A DESIGN APPROACH TO FDI/FTC OF COMPLEX NETWORKED CONTROL SYSTEMS

J. Sá da Costa[*], B. M. S. Santos[*], M.J.G.C. Mendes[**]

[*] *IDMEC/IST TULisbon*
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
[**] *IDMEC/ISEL Polytechnic Institute of Lisbon*
Rua Conselheiro Emídio Navarro 1, 1959-007 Lisboa, Portugal

Abstract: The design of fault tolerant control (FTC) systems for large-scale complex networked control systems (NCS) is a difficult task due to the large number of sensors and actuators spatially distributed and networked connected. Despite the research effort on developing FTC systems most of these developments are designed globally leading to centralized FTC solutions inadequate to NCS. In this paper we present the first version of a toolbox based on multi-agent systems to design FTC systems for large-scale complex NCS. This toolbox is based on a decentralized FTC of NCS which relies on causal graph partitioning of the NCS digraph model and on intelligent distributed computing using multi-agents systems.

Keywords: Fault-tolerant control, networked control systems, decentralized design, multi-agents systems, graph partitioning.

1. INTRODUCTION

Communication networks are often adopted in many control and management systems to fulfil the information exchange and control signal transmission among system components of large-scale complex systems. These type of systems, named networked control systems (NCS), are systems whose sensors, actuators, estimator units, and control units are connected through communication networks having a pervasive mixed data flow with both time-critical data (periodic variables and events) and non-critical data (messages).

As any other system, NCS are subject to faults or malfunctions or simply performance deterioration of equipments, network or processes, forcing to interrupt the normal operation and, if not detected in early stages, frequently leading to expensive repairs. To avoid performance deteriorations or components damage and humans risks or, at a larger scale, ecological disasters, faults have to be found as quickly as possible and actions that stop the propagation of their effects or to diminish their impact on the performance of the system have to be taken. These actions should be carried out by appro-

priate control equipment through a fault diagnosis (FDI) system to detect, isolate and identify the kind of fault and its severity, and an appropriate reconfiguration or substitution of the control algorithm to adapt to the faulty plant condition. If the control reconfiguration is successful, the system operation will be satisfied and performance fully recovered, or slightly decreased in cases of severe faults. Figure 1 shows schematically the main components of a fault tolerant control system, namely the plant and its controllers, the FDI system and the reconfiguration control system.

In the last decade, a significant amount of research both theoretical and applied has been performed to develop FTC systems, i.e. control systems aiming to make the system stable and retain acceptable performance under system faults, (Blank *et al.*, 2003). However, most of these developments follow a global and centralized design, i.e. it considers the system as a whole and the resulting FTC system needs to monitor simultaneously all system variables to take appropriate actions.

Despite enormous research effort focused on NCS, only recently, few works address the FDI and the FTC problem in NCS (Ding and Zhang, 2006; Fang and Zhong, 2006; Sauter *et al.*, 2006; Vatanski *et al.*, 2006; Zhang and Ding, 2006; Zheng *et al.*, 2006). Even so, most of these FDI and FTC developments for NCS are globally designed and centralized. However, for large, physically distributed and heterogeneous NCS, where the communication network among the components of the system forces a distributed vision of the overall system, this approach is completely inadequate. Also, the network itself is subject to malfunctions and faults making the design of FTC systems even more difficult and making most of the times the previous methodologies inadequate to address this design. Thus, there is a pressing need to develop new approaches that intrinsically takes advantages of the characteristics of large, physically distributed and heterogeneous NCS.

Recently, (Sa da Costa and Mendes, 2006) propose a general framework for the FDI/FTC design of complex NCS, having in mind their distributed nature, using the distributed computing power given by software agents organized in society, forming a multi-agent system spread through the NCS, and minimizing the number of critical communications needed among parts of the system to guarantee safe operation and performance even in faulting conditions. The proposed approach is an autonomous hybrid framework involving simultaneously decentralized and centralized topologies, being independent of the methodologies used to tackle the design of the different stages of the FTC system. This new framework resembles a shell to be filled with fault detection (FD), fault isolation (FI) and reconfiