer for a task of high priority consider the total capacity of which it disposes, without considering the current executions except if the alter are also marked as having high priority. This process will be re-launched after a new selection between the remaining candidates. After a first candidates selection, based on the necessary memory for the process execution (physical and virtual), we pass on to a successive refinement of the set of candidates,

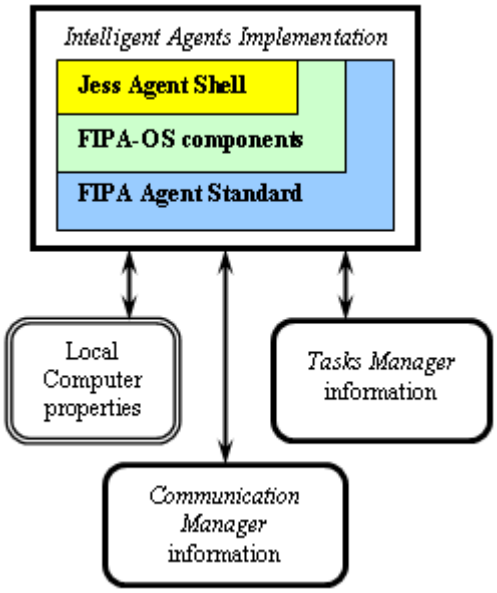


Figure 1: Tasks control scheme.

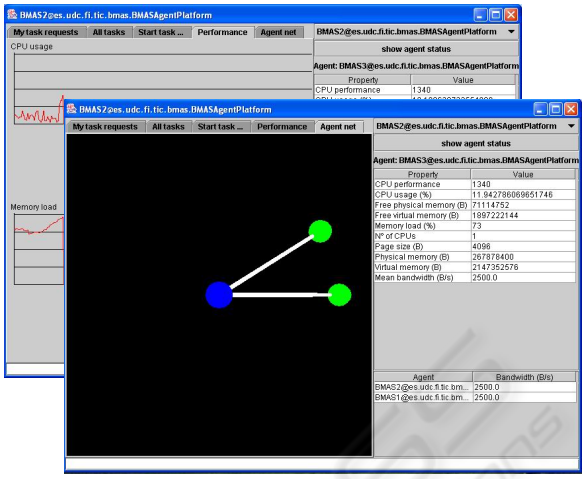

Figure 2: Agent control interface.

until there is only one left. At that moment, the system is informed about the decision, and the agent sends the pertinent message for the start of the execution.

## 3.2 Communications Control

The implementation of the *Communications Manager* is based on the real-time management of queues, multi-thread and with low TCP/IP transmissions. This avoids the implementation of a centralized events trigger (Schmidt, 1997), although we did define mechanisms for the reconfiguration of the system and the task handling.

When a new bedside PC (i.e. a new patient) connects with the central system, initially the two following connection channels are always established:

*Control and medical Analysis Information (CAI).* This transmission thread allows us to transmit and receive information on the remote control of the bedside PC from the central system, send the information generated by the intelligent module of the bedside PC (partial states obtained from the physiological subsystems; analyses of stabilities; alarms; etc.), and transfer messages between the physicians.

*Signals digitalised in Real Time (SRT).* The signals are compressed and transmitted to the Central System for each acquisition of an intermediate storage or *buffer* in the bedside communications controller (BCC).

Both execution threads for data transmission are established with high priority. The SRT and CAI channels are considered essential for the functioning of the system, since they allow a minimally acceptable monitoring, and let the central system dispose of information on physiological parameters, alarms in the acquisition system, tendency studies, etc. In fact, in the worst case, the CAI queue is always maintained because the supported traffic is minimal and informs us on the state of the remote patient.

During the course of the monitoring, any of the two nodes that maintain the communication may solicit a communication channel of the TDI type (*Transmission of Digitalised Information*, a generic channel for digitalised files), or establish a channel for the *Collaborative System* (CS) through the developed environment. These two transmission threads will initially be executed with normal priority. It is important to note that the execution of the CS will not be allowed if the measurement of the available bandwidth, made by the agents, indicates a bandwidth of less than 40Kbps.

