Chapter Number

Remote Control of a Multi-robot Infrastructure for E-Learning Training Sessions

Theodor Borangiu, Florin Daniel Anton, Silvia Anton
University Politehnica of Bucharest, Faculty of Automatic Control and Computers
Romania

1. Introduction

Robotic systems are in general composed by a multitude of software and hardware components that are susceptible to faults. For the majority of discrete, repetitive production systems including networked robots in manufacturing structures (such as lines, cells or systems) such component faults can lead to an unexpected behaviour, potential faults or even interruption of the supplied services.
Some systems are designed to be fault tolerant, meaning that in case of malfunction of a component the system will present a well known behaviour or will "hide" from the user the malfunction of such components (correcting actions are executed automatically) – i.e. it will continue to provide the specified service despite malfunctions. In many cases wrong behaviour in case of malfunction can lead to important economic loss (Lascu et al., 2005).
Understanding and designing fault tolerant distributed systems is a recognized difficulty because at the same time one must know the normal behaviour of the system but also the complex situation that occurs in case of a malfunction of a component. The difficulty of this activity is also increased due to the lack of structural coherent concepts and to the confusing terminology. That's why some basic architectural concepts, and a short fault- and paradigm classification used for structuring the fault tolerant software are presented here.
Production flows are nowadays modular, which means that each module in the enterprise is specialized and used to achieve a particular task. In many cases the modules are connected and materials are sequentially processed in each module resulting in a final, unique product or assembly. One typical such production module is a flexible cell/system using multiple robots. To obtain fault tolerance, the architecture of a distributed system must include redundant handling components.
The chapter describes a system that can be used to unify, control and observe the cell's devices (in particular each robot-vision system) from a remote location, e.g. the CAM/CAQC server linked to other design and planning compartments.

2. Faults: Classification and Semantic

A server has a correct behaviour if its answer to a request is consistent in respect to the description of the service provided by this server. We are talking about malfunction when

the behaviour of the server disrespects the service description. Considering the server behaviour the following types of malfunctions are distinguished:

a) *Byzantine malfunction*: when the malfunction is arbitrary and the server behaviour is fully random.
b) *Timing malfunction*: when the server response is correct from the behavioural point of view, but has a delay greater than the limit waiting time specified for a correct behaviour.
c) *"Omission" type malfunction*: when the server "forgets" to answer to the clients requests.
d) *Answer malfunction*: when the server answer is wrong (as value).
e) *Crash*: when the server is not answering to any request until the system is restarted. Depending on the way the server begins to function at restart, there are a number of types of crash:
   - crash with amnesia: the starting state is predefined and does not depend on the state preceding the crash;
   - crash with partial amnesia: the server keeps only a part of its state preceding the crash, the rest being reset to a previous predefined state;
   - crash with pause: the server, after an amount of time, takes the state preceding the crash.
f) *Halting – crash*: when the server never retakes the state of functioning.

2.1 Fault semantic

When the actions for retaking the operational state after detecting the malfunction are programmed, it is important to know the server's behaviour to each defect. Therefore, in a fault tolerant system it is necessary to extend the server standard specifications in such a way to include, onto the normal functioning semantic (without defect), also the fault semantic that may occur. Here the "semantic" term is used with the sense of "behaviour".
The designer of such fault tolerant systems must assure the implementing of a well defined and convenient fault semantic at the level of each system part.
Depending on the imposed restrictions for such behaviour, the fault semantic can be classified in "strong" semantic and "loose" semantic. If the specified behaviour in case of defect is more restrictive, the malfunction semantic is stronger, whereas if the behaviour is freer, the semantic is looser. The most loose semantic is the random semantic (any behaviour is permitted).
In general if the malfunction semantic is looser, then the server is more expensive and more complicated to implement.

3. The paradigm of the structure of fault tolerant software

Such paradigms have been developed to help programmers to structure the fault tolerant software. So, each of these paradigms is used for miscellaneous applications reducing the complexity of applications development.
Fault tolerance in any system implies a certain form of redundancy. There are two types of redundancies: in time and space. *Redundancy in time* means that the activity (computation) of a defected server is launched again on the same processor (after the malfunction cause has been eliminated) or on another processor and repeated after being successfully completed.

Redundancy in space implies simultaneous execution of server activities on several processors in parallel; then the final result is chosen by voting.
The most important paradigms for the proposed objectives are:

- **Transactions**: theses are software structuring mechanisms for applications, that access shared data (typically databases). The system guarantees three properties for transactions: atomicity, order and consistence. If a malfunction appears during a transaction or if the order cannot be guaranteed, then the system executes again the transaction to keep the coherence of the system states.
- **Check pointing**: a mechanism which, in case of malfunction, allows restarting the activity (computation) from a coherent state preceding the malfunction (this state is periodically memorized on a stable magnetic support). The states previously memorized are check points. The rehabilitation of the system starting from a check point is named recovery.
- **Replicated State Machine**: the provided service is executed in parallel on few processors (collection of duplicated servers). The requests of each client are sent to every copy (atomic broadcast) where are treated in a deterministic way.

Passive replication means that a service is implemented on several processors but only one is active (primary) and treats the clients requests. If the first copy is malfunctioning, its activity is retaken from the point of failure by another process (secondary backup).

4. The structure of the system

The system is composed by two applications (Fig. 1.):

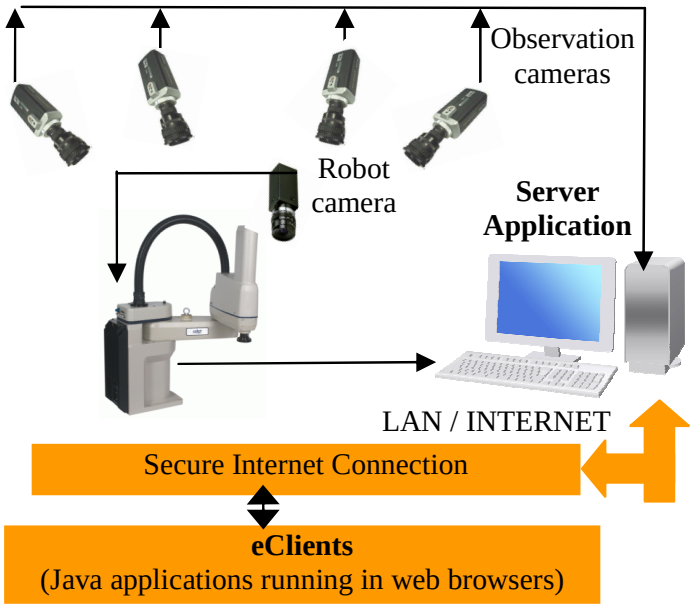

Fig. 1. The Structure of the System

1. The **Server Application (SA)**: Remote visual control and monitoring of multiple robot controllers from mobile and stationary matrix cameras (runs on Supervisor PC).
   - *Visual control*: the Server Application supports almost all V+ and AdeptVision program instructions and monitor commands. The robot training and control is interactive –

menu-driven and acknowledged by image display in a VISION window. Some of the main functions available in this window are: choice of the physical and virtual cameras and of the image buffers; selecting the display mode and resolution; histogram and average curve contrast analysis; selection of switches and parameters for virtual camera construction; display of system status; training and planning multiple ObjectFinder models for recognition and locating; learning fingerprint models for collision-free grasping; editing, saving, loading and running V+ programs. Experiments were done with Adept Technology hardware (robot controllers) and vision software (Adept, 2001).

- *Monitoring*: a Monitoring/Treatment scheme can be defined for each Client/Controller (the latter can be selected from a drop-down list of robot controllers connected to the server, by adding/removing them from the Client window) (Matsubara et al., 2002; Lascu et al., 2002). For each client a list of events and controller variables to be monitored according to a user-definable timing, and reactions taken by user-definable actions/sequences can be specified in an Automatic Treatment Window.
- *Access to image pixels*: Images taken over from clients are stored in a standard format allowing accessing the individual pixels for specialized treatment; the processed images, extracted features or computed measurements can be stored or transferred back to the client for further use.
- *Communication management*: the Server Application manages the communication with the robot controllers and the observation cameras, transfers real-time images from the cameras observing the robot workplace and production environment, reports status information, stores in a database and displays images taken by the robot camera via its controller. Finally, the SA outputs commands which are received from the eClients or acknowledges task execution.

The **eClients Applications (eCA)**: Java applications running in web browsers. They provide portal services and the connection of networked production agents: image data and RV program / report management; real-time robot control and cell / workplace observation. The eCA are composed by two applications:

- one application that retrieves the images from the observation cameras (AXIS 214 PTZ), displays them in real-time and also gives the user the possibility to change the orientation and zoom factor of the cameras.
- the second application is a VNC client.

The VNC viewer (Java client) is a web teleoperation application which can be executed into a web browser. The application connects to a Domino web server which provides a secure connection using a TCP/IP tunnel with a server having a private IP address, which cannot be accessed from Internet but only using the Domino server. The private IP machine has a VNC server that exports the display, and also the teleoperation application. Using the exported display, the user can view and use the application as when the application runs on his computer. The access is made using a username and a password, process managed by the Domino server.

For team training and application development, the system allows accessing related documents, presentations and multimedia materials, Web conferencing, instant messaging and teamwork support for efficient and security-oriented creation and use of group workspaces (students, trainers, researchers).

The Server and eClients Applications run on IBM PC workstations on which IBM Lotus software offerings are customized and integrated with other applications (error-free data and command transfer, V+ program editing, real time image display and refresh in multiple Client windows, replication functions, Client authentication, etc) in a virtual training and research laboratory across geographical boundaries.
Lotus-oriented solutions have been considered for transferring messages, status reports, data and video camera images, interpreting them and accessing databases created by all partners. The final objective of the platform is to develop an E-Learning component allowing students to access and download technical documentation, create, test, debug and run RV and AVI programs, attend real-time laboratory demonstrations, and check their skills in proposed exercises.
Thus, IBM Lotus software unifies all three Application modules, providing the necessary management and interconnecting tools for distributed industrial controllers, and the collaborative tools with back-end relational databases for team training and research.

5. Using the system

To have access to the system, a user must have a username and a valid password to enter the system. First the user must access the portal site using a Java aware browser (Internet Explorer, Opera, Firefox, with the JRE installed). After entering the correct username and password, the user is allowed in the system and has access to the teleoperation application which is a menu driven interface that allows him to interact with the system (see Fig 2).

