A Holonic Fault Tolerant Manufacturing Platform with Multiple Robots

Theodor Borangiu, Florin Daniel Anton, Silvia Tunaru, Anamaria Dogar

Dept. of Automation and Industrial Informatics, University Politehnica of Bucharest, 313, Spl. Independentei sector 6, RO-060032
E-mail: borangiu@cimr.pub.ro

Abstract: *To be competitive, manufacturing should adapt to changing conditions existing in the market. Besides the market-based challenges, manufacturing firms also need constantly to adapt to newly developed processes and technologies and to rapidly changing environmental protection regulations. Modern automated manufacturing systems need robotized material-conditioning systems capable of moving materials efficiently throughout the entire production area. The objective of this paper is to describe the design, implementing, testing and validation of a holonic, fault-tolerant manufacturing control platform integrating robots with Visual Guidance (VG) for on demand material conditioning and AVI.*

Keywords*: Remote control, Network reliability, Flexible manufacturing systems, Monitoring, Fault tolerant systems*

I INTRODUCTION

In general robotic systems are composed by a multitude of software and hardware components which are susceptible to faults.
Some systems are designed to be fault tolerant which means 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 those components (the correcting actions are executed automatically) – which means that it will continue to provide the specified service despite of malfunctions. In many cases wrong behaviour in case of malfunction can lead to important economic loss.
Understanding and designing fault tolerant distributed systems is a recognized difficulty because in the same time one must know the normal behaviour of the system but also the complex situation that occur in case of a malfunction of a component. The difficulty of this activity is increased also because the lack of a structural coherent concepts and using a 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 in this paper.
The paper describes a system which 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.

II FAULTS: CLASSIFICATION AND SEMANTIC

A SC or cell server has a correct behaviour if its answer to a request is consistent with respect to the description of the service provided by this server. Malfunction is considered when the behaviour of the server disrespects the service description. Considering the server behaviour the following types of malfunctions are currently distinguished:

a) *Byzantine malfunction*: the malfunction is arbitrary and the server behaviour is fully random.
b) *Timing malfunction*: the server's response is behavioural correct, but has a delay greater than the waiting time specified for a correct behaviour.
c) *'Omission' type malfunction*: the server 'forgets' to answer to the clients requests.
d) *Answer malfunction*: the server answer is wrong (the value related to the answer type).
e) *Crash*: the server is not answering to any request until the system is restarted; depending on how the server begins to function at restart, there have been classified the following 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.

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.

In general if the malfunction semantic is more loose then the server is more expensive and more complicated to implement.

III THE FAULT TOLERANT SOFTWARE PARADIGM

Paradigms have been developed to help programmers to structure the fault tolerant software. So, each of those paradigms is used for miscellaneous applications reducing the developing complexity of such applications.

The most important paradigms for the proposed objectives are:

- *Transactions*: software structuring mechanism for applications which access shared data. The system guarantees three properties for transactions: atomicity, order and consistence.
- *Check pointing*: mechanism which, in case of malfunction, the activity can be restarted from a coherent state preceding the malfunction, state which is periodically memorized on a stable magnetic support.
- *Replicated State Machine*: the provided service is executed in parallel on few processors. The requests of each client are sent to every copy where are treated in a deterministic way.
- *Passive replication*: a service is implemented on few processors but only one is active (primary) and treat the clients requests.

IV THE SYSTEM STRUCTURE

The physical layout of the multiple robot development platform controlled according to the holonic manufacturing concept is shown in Figure 1. The platform contains four robots (of vertical and horizontal articulated type) accessing a double-path, closed loop conveyor for part transportation. Part diverting from the inner ring belt to the outer ring belt of the conveyor is performed under the control of a Programmable Logic Controller (PLC) directly accessible for program editing and downloading from an IBM PC-type computer.

Robot controllers (RC) have master positions over the four workstations, communicate between them and are connected in a fault-tolerant multi-network with Station Controllers of IBM PC-type.

A significant scientific contribution consists in adding fault-tolerance to the cell's communication system. A fault-tolerant communication architecture is proposed in Figure 2, providing redundancy at both the Station Controller level (a break down of any Robot Controller is detectable, the production tasks can be rescheduled to the remaining valid units for graceful degraded behaviour), and at the Station Computer level (replication of data bases for the IBM PC-type device terminals, reassignment of station computers in case of individual break down).