Obviously, these different communication channels do not have the same priority, and they can change depending on the state of the patient, the state of the data network, the needs of the physician, etc. The designed mechanism functions on the application level, and is the only one that can work in such a heterogeneous environment, allowing all types of connections and with no more limitations than those imposed by the available bandwidth. We will impose these limitations through the proposed mechanism, reducing the functionality of the system

as much as necessary so as to make a certain level of medical monitoring possible.

## 3.3 Visualisation Interface

The agents control interface was implemented with JFC (Java Foundation Classes) Swing components and the API Java 3D (JAVA 3D, 2003) component for the tridimensional representation of the agents network.

The Java 3D API provides a set of object-oriented interfaces that support a simple, high-level programming model that can be used to build, render, and control the behaviour of 3D objects and visual environments (Sowrizal, 1998).

The control system that was developed for each agent possesses a user interface that is equipped with a series of functionalities, which allow us to interact and follow the behaviour of the agents network and the tasks that are being executed at each moment. This interface consists of a main panel that visualises the agents network in a 3D graphic. (Figure 2), shows the output of the local computer, allows to trigger a task and to see the information of all the tasks of the agents system, and shows information on the tasks that were requested by the telemedicine system or by the user himself.

Some of the parameters that show information on the selected agent are the following: CPU performance; CPU usage; Free physical memory; Free virtual memory; Memory load; Number of CPUs; Page size; Physical memory; Virtual memory; Mean bandwidth; etc.

A list inside the interface shows the bandwidth between the selected agent and all the other agents of the system.

The agents are represented as circular objects whose colour indicates the type: blue is the local agent, green is a remote active agent, and red is a remote inactive agent.

The bandwidth between two agents is represented by a cylindric line that connects them. The thickness of the cylinder depends on the bandwidth between the agents, estimated in real time: a thicker cylinder indicates more available bandwidth. These links also follow a colour classification that clearly indicates the available bandwidth: red is low bandwidth, blue is normal bandwidth, green is high bandwidth, and white is very high bandwidth.

Inside the 3D interface, the agents are positioned around an invisible sphere to avoid that the links pass through other agents, and to enhance the visualisation when a large collection of agents is active. The user can interact with the virtual world through certain actions, such as movements, displacements, zooms, rotations, agents selections, etc.

When an agent is selected, the system shows his data (state and characteristics).

Finally, if the user wishes to carry out tests from the interface, he can directly request the execution of a task to the agents system and determine the parameters (e.g.: priority, estimation of CPU use, estimation of memory use, etc.).

# 4 CONCLUSIONS

This article describes a telemedicine system that is designed to acquire, monitor and control the clinical evolution of ICU patients. Our system follows a client/server architecture, which means that it consists of two related but independent subsystems: the local subsystem and the central remote subsystem. Both subsystems are equipped with a series of multimedia options that improve the global system (videoconferencing communication; collaborative whiteboard; transmission of files; etc.).

Since our telemedicine monitoring system for ICUs requires a great quantity of data communications and processing tasks, with considerable temporal restrictions (real time), we have developed a tasks manager (by means of an intelligent agents technology). This module provides the following advantages:

The agent network manages and distributes the tasks by means of a KBS inside each agent. In this way, concurrence can be increased and execution time decreased.

The agents system is able to regulate the bandwidths needed for the medical network monitoring system, by managing the priorities dynamically and by adapting to the autonomously available technologies.

In the global medical monitoring system, the agents include the capacity to react in case of asynchronous events such as changes in the task priorities requested by the physician; cancellation of tasks; changes in network capacity; medical alarms; etc. We have foreseen possible concurrence problems by synchronising the accesses to shared resources, by creating copies of these resources, and by determining that certain objects cannot be modified once they are created and initialised.

The system disposes of a graphic user interface that allows us to monitor the state of the agents network, and to easily execute and eliminate tasks.

The decision mechanism (rules system) was developed in the knowledge that certain hardware elements (CPU; memory; etc.) evolve very rapidly, which inevitably requires changes in the rules. We

therefore include the possibility to modify the basic system configuration and to specify changes in the value ranges of the semantic labels without having to change the rules nor the classes.

## REFERENCES


Arcay, B., Dafonte, C., Taboada, J.A., 2002. Database Based Reasoning for Real Time Monitoring and Data Analysis, *Encyclopedia of Microcomputers*. Marcel Dekker Inc, 26(1):73-103.

Bashshur, R.L., Armstrong, P.A., Youssef, Z.I., 1975. Telemedicine: Explorations in the use of telecommunications in health care. *Telemedicine: Explorations in the use of telecommunications in health care*, Charles C. Thomas, Springfield, Illinois.

Bird, K.T., 1975. Telemedicine; concept and practice. In: R.L. Bashur, P.A. Armstrong and Z.I. Youssef (eds.), *Telemedicine; explorations in the use of telecommunication in health care*, Springfield, Illinois.

Bozios, E., Pangalos, G., Pomportsis, A., 1995. Evolution of hospital communication systems towards multimedia applications, *Medical Informatics*, 20(1):53-66.

Brenner, W., Zarnekow, R., Wittig, H., 1998. Intelligent Software Agents, *Foundations and Applications*, Springer Verlag, Berlin, Germany.

Dafonte, C., Gómez, A., Arcay, B., Taboada, J.A., 2000. Intelligent Management of Processes in a ICU Telemedicine System, *Chicago 2000 World Congress on Medical Physics and Biomedical Engineering*, Chicago, U.S.A.

*FIPA Standard*, 2003. Available in: http://www.fipa.org

*FIPA-OS toolkit*, 2003. http://fipa-os.sourceforge.net

Ferber, J., 1999. Multi-Agent Systems, *An Introduction to Distributed Artificial Intelligence*, Addison Wesley.

Forgy, C.L., 1982. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17-37.

Friedman-Hill, E.J., 2003. *Jess, the Expert System Shell for the Java Platform*, Distributed Computing Systems, Sandia National Laboratories, Livermore, California, U.S.A..

Gomez, A., Dafonte, C., Arcay, B., Santos, A., 2000. Intelligent system for the acquisition and telemonitoring in intensive care units, *Proc. 18th IASTED International Conference: Applied Informatics*, 397-401.

Gomez, A., Dafonte, C., Arcay, B., Castro, A., Pereira, J., 2002. Real-Time 3D Monitoring for Telemedicine in Critical Units, *2nd European Medical & Biological E ngineering Conference (EMBEC 2002)*.

*JAVA 3D Specification* v.1.3, 2003. Available in: http://java.sun.com/products/java-media/3D/index.jsp

*JESS v.6.1RC1, 2003*. http://herzberg.ca.sandia.gov/jess

Jennings, N.R, Wooldridge, M., 1998. Applications of Intelligent Agents, *Agent Technology Foundations, Applications, and Markets*. Springer-Verlag.

Rezazadeh, M., Evans, N.E., 1988. Remote vital signs monitor using a dial-up telephone line, *Med. & Biol. Eng. & Computing*, 26 (5):557-561.

Schmidt, D.C, Fayal, M.E., 1997. Lessons learned building reusable OO frameworks for distributed software. *Communications of the ACM*, 40(10):85-87.

Shoham, Y., 1993. Agent-oriented programming. *Artificial Intelligence*, 60(1):51-92.

Sowrizal, H., Rushforth, K., Deering, M., 1998. *The Java 3D API Specification*, Addison Wesley.

Stankovic, J.A., Ramamritham, 1993. *Hard Real-Time Systems: A Tutorial*, Computer Society Press of IEEE, 1730, Massachussets Avenue, Washington D.C.

Taboada, J.A, Arcay, B, Arias, J.E., 1988. Real time monitoring and analysis via the medical information bus. *Med. & Biol. Eng. & Comp.*, 35(1):528-534.