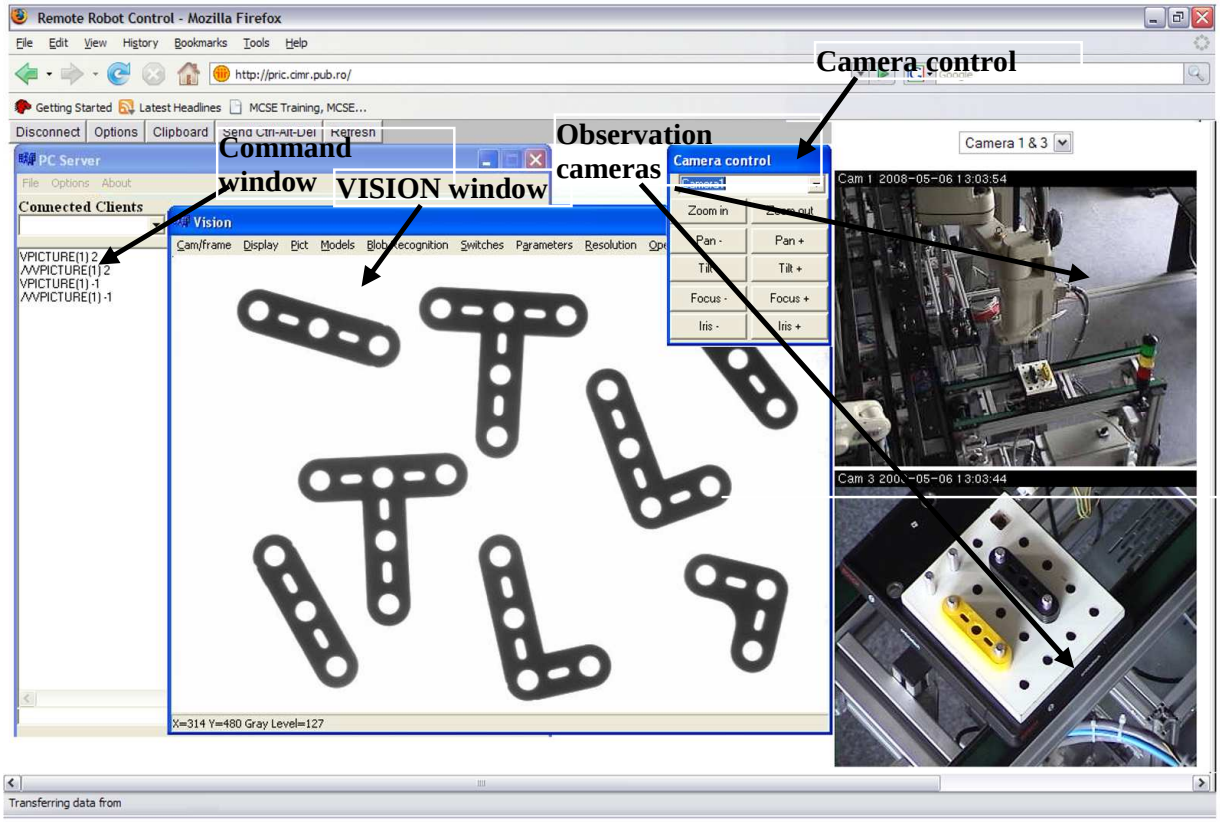

Fig. 2. Accessing the system

The teleoperation application is composed by two windows:
A command window (Fig 3), where the user can select the robot system to be controlled and issue commands from the command line or activate the vision window.
The robot stations are commanded using the command line and the menus. When a client is connected, the IP address is checked and if the client is accepted the name attached to the IP address is added to a drop down list from which the user can select what client he wishes to command. When a client who has a video camera attached the VISION button is enabled and if this button is pressed the VISION Window will open.

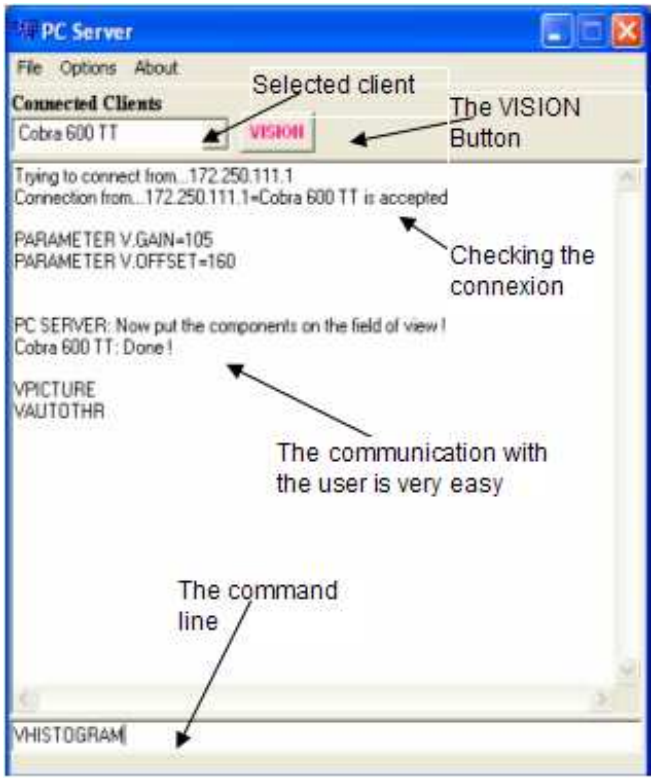

Fig. 3. The command window

From the VISION window, vision commands can be issued by selecting the desired actions from the menus (Fig 2). The most important functions are:

- selecting the physical and virtual cameras, and the virtual image buffers;
- selecting the display mode and the resolution;
- image acquisition;
- issuing primary operations (histogram, thresholding, etc.);
- displaying the vision system status;
- training models;
- configuring switches and parameters for virtual camera set-up.

The advantage of the Vision window is that all commands can be issued using menus, and the fact that the image acquired by the camera and sent to the server can now be accessed at pixel level. Another major advantage is that the training of the part recognition and grasping models become a single-step process during which a unique window is used for parameters and constraints specification.

The client application can acquire full or partial images via the VGETPIC V+ operation and send them to the server (Adept Technology, 2001).
Captured image can be processed via the menus (filtering, binarization, convolution, morphing, etc.), saved into a common format and sent to a specialized image processing application. After processing, the image can be sent back to the client and integrated to the image processing board using the VPUTPIC operation, for further use (an application using this mechanism is in course to be developed, and consists in a new part identifying algorithm based on skeleton computation and matching.
In order to execute vision programs the user must setup the vision parameters in such way that the system will "see" the objects with the best image quality. The system have specialized functions to establish those parameters automatically or, manually if some special settings are required.
After setting the parameters and loading the calibration camera-robot, the user can measure objects, apply filters, apply morphological operators, train and recognize objects.
The measurements include signature analysis, polar and linear offset signatures are computed, and stored to be used in other applications, also the skeleton computation is included in the measurements category (Borangiu, et al., 2005).
An important feature of the system is the mechanism of training object models and multiple grasping positions related to each model in order to accomplish a collision free grasping position on the runtime.
The training of object models can be done in two wais:

- first is the standard ObjectFinder model, which is computed by the vision board included in the system. The procedure to train such a model requires a set of steps which have been compacted in a single form in the application.
- the second way is to store a set of object features into a structure which characterize each model (Borangiu, 2004).

After the models are trained and stored the user can write applications using the trained models, and/or can learn also grasping positions in order to manipulate the objects.

6. The supervising function

The server application (Supervisor PC) is capable to command and supervise multiple client stations. The material flow is supervised using the client stations and the status from each station is recorded into a data base.
For the supervising function a variable and list of signals is attached to each client (Fig. 4).
The variables and signals are verified by the clients using a predefined strategy, and if a modification occurs the client sends to the server a message with the state modification. Supervising can be based on a predefined timing or permanent.
If the status of a signal or variable is changed the server analyzes the situation and takes a measure to treat the event, so each client has a list of conditions or events that are associated with a set of actions to be executed (Fig. 5). This feature removes much more from the human intervention, the appropriate measures being taken by a software supervisor.
When the *supervise* mode is selected, the server sends to the client the list of variables to be supervised and the time interval when the client must verify the status of the variables (in the case when the supervise mode is periodic).

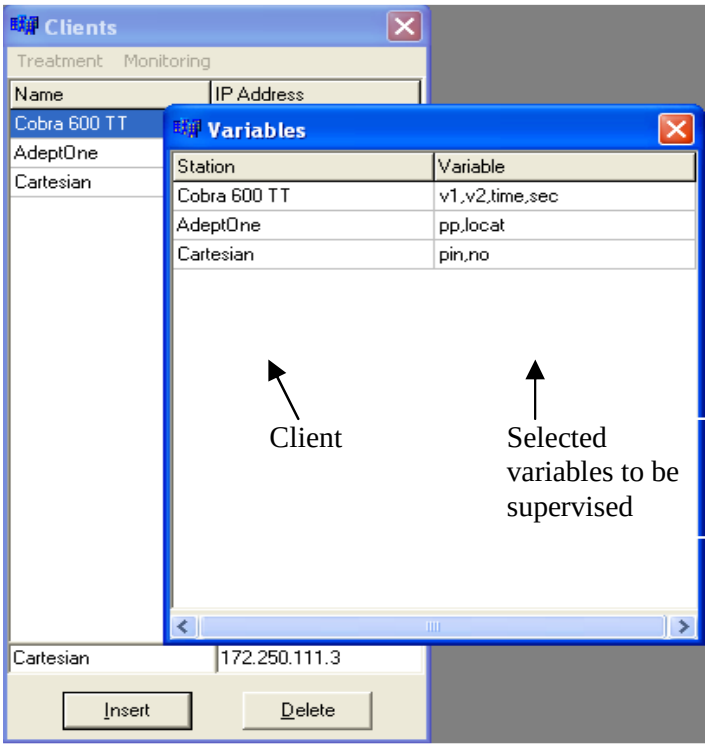

Fig. 4. Selecting the variables and the signals to be supervised

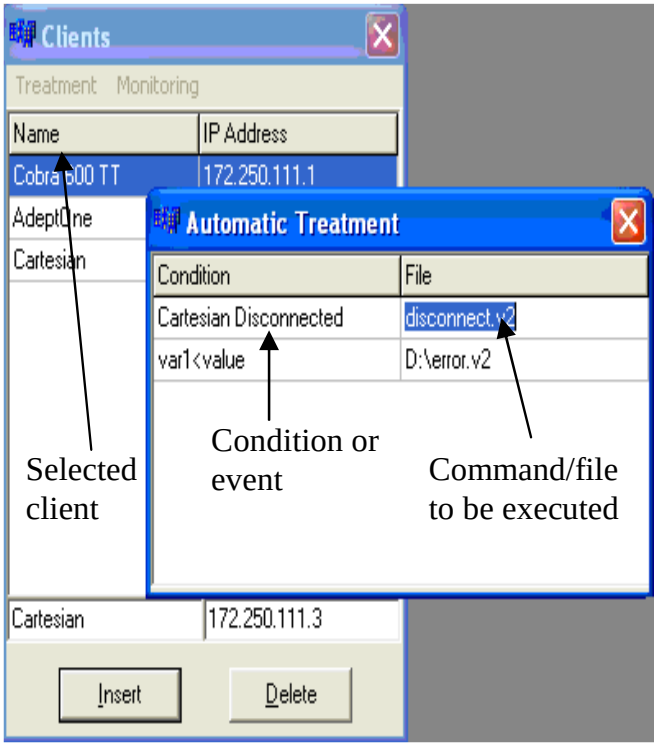

Fig. 5. Events and the corresponding actions

The events which trigger response actions can be produced by reaction programs run by the controller or by special user commands from the terminal. The list of actions contains direct commands for the robot and program execution commands (EXECUTE, CALL).

7. The solution for the design of a fault tolerant communication level

The communication level represents the key element of the management and command systems integrated with robot controllers networked in FMC structures.
A critical aspect in designing a communication level is the building, partitioning and on-line/off-line data bases transfer, which involve the multiplication of the communication links insuring a global fault tolerant behaviour. Dynamic interoperability modes must be considered, allowing the modules connected to the communication system to cooperate between them. The features announced above belong to a communication level which must combine the reliability and the performance of an industrial network with the building simplicity of a communication system used for parallel applications executed on multiprocessor machines.
To design such a communication system at the level of Controllers (C), the results from the parallel informatics with reference to the communication and process cooperation will be used. In a normal way (in the absence of malfunctions in the FMC), each controller is connected to the communication network. If in the setup stage of manufacturing or during manufacturing, a controller is malfunctioning another controller will take over the tasks. To make possible this operating mode, it is necessary to make available the data bases on each controller, and also a commutation of the informational routes with the help of the network. The availability is provided by software redundancy which involves keeping at least one replica of each data base.

7.1 CIM data base preservation

The data base preservation is realized by using a shared memory named **Tuple Space** (TS) (Borangiu & Nis, 1999), memory used by processes for communication and synchronization. The TS represent an associative set of numbers (addressable by content) of tuples. A tuple is formed by a logic name and a list of data elements which can be values or formal (logic_name, $param_1$, $param_2$, ..., $param_n$).
An example of tuple from the data base for the robot movement can be:

logic_name: data identifier {string};
$param_1$: robot identifier {integer [0..9]};
$param_2$: space coordinates identifier (Cartesian, joint, polar){char};
$param_3$: coordinate 1 : x/j_1 {reals};
$param_4$: coordinate 2 : y/j_2 {reals};
.....
$param_n$: Open/Close gripper {boolean}.

TS is a conceptual space capable to receive and memorize tuples; the basic operations defined on TS are: `out`, `in`, `read`, `eval`.
The `out` operation inserts a tuple in TS and the extraction is realized by the in operation. Searching of an appropriate tuple is made using a model (template). A model matches a tuple if they have the same logic name, the same number of parameters and the parameters

have the same type and values. If there is no tuple that match the model then the process which has executed the in operation remain blocked until the corresponding tuple will appear. The `read` operation is like the `in` operation with the exception that `read` does not erase the tuple from TS. The `eval` operation creates a dynamic tuple (a process which is executed in parallel with the process which has executed the `eval` operation). The result of this process is stored in TS as a regular tuple.

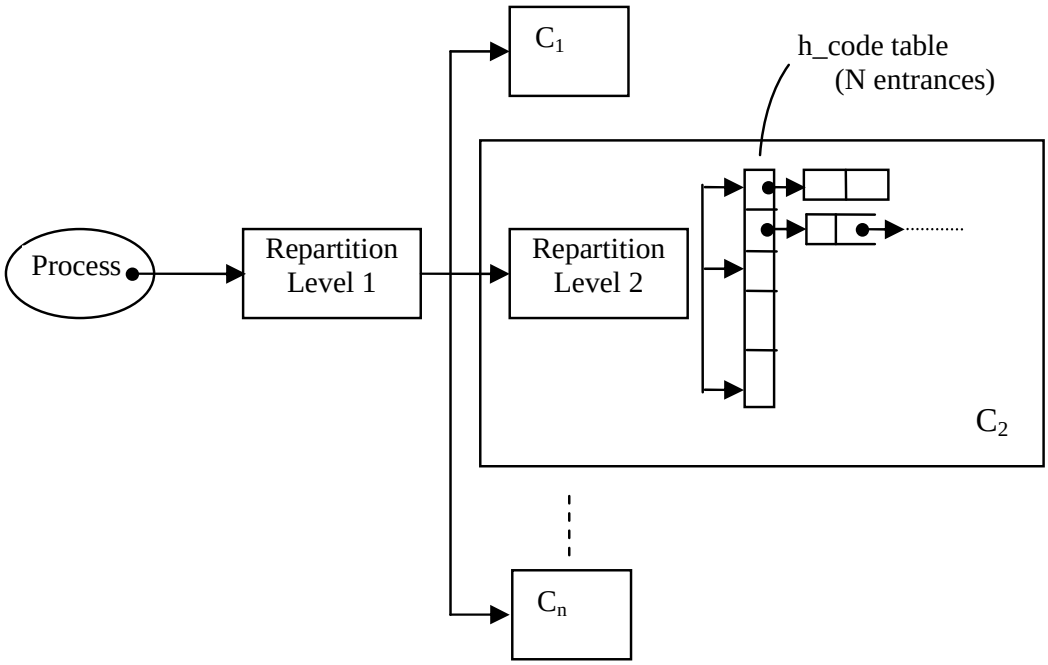

Fig. 6. Tuples distribution

7.2 Tuple distribution

In the proposed architecture the TS of the system, in particular the Computer Integrated Manufacturing data base, is distributed on the controller's network. The distribution is achieved at two levels (Fig. 6.): first establish the C and the second establish, at C level, the entry where the data will be stored.

The first level selects the C that has the physical resources referred by data and the second level selects an entry depending on the tuple logical name, each entry being the beginning of a FIFO list of tuples.

7.3 Copy management and coherency between the copies

The copies management algorithms represent an important part that achieves the fault tolerance by data redundancy. The solutions in this domain are divided in two classes: centralized management and distributed management.

- *Centralized management*: the original C makes the data duplication.
- *Distributed management*: the process which requests the access to write in TS sends the tuple to the copies locations.

For the described CAM architecture a redundancy of minimum two different locations of the same data (tuple) was chosen as is presented in Fig. 7.

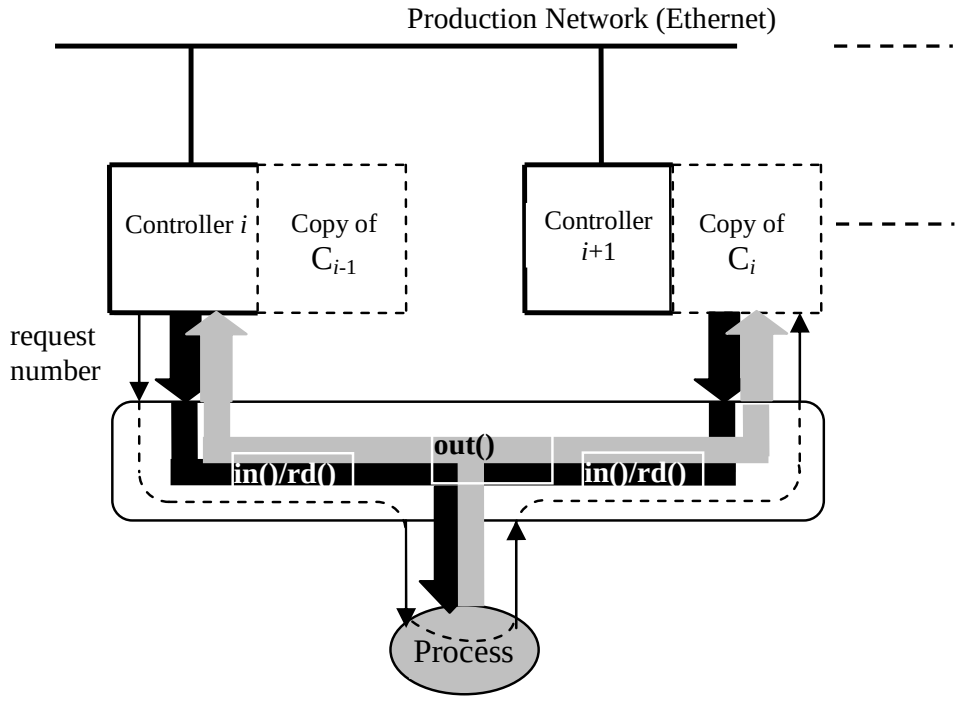

Fig. 7. Distributed copy management

It is obvious that the two classes of solutions are convenient because they allow transaction execution with only one C. In the first solution the original C must do all actions and becomes a critical resource; on the other way, the supplementary load of one of the duplicated C is avoided. The nondeterministic access to tuples and the associative selection makes uncertain the extraction of the same tuple from the two C (original and copy).
To solve this problem the request numbers (*rn*) that uniquely identify every `out` request at every TS were used. The request number is established by the original C and transferred to its replica using the calling process. In this way the tuple will have the same *rn* in each location and the coherency is assured.

7.4 The data base restoring mechanism

The logical stages that a system must accomplish are:

- malfunction detection;
- blocking each process during the reconfiguration;
- data redistribution;
- unblocking the previous blocked processes.

The data redistribution stage represents the data base "rebuilding", in fact the data base replica rebuilding. So the purpose of this stage is that for a network with *n* C, in the case of a C malfunction, the same fault tolerant structure is kept for the rest of *n*-1 left C. This stage is shown in Fig. 8, and has two sub-stages: first, the replicas are eliminated to avoid memory

saturation during the second sub-stage. Then, a redistribution process (p_thread) handles the remaining data on each C, and sends them to the new two locations.
Malfunction detection is the first function that a fault tolerant system must offer. In the presented architecture, malfunction detection is made by the processes that use the network. The malfunction is then reported to the Supervisor PC which will decide whether a new reconfiguration is necessary or not.

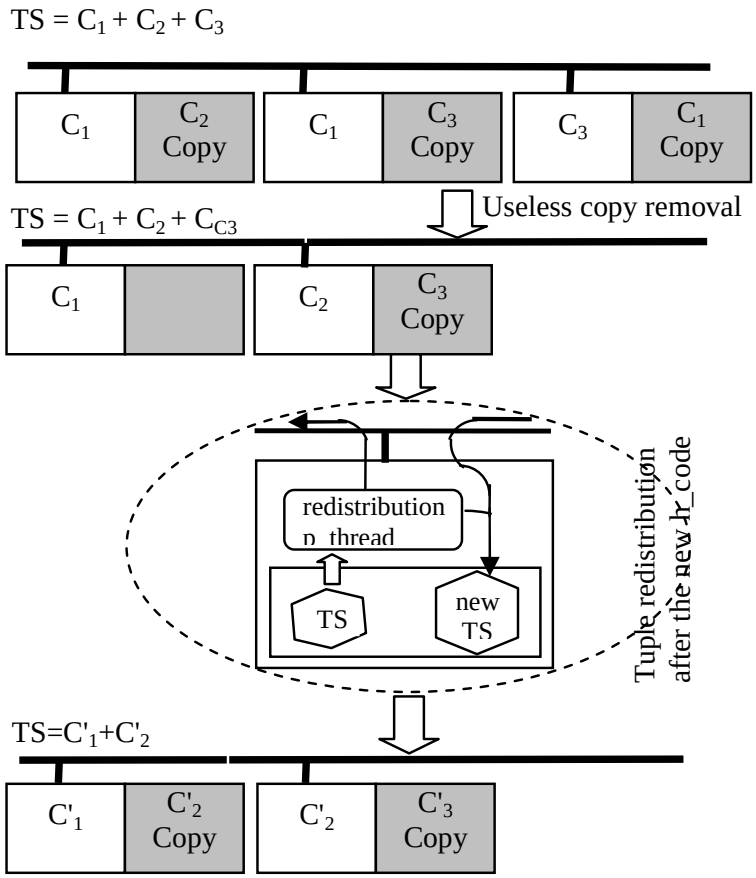

Fig. 8. Tuples redistribution

The detection strategy selected uses a *confirmation with time-out* method, meaning that at each message exchange during the process access to TS, the process waits a confirmation (acknowledge). A process is capable to detect a malfunction only when it tries to access one of the servers that implement the TS. These moments occur when:

- the process send a request to the server;
- the data transfer between server and process a result of request processing;

In the following paragraph the communication protocol for the `in` operation at the process level and at the TS server level will be presented in detail.
The description of the protocol at server level for the operation `in` resides in considering that the end of the dialog can eventually coincide with the end of the reconfiguration (see Fig. 9). In this case the process must restart the dialog by reconnecting to the Supervisor PC in order to know the new system configuration.

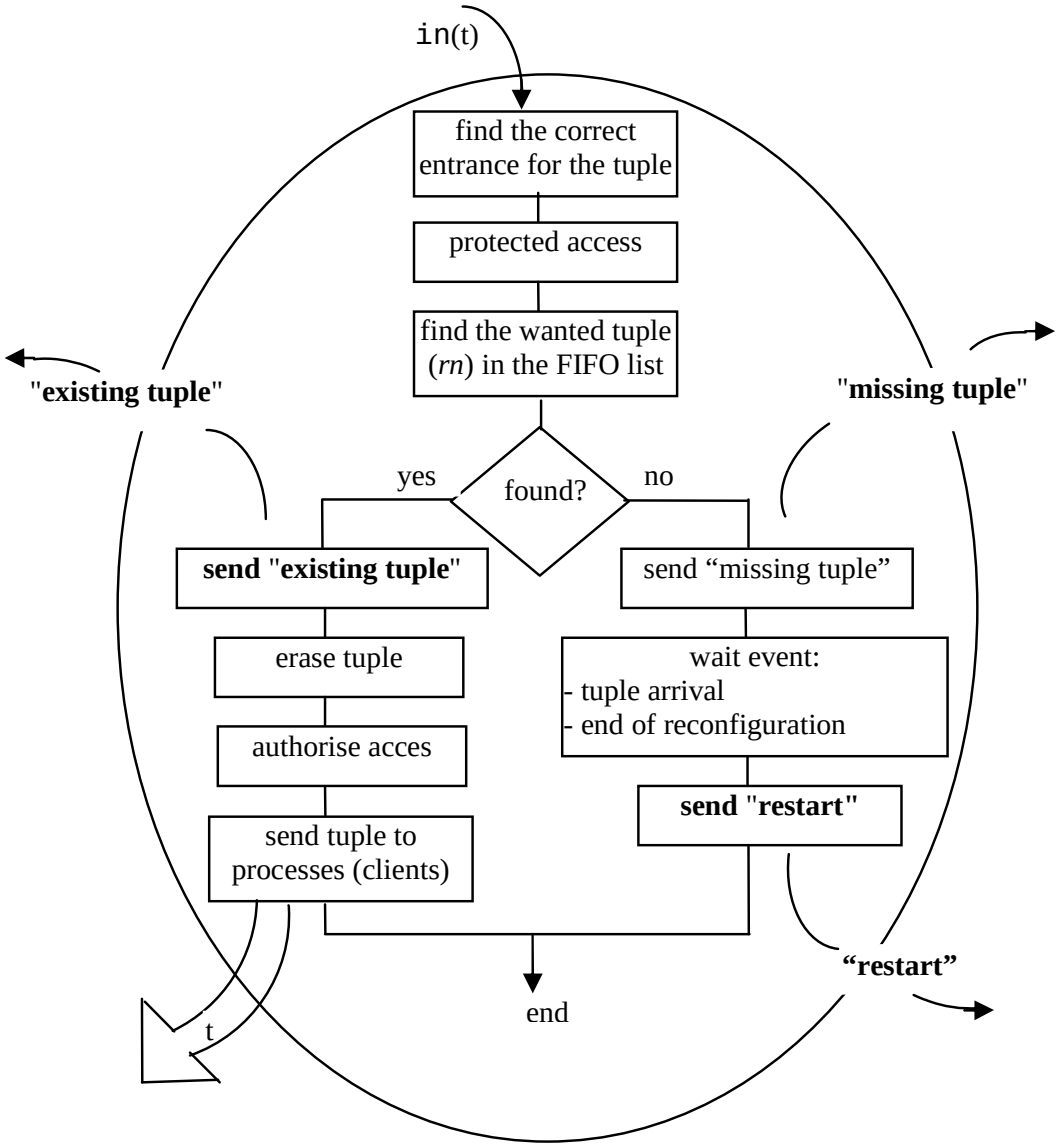

Fig. 9. The `in()` operation at the server level

It must be mentioned that during the configuration, the process will be blocked. This is achieved by each server that does not respond at any request issued by processes until the end of the reconfiguration.

To unblock the processes, their second connexion with the TS server will be used.

Another problem to be solved at the server level is to make the difference between the tuples arriving in the TS as result of redistribution during reconfiguration and the tuples sent by the processes. This aspect is important to avoid unblocking a process which awaits a tuple as the result of another tuple's arrival due to redistribution. The simplest solution is to reserve a communication channel (private port) between every server dedicated only for tuple transfer during reconfiguration.

On the other hand, the same malfunction can de detected simultaneously by more than one process but that malfunction must be considered only once. That’s why every system state

between any two reconfigurations has a version number (vn**)** and thereby any report with a vn less than the current vn is ignored.
At process level, when a tuple is required, the process awaits with time-out the confirmation messages from the two tuple locations. The diagram in Fig. 10 shows that the process detects a malfunction upon receiving the presence ("missing", "existing") or restart messages only from one location.

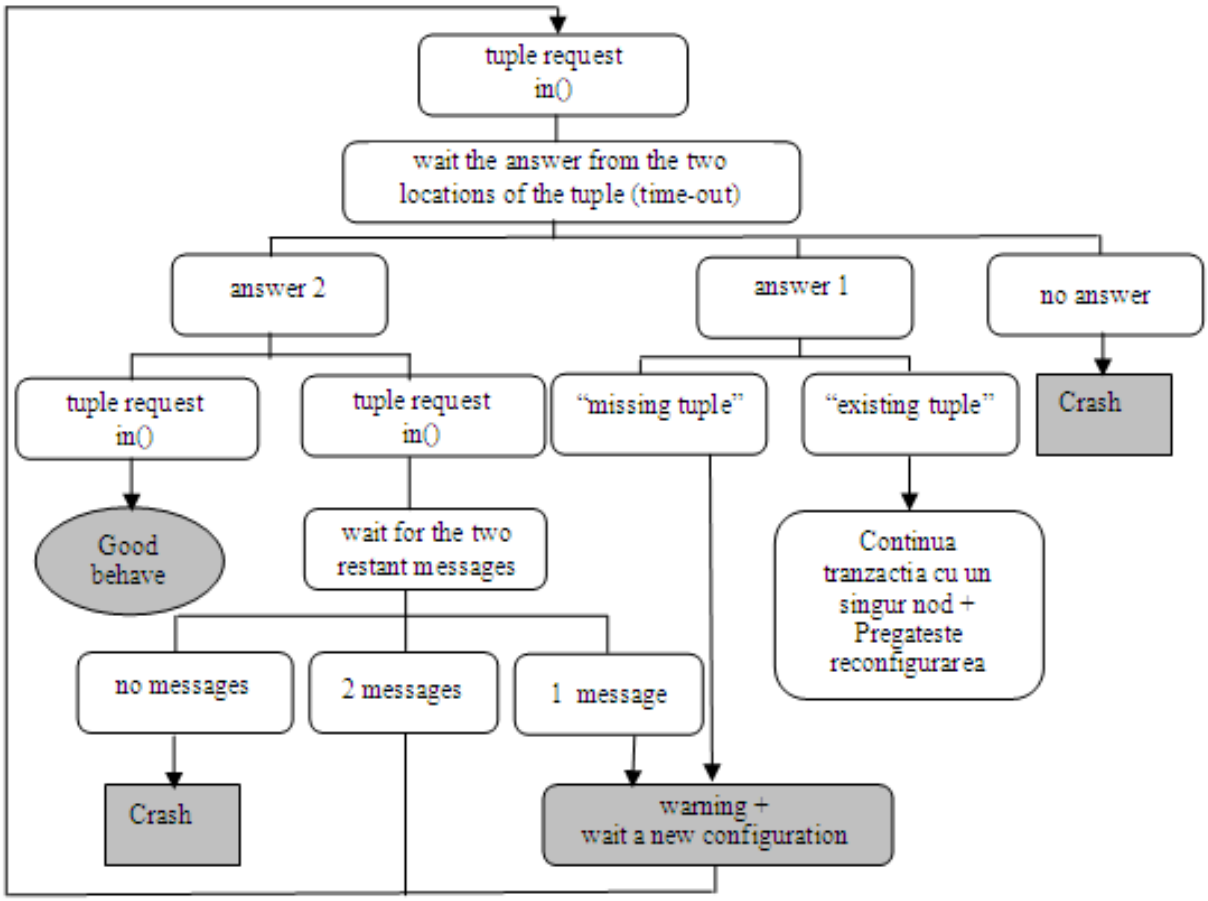

Fig. 10. The `in()` operation at process level

Two different behaviours are distinguished:

- if the tuple does not exist in any location, then the process reports to the supervisor and awaits the new configuration;
- if the tuple is present at least in one location then the process continues the execution after its reports to the supervisor to prepare the reconfiguration.

When there is no presence or restart message, the state of the system is considered as crash.

7.5 Reconfiguring the network

Rebuilding the communication route in the network represents the last step required to restart the normal behaviour of the FMC control system.
In case of malfunction of the communication network the following most important cases can appear:

1. If the connection between the Switch and the Supervisor PC is down the remote control will be lost, but the FMC will reconfigure as follows: the controller will use the Ethernet network for communication, and the controller with the first IP from the class will take the functions of the Supervisor PC. If the connexion is re-established the Supervisor PC makes a query, finds the replacing controller and transfers the databases and restarts the normal behaviour.
2. If the switch is not functioning, all the Ethernet connexions are lost but the controllers will use the serial "network". The behaviour is like in the first case only that the web users can view the status from the Supervisor PC, including the images acquired by the observation cameras.
3. If a controller loses the Ethernet connexion, it will use one of the two serial lines to reach the Supervisor PC depending on the CPU time of the neighbours.

8. Lotus Domino platform and E-Learning aspects

The strong impact of this project consists in stimulating the cooperation between different networked areas of an enterprise. The objective was to build a system that provides access to public information for a wide audience and at the same time supports collaboration between registered members, provides safe access to protected contents and enables the publication and editing of contents. The system can be accessed from a great variety of places. Therefore care was taken to also ensure optimal use in case of lower bandwidth and poorer quality hardware. The high level of security and availability are key components, which were ensured by the selection of high quality technical devices and well-planned loading. As the portal must be prepared for a growing number of users, the tool must be highly scalable. It needs to integrate contents and services and provide access for document repositories. User groups must be able to access personalised contents.
High-level availability must be guaranteed. The system should be scalable according to load and requirements. The use of NLBS (Network Load Balancing System) provides a solution for an even (balanced) load of the network. The portal user interface needs to be customized, and templates must be created and uploaded. The authorisations and user groups must be defined.
The eClient application Lotus Domino Server implements a security access policy to the virtual workspace. The access to the eClient application is granted based on the Domino defined ACL's (Access Control Lists), such that in order to connect to the application the user must specify a user name and a password. There were defined two classes of privileges:

1. A **user class** where the operator can observe images acquired from the observation web cameras and images from the VISION system taken by multiple area cameras; he can also view the commands issued by the trainer and watch the results of the commands;
2. A **trainer class** where the operator is authorized to issue commands for every connected robot system, upload, download and modify programs. The trainer can also take pictures from an individual camera and use the specific vision tools to process that image. The observation cameras can also be moved and positioned in the desired location by the trainer. The trainer can give full or partial permissions to users for program testing purposes. The communication between the users is achieved by help of the integrated console (text mode) or using an Instant Messaging and Web Conferencing application (Sametime).

IBM Lotus Domino server software was used to combine enterprise-class messaging and calendar / scheduling capabilities with a robust platform for collaborative applications on a wide variety of operating systems. The design of the Lotus Domino Server made available three offerings: Domino Messaging Server (messaging only), Domino Utility Server (applications only), and Domino Enterprise Server (both messaging and applications) (Brooks, *et al.*, 2004).
The most important Lotus Domino features used in the project are:

- Encryption, signing, and authentication using the RSA public-key technology, which allows to mark a document in such way that the recipient of the document can decisively determine that the document was unmodified during transmission.
- Access Control Lists (ACLs) determining who can access each database (application) and to what extent.
- Usage of Domino's new features to reduce network utilization. Network compression reduced the number of bytes sent during transactions by up to 50 percent. Connections across heavily loaded links such as WANs and XPCs see the most benefit.
- Availability for the Windows NT and XP platforms, automatic fault recovery after shutdown and server restart without administrator intervention after the occurrence of an exception. Fault recovery uses operating system resources, like message queues.

Because, the application eClient is accessed over the Internet, security represents a critical element. The access to different levels of the application is controlled by xACLs (extended ACLs) to allow or disallow access. The existing database Access Control Lists (ACLs) and the new ACL file feature ensure that application-private databases remain secure. In addition, file protection documents for the Domino Web server which is used to serve the eClient (Java application) provide additional access control for files accessed via HTTP.

9. Portal performances

In this section, portal performances aspects are presented. The tests have been conducted using two networks: an internal 10/100 Ethernet network which has the Internet bandwidth of 61.38 Mbps (resulted from tests), and an external wireless network using a 3G+ HSDPA modem, and having a bandwidth of 3.6 Mbps.
For internal testing, two computers have been used:

A) A PC workstation having the HW configuration: CPU 3.4GHz, 4GB RAM, 75GB HDD SCSI, installed with Windows XP SP2, Internet Explorer as browser, with ActiveX controls: AXIS for viewing the Motion JPEG images, QuickTime to view MPEG-4 streams, and jre 1.4.2 to control the teleoperation application.

B) A PC workstation having the HW configuration: CPU 3.4GHz, 4GB RAM, 75GB HDD SCSI, installed with Linux: KNOPPIX, the internet browser Firefox with plugging: AXIS for viewing the Motion JPEG images, QuickTime to view MPEG-4 streams, and jre 1.4.2 to control the teleoperation application.

For the external testing a DELL Inspiron 1720 Laptop has been used, CPU 2GHz, Dual Core, 2GB RAM, 240 GB HDD, Modem HSDPA Vodafone installed with Windows XP SP2, the Internet browser Firefox with plugging: AXIS for viewing the Motion JPEG images, QuickTime to view MPEG-4 streams, and jre 1.4.2 to control the teleoperation application.

9.1 Teleoperation and image transmission tests

The first test has been conducted to evaluate the impact of the two applications (teleoperation and image transmission) on the bandwidth. For the teleoperation application a bandwidth test has been conducted for two cases: a case where there is no teleoperation activity, and a case where a high activity has been simulated. The results are presented in Table 1.

PC workstation	High activity	Idle
A	86.22 kbps	0 kbps
B	81.5 kbps	0 kbps

Table 1. Test results for the teleoperation application

The image transmission application test consists in two separate tests to detect the differences between the transmissions for colour/grey level images. The compression level used was Motion JPEG and MPEG-4, and the measurements have been done for the used bandwidth and the rate of frames per second (fps) for the following resolutions: 4CIF, 2CIF Expanded, 2CIF, and CIF. The results are presented in table 2 and 3.

Colour/Resolution			4 CIF (704x576)	2 CIF E (704x576)	2 CIF (704x288)	CIF (352x288)
A	B&W	fps	25.02	25.01	25.02	25.02
		kbps	6151	6215	4467	2443
	Colour	fps	25.01	25	25.02	25.02
		kbps	6428	6221	4353	2461
B	B&W	fps	25.02	25.02	25.02	25.02
		kbps	6203	6198	4420	2420
	Colour	fps	25	25	25.02	25.02
		kbps	6421	6217	4375	2443

Table 2. Image transmission test for Motion JPEG compression

Colour/Resolution			4 CIF (704x576)	2 CIF E (704x576)	2 CIF (704x288)	CIF (352x288)
A	B&W	fps	13.83	15.09	17.31	25
		kbps	1342	1342	1450	1285
	Colour	fps	13.83	15.09	17.07	24.03
		kbps	1421	1412	1441	1298
B	B&W	fps	13.83	15.09	17.25	25
		kbps	1335	1351	1445	1288
	Colour	fps	13.83	15.09	17.20	25
		kbps	1330	1375	1451	1293

Table3. Image transmission test for MPEG-4 compression

9.2 Teleoperation and image transmission combined tests in the internal network

These tests have been carried out to check the application performances such as bandwidth and rate of frames per second in two types of tests: (i) a test in which the bit rate is constant without specifying a maximum rate of bits, using the priorities: *image quality* and *fps*; (ii) a

second test using a variable rate of bits, but in which a maximum bit rate has been configured and where the video stream has been optimized for the bandwidth utilization or for the rate of fps. The compression rate was MPEG-4 and the resolution CIF. If the compression is Motion JPEG the bandwidth is 9910 kbps and the fps rate is 25.02. The tests have been conducted for the following bit rates: 48 kbps – GPRS Class 12, 52 kbps – 2G, 384 kbps – EDGE, 512 kbps, 1 Mbps, 3.6 Mbps – 3G, 7.2 Mbps – 3G+, 10Mbps – LAN. The results for the internal network are presented in Tables 4 and 5, and the results for the external network are presented in Tables 6 and 7.

Priority\Bit rate			48k	52k	128k	384k	512k	1M	3.6M	7.2M	10M
A	Image quality	fps	2	3	5	14	18	35	39	39	40
		kbps	116	140	275	761	1035	1974	2424	2436	2442
	Fps rate	fps	10	20	22	41	50	50	39	39	39
		kbps	89	105	242	775	1032	1980	2400	2415	2420

Table 4. Application testing using the internal network and a constant bit rate

Optimization\Bit rate			48k	52k	128k	384k	512k	1M	3.6M	7.2M	10M
A	Bandwidth	fps	1.5	2	5	15	20	35	50	50	50
		kbps	108	113	260	772	1016	1824	1960	1970	1970
	Fps rate	fps	2	2	6	16	22	35	50	50	49
		kbps	120	120	267	773	1180	1746	1893	1859	1836

Table 5. Application testing using the internal network and a variable bit rate

Priority\Bit rate			48k	52k	128k	384k	512k	1M	3.6M	7.2M	10M
A	Image quality	fps		2	5	14	19	35	39	39	39
		kbps	177	156	278	557	718	1412	1475	1510	1722
	Fps rate	fps	8	6	13	41	50	50	39	39	39
		kbps	104	112	254	755	1026	1815	1526	1594	1640

Table 6. Application testing using the external network and a constant bit rate

Optimization\Bit rate			48k	52k	128k	384k	512k	1M	3.6M	7.2M	10M
A	Bandwidth	fps	2	2	4	15	19	34	50	50	50
		kbps	93	116	280	596	980	1355	1944	2043	2024
	Fps rate	fps	2	2	5	15	19	34	49	50	50
		kbps	93	114	268	743	722	1152	1529	2006	2002

Table 7. Application testing using the external network and a variable bit rate

The first test has been executed in order to evaluate the differences between the PC workstations A and B concerning the bandwidth, where it has been observed that the Linux machine uses a bandwidth smaller with 4.72 kbps which is not concluding enough to make a difference between the two workstations.

The second test (Table 2) has been executed in order to estimate the difference between colour and grey level video streams for workstations A and B. As can be seen from Figs. 11, 12, 13 and 14 the performance differences between the two workstations are insignificant, so we can conclude that the teleoperation system has the same performance regardless the operating system of the user.

From the point of view of image encoding (in accordance with the type of the image – colour, grey level) it can be seen that the differences between colour and grey level images are too small to be considered, but, if the two types of encodings are compared one can see that Motion JPEG uses a bandwidth of 2500 kbps for the CIF resolution, while the MPEG 4 brings an improvement of 200% (the bandwidth used in this case being less of 1300 kbps). Due to this, MPEG-4 is used to minimize the bandwidth, while Motion JPEG is used for a better image quality.
From the point of view of image resolution, the CIF resolution was selected because the system uses a relative small bandwidth and also the size of the image is acceptable, considering that on the user workstation two images must be displayed and the teleoperation application has a resolution of 800x640 pixels.

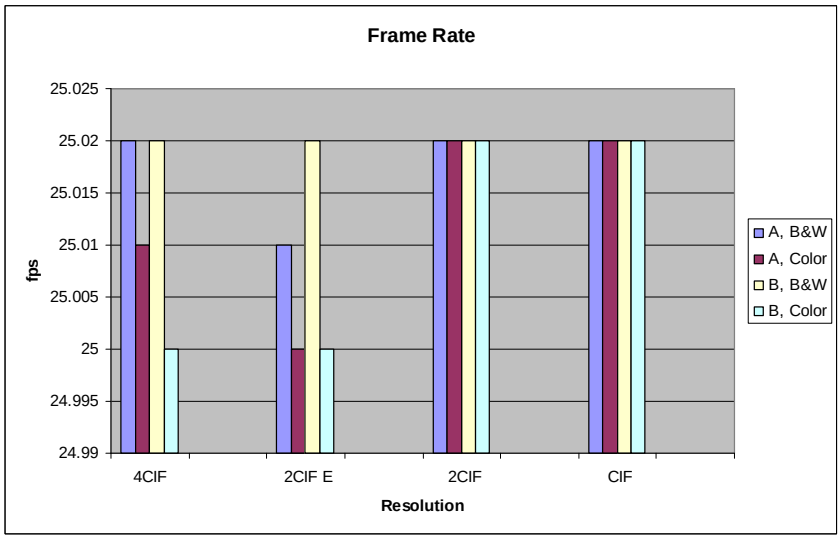

Fig. 11. FPS rate for Motion JPEG encoding

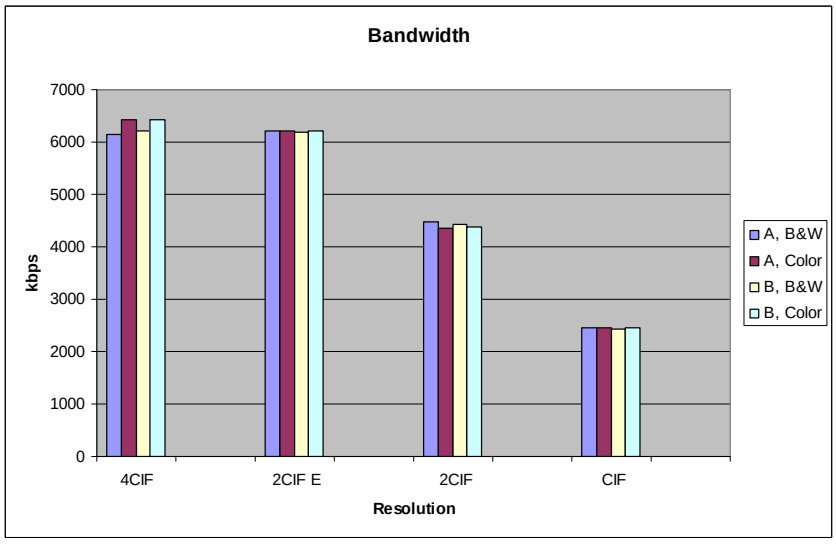

Fig. 12. Bandwidth for Motion JPEG encoding

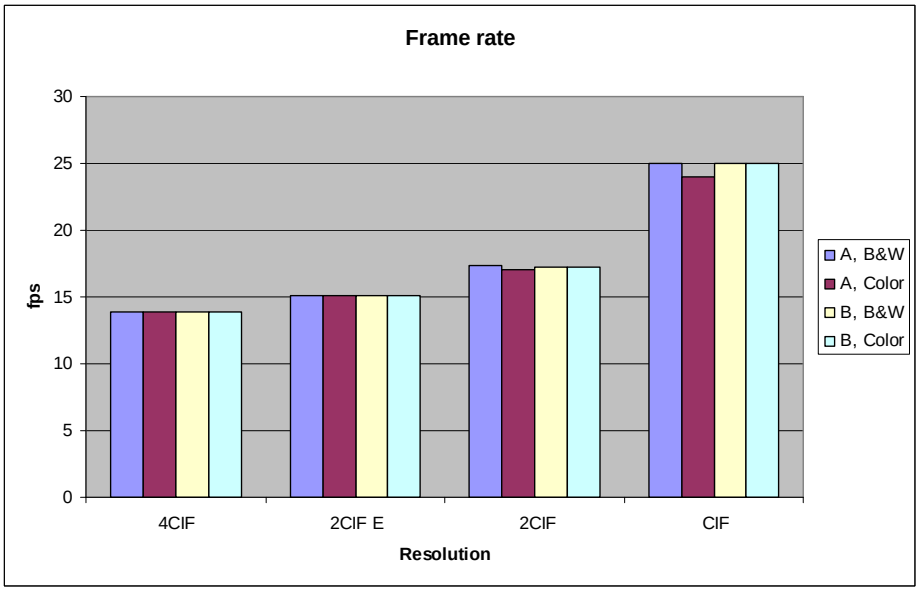

Fig. 13. FPS rate for MPEG-4 encoding

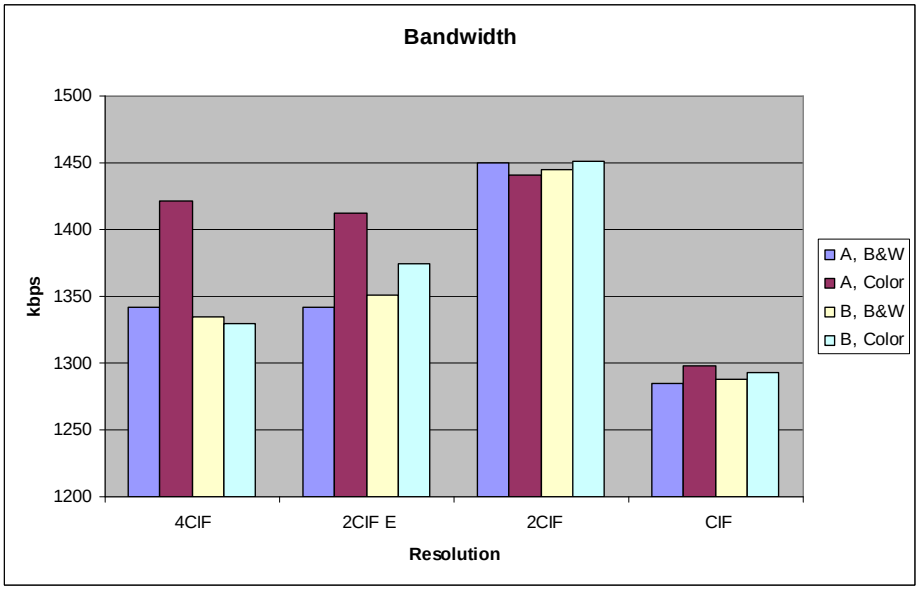

Fig. 14. Bandwidth for MPEG-4 encoding

For the internal network (Figs. 15, 16, 17, and 18) it can be seen that the solution which must be chosen both for image quality and bandwidth is the solution with constant bit rate, which works satisfactory from a bit rate of 384kbps, the total rate being approximately 750 kbps. From the point of view of quality the priority is the frame rate (fps), the differences being over 200% for bit rates under 1Mbps against the priority *image quality*.
For the external testing (Figs. 19, 20, 21, 22) it can be seen that the same rules are met, and the conclusions are the same as for the internal network.