The Global Scheduler (GS) is implemented by a Cell Server (IBM xSeries Server), which transfers the recommendations of job scheduling to the cluster of Order Holons implemented in the IBM PC-type Station Computers (SCs) via a GS server-SC client Direct Network (Ethernet) [1]. This holonic control layer is *hierarchical*.

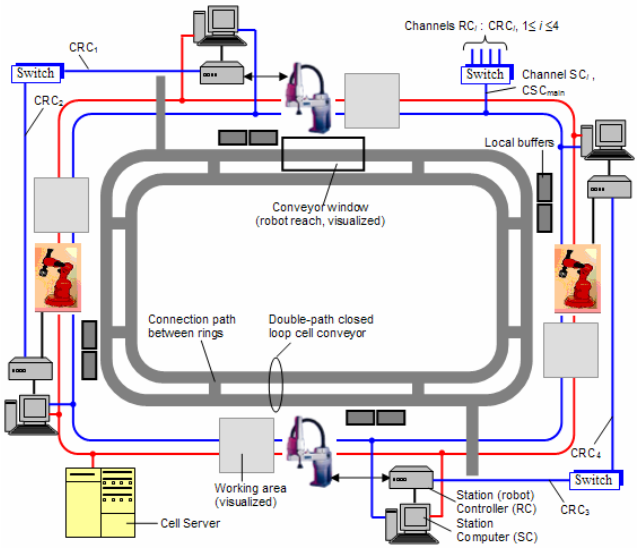

Figure 1
The layout of the platform with multiple robots, vision and holonic control

The cluster of SCs implementing the Order Holon (OH) and Material Handling Holon holonic control layers is interconnected to the process devices (Robot Controllers) via a SC server – RC client Switched Ethernet Network, creating a heterarchical fault-tolerant architecture:

- the failure of a Station Controller is detected by continuous monitoring via the direct serial links SC_i – RC_i and determines in consequence the rescheduling of jobs for the $(n - 1)$ remaining valid Robot Controllers j, $1 \leq j \leq n, j \neq i.$;
- if one of the Station Computers is down, its role is taken over by the remaining $(n - 1)$ workstations, as each SC data base is replicated and updated on line in all the other $(n - 1)$ ones;
- if the switch is down, the heterachical communication between the SC and the RC clusters still operates via the direct links SC_i – RC_i and the Ring RC link.

Finally, the vital inter-operational conditioning between device tasks (mutual exclusion, synchronization) is provided by a cross- connection I/O network.

V THE FAULT TOLERANT COMMUNICATION LEVEL

The communication level represents the key element of the management systems and command integrated with robot controllers of FMC.

A critical aspect in designing a communication level is the building, partitioning and the on-line/off-line data bases transfer, facts that involve the multiplication of the communication links insuring the global fault tolerant behaviour. 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.

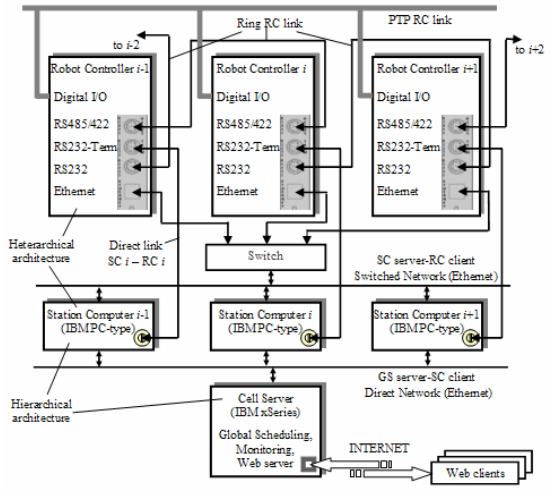

Figure 2
Fault-tolerant communication architecture for the multiple-robot platform with holonic production control

VI CIM DATA BASE PRESERVATION

The data base preservation is realized by using a shared memory named Tuple Space (TS) [2], memory used by processes for communication and synchronization. The TS represent a 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, $param_1$, $param_2$,..., $param_n$).