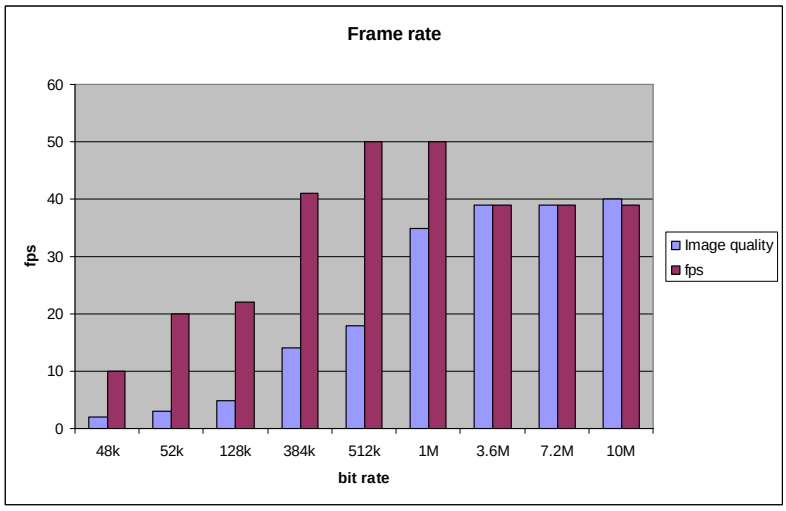

Fig. 15. FPS for internal network, constant bit rate, priority: *Image quality, fps*

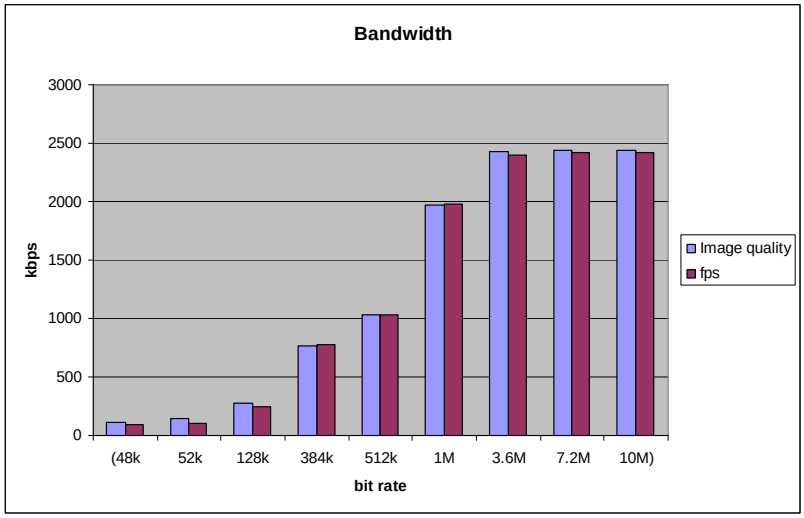

Fig. 16. Bandwidth for internal network, constant bit rate, priority: *Image quality, fps*

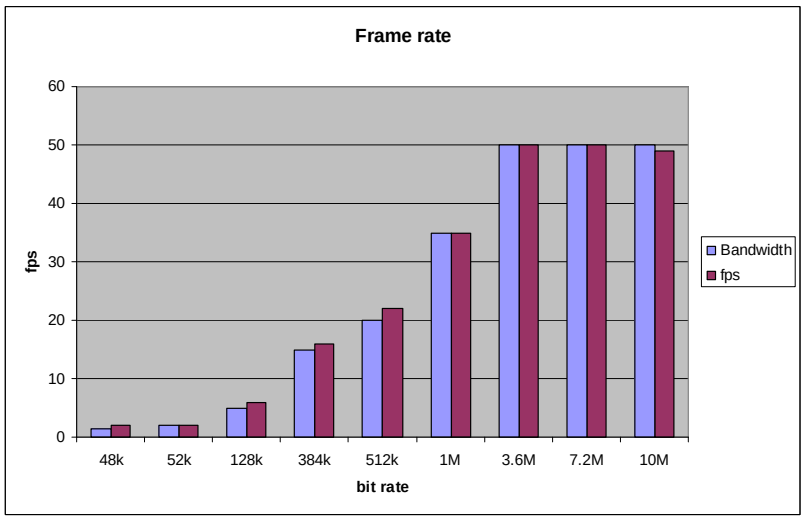

Fig. 17. FPS for internal network, variable bit rate, stream optimization: *Bandwidth, fps*

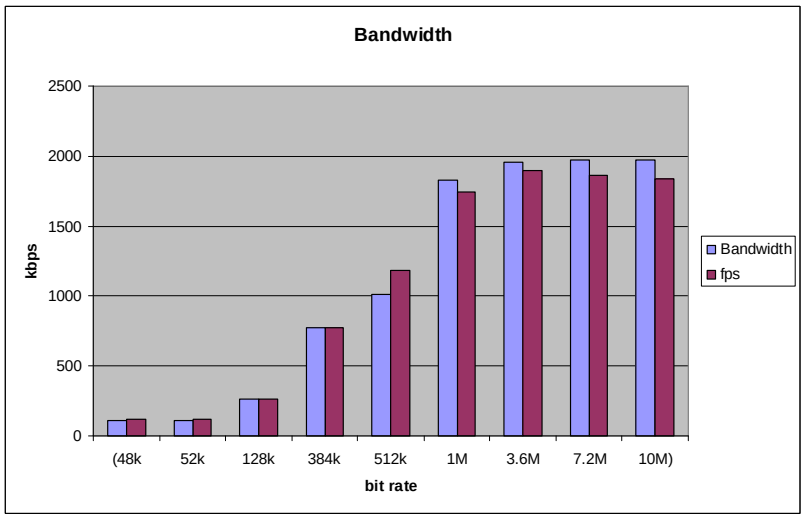

Fig. 18. Bandwidth - internal network, variable bit rate, stream optimization: *Bandwidth, fps*

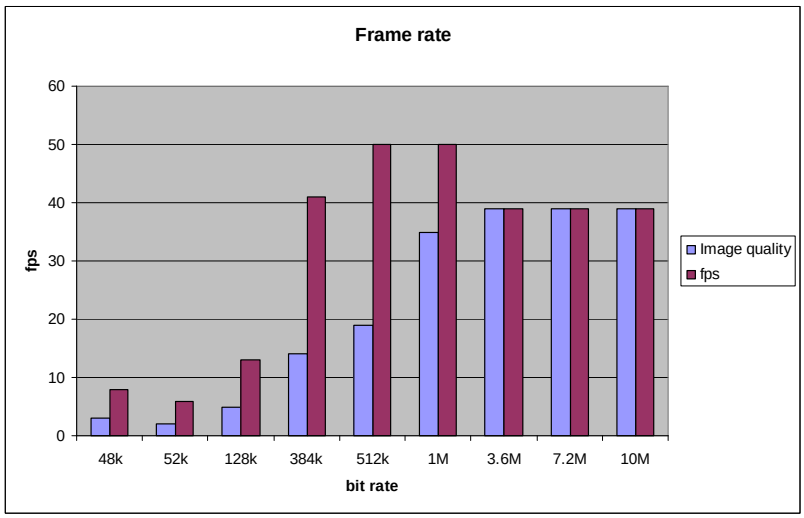

Fig. 19. FPS for external network, constant bit rate, priority: *Image quality, fps*

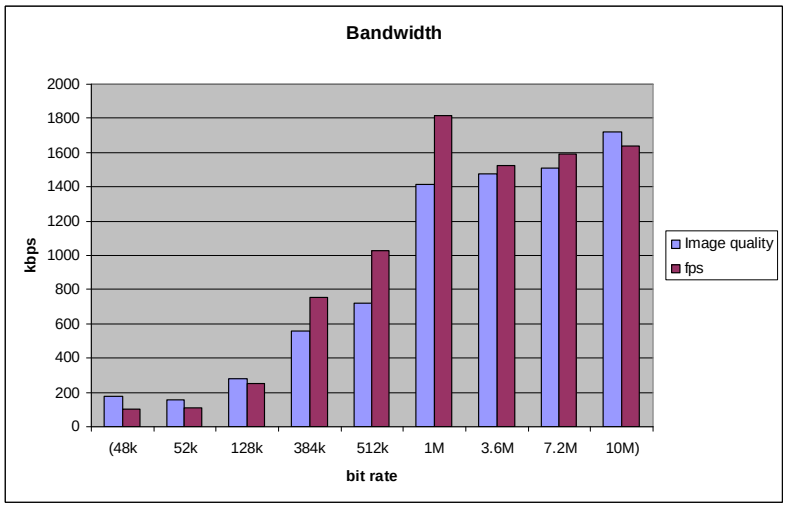

Fig. 20. Bandwidth for external network, constant bit rate, priority: *Image quality, fps*

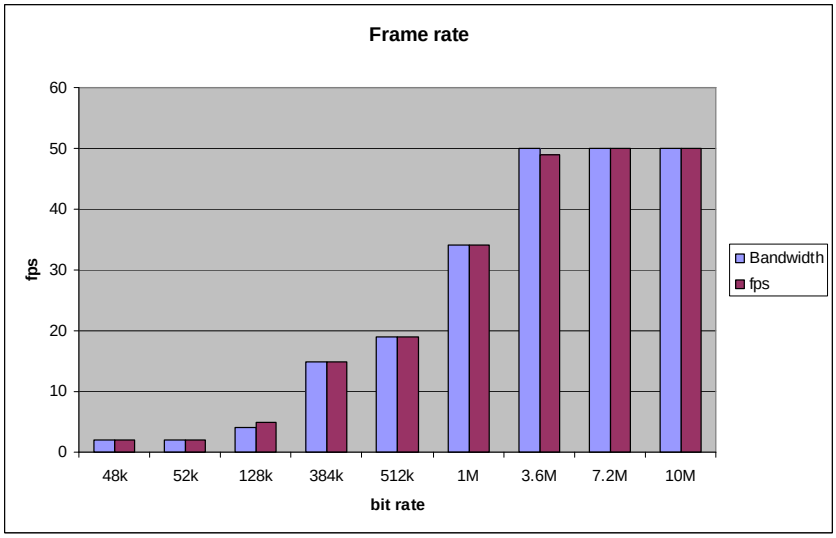

Fig. 21. FPS for external network, variable bit rate, stream optimization: *Bandwidth, fps*

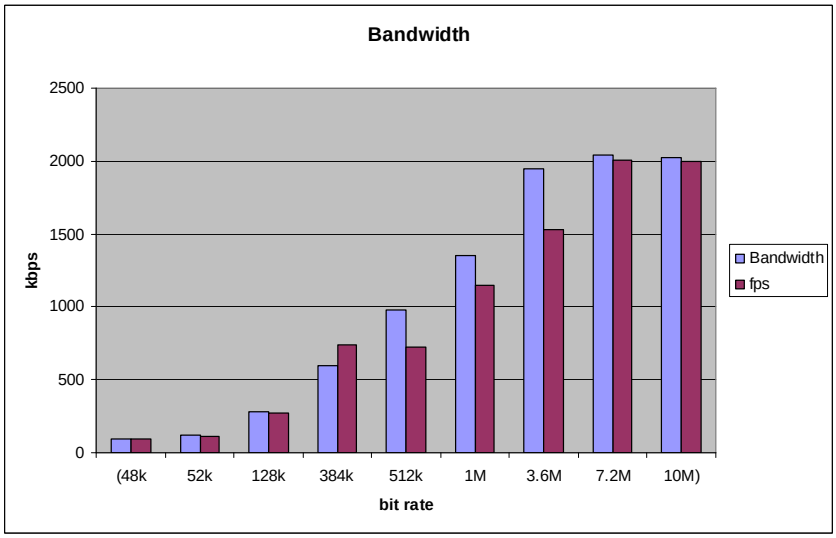

Fig. 22. Bandwidth - external network, variable bit rate, stream optimization: *Bandwidth, fps*

In conclusion, the tests have shown that in order to obtain a high image quality and a maximum fps, in other words a maximum performance for the teleoperation application, the user can use an internal or external network with a bandwidth of at least 1Mbps, or 128KB/s using MPEG-4 image encoding, a constant bit rate of 512kbps and using as priority *fps*.

7. Implementing results. Conclusions

Fault-tolerance is provided to the cell communication system (Fig. 23), providing redundancy at both Station Controller level (a break down of a Robot Controller is detectable, the production tasks can be rescheduled to the remaining valid units for graceful degraded behaviour) and at Station Computer level (replication of data bases for the IBM PC-type device terminals, reassignment of computers in case of break downs).

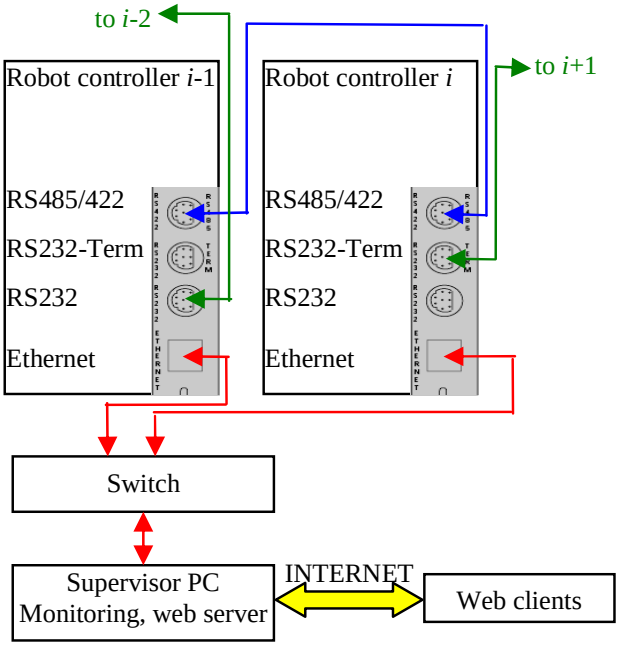

Fig. 23. Fault-tolerant communication architecture

The fault tolerance solution above presented is worth to be considered in environments where the production structure is reconfigurable, and where the manufacturing control must assure a continuous production flow at batch level (job shop flow).
There are also some drawbacks in this solution. The spatial layout and configuring of robots must be done such that one robot will be able to take the functions of another robot in case of failure. If this involves common workspaces, programming must be made with much care using robot synchronizations and monitoring continuously the current position of the manipulator.
The main advantage of the proposed solution is that the structure provides a continuous production flow with an insignificant downtime (during reconfiguration).
The solution was tested on a four-robot assembly cell located in the Robotics and IA Laboratory of the University Politehnica of Bucharest. The cell also includes two CNC milling machines and one Automatic Storage and Retrieval System, for raw material feeding and finite products storage.
During the tests the robot network has detected a number of errors (end-effector collision with parts, communication errors, power failure, etc.) The robotic network has evaluated the particular situation, and the network was reconfigured and the abandoned applications were restarted in a time between 0.2 and 3 seconds.
The network failure was simulated in the tests, as a complex example.
One robot (R2) was disconnected from the Ethernet network, the heartbeat packet sent by the robot to the other cluster members has detected the malfunction and the robot has switched the communication using the serial line; this was done in 0.3 seconds after the Ethernet cable was removed. The communication between the affected robot and his neighbours was done using the serial lines, and the communication with other robots was done by routing the communication using the Ethernet line of the neighbours (R1 and R3). In this way the communication latency was reduced.

After the communication was re-established, the serial lines of the robot have been disconnected. The robot has detected the communication failure, has stopped the manipulation program and retracted the manipulator in a default position in the exterior of the working area in 0.8s.

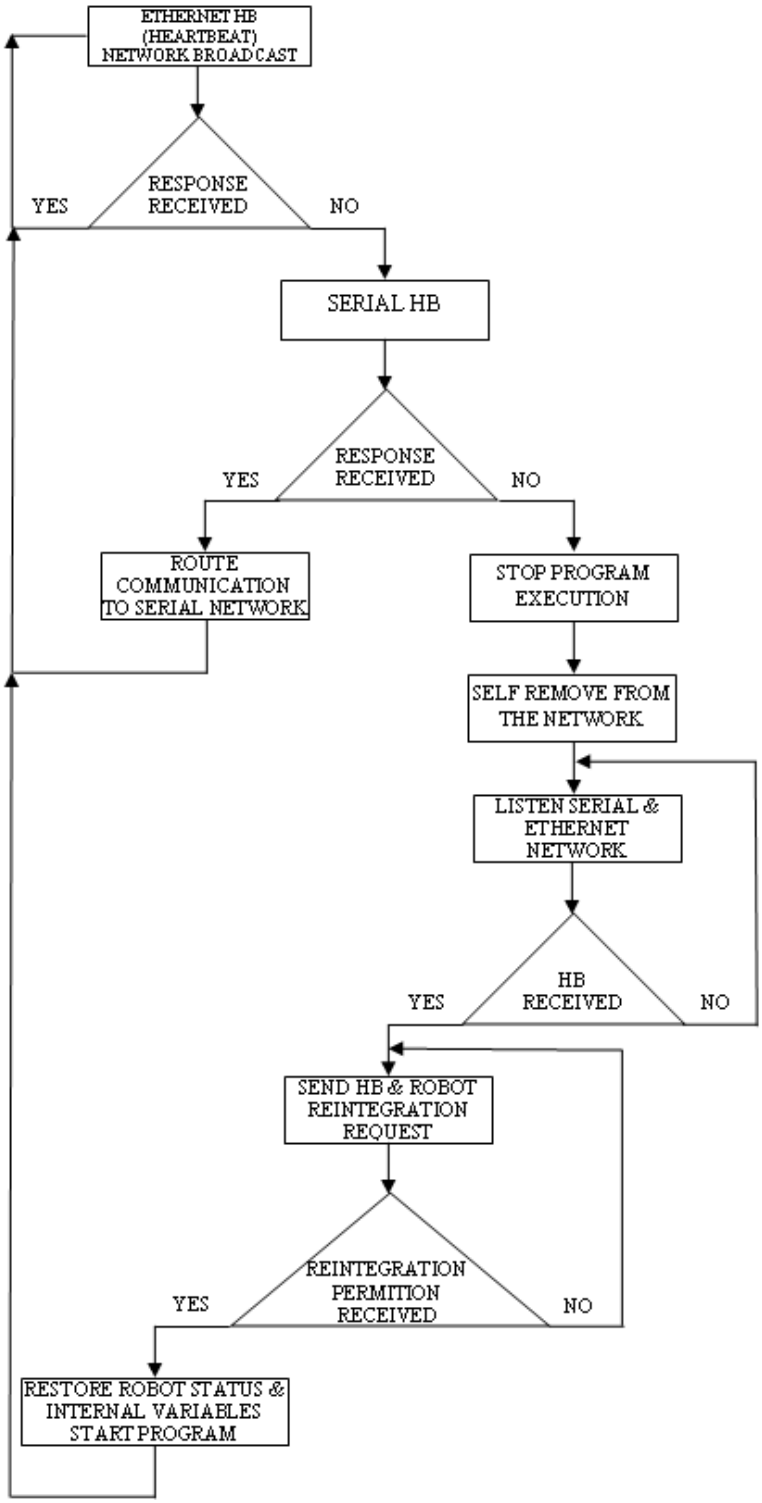

Fig. 24. Algorithm for network failure

The neighbours have sent the heartbeat packets using the serial lines and detected that they do not have any connection with the robot and announced the group leader (GL) which has removed the robot from the cluster and reconfigured the cell (Fig. 24). The neighbour R1 having the same working area as the affected robot R2 has loaded the variables values and the production program from the shared storage, and started the production resuming from the point where R2 has been stopped. The cell reconfiguration from the point where the serial lines where disconnected has taken 2.2 seconds.
Another communication test consisted in disconnecting both the serial lines and the Ethernet line at the same time; in this case the cluster tested the communication sequentially and cluster reconfiguration took 2.5 seconds. When configuring the cluster to test the communication lines in parallel, the reconfiguring took 2.3 seconds but the controllers processed a higher communication load.
The most unfavourable situation is when a robot manipulator is down; in this case the down time is greater because the application which was executed on that controller must be transferred, reconfigured and restarted on another controller. Also if the controller still runs properly it will become group leader to facilitate the job of the previous GL.
The solution is not entirely fault tolerant, but in some situations the solution could be considered as a fault tolerant system due to the fact that even if a robot controller failed, the production continued in normal conditions.
The research project will provide a portal solution for linking the existing fault tolerant pilot platform with multiple V+ industrial robot-vision controllers located in different R&D labs.

8. References

Adept Technology Inc., (2001), AdeptVision Reference Guide Version 14.0 Part Number 00964-03000, *Technical Publications, San Jose*.

Borangiu, Th., Nis, C., Case Studies in Open Architectures, *Printech Publishing House, Bucharest, 1999.*

Borangiu, Th., (2004). *Intelligent Image Processing in Robotics and Manufacturing*, Romanian Academy Press, Bucharest.

Borangiu, Th., F.-D. Anton, S. Tunaru, M. Manu, (2005); Internet-Based Messaging and Team Workplace Software for Remote Robot Control, *18th International Conference on Production Research ICPR'18, Fisciano-Salerno, Italy*

Brooks, K., R. Dhaliwal, K. O'Donnell, E. Stanton, K. Sumner and S. Van Herzele, (2004). *Lotus Domino 6.5.1 and Extended Products Integration Guide*, IBM RedBooks.

Lascu, O., Sayed, R., Carroll, S., Coleman, T., Haehnel, M., Klabenes, P., Quintero, D., Reyes, R., Jr., Tanaka, M., Truong, D., D., 2002. *An Introduction to Security in a CSM 1.3 for AIX 5L Environment,* IBM International Technical Support Organization. 1st edition.

Lascu, O., Bodily, S., Esser, M.-K., Herrera, M., Pothier, P., Quintero, D., Sebesteny, V., Steel, A., Prelec, D., Raymond, K., Socoliuc, A., 2005. *Implementing High Availability Cluster Multi-Processing (HACMP) Cookbook,* IBM International Technical Support Organization. 1st edition.

Matsubara, K., Blanchard, B., Nutt, P., Tokuyama, M., Niijima, T., 2002. *A practical guide for Resource Monitoring and Control (RMC).* IBM International Technical Support Organization. 1st edition.