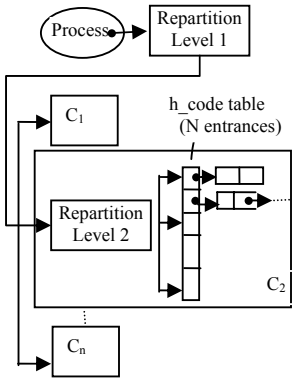

Figure 3
Tuples distribution

TS is a conceptual space capable to receive and memorize tuples, the basic operations defined on TS are: out, in, read, eval [3].

In the proposed architecture the TS of the system, in particular the CIM data base, is distributed on the controller's network. The distribution is achieved at two levels (Figure 3).

VII COPY MANAGEMENT AND COHERENCY BETWEEN THE COPIES

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

- Centralized management: the original C (Controller) makes the data duplication.
- Distributed management: the processes which request the access to write in TS send the tuple to the copies locations.

For the described CAM architecture a redundancy of minimum two different locations of the same data was chosen as is presented in Figure 4.

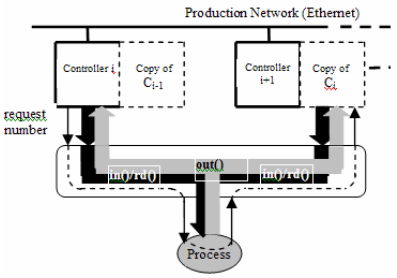

Figure 4
Distributed copy management.

The nondeterministic access to tuples and the associative selection makes uncertain the extraction of the same tuple from the two Controllers.

To solve this problem were used the request numbers (*rn*) which uniquely identify every out request at every TS. The request number is established by the original C and transferred to his replica using the appellant process. In this way the tuple will have the same *rn* in each location and the coherency is assured.

VIII DATA BASE RESTORING MECHANISM

The logical stages that a system must accomplish are:

- malfunction detection;
- blocking every process during the reconfiguration;

- data redistribution;
- unblocking the previous blocked processes.

The data redistribution stage represents the database "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, to keep the same fault tolerant structure for the rest of *n*-1 left C. This stage is shown in Figure 5, and has two sub-stages: first, the replicas are eliminated to avoid memory saturation during the second sub-stage. Then, a redistribution process handles the remaining data, and sends them to the new two locations.

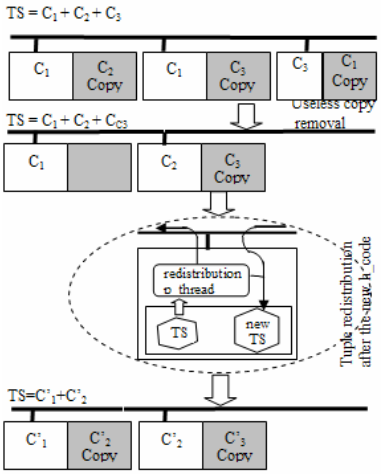

Figure 5
Tuples redistribution

In the presented architecture, malfunction detection is made by the processes that use the network. The malfunction is then reported to the GS which will decide whether it is the case or not for a new reconfiguration.
The chosen detection strategy uses a confirmation with time-out method; that means at each message exchange during the access to TS of a process, the process waits a confirmation (acknowledge). A process is capable to detect a malfunction only when trying to access one of servers that implement the TS.
The detection moments are when:

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

Conclusion

The project is under construction; most of the TS and GS functions are already implemented and tested on a pilot platform in the Laboratory of 'Robotics and AI' of the University Politehnica of Bucharest. The research project will contribute to the development of a high-performance infrastructure and concentration of a critical mass of highly-skilled human resources (MSc, PhD, PostDoc researchers), in national partnership, for the strategic ICT domain of *European technological platform*: Manufacturing of the Future – Manufuture.

References

[1] Borangiu, Th., F.-D. Anton, S. Tunaru, M. Manu: Internet-based Messaging and Team Workplace Software for Remote Robot Control, 18th International Conference on Production Research ICPR'18, Fisciano-Salerno, Italy, July 31-August 4, 2005

[2] Borangiu, Th., Nis, C.: Case Studies in Open Architectures, Printech Publishing House, Bucharest, 1999

[3] Kambhatla, S: Replication Issues for a Distributed and Highly Available Linda Tuple Space, PhD Thesis Oregon Graduate Institute of Science and Technology, USA, 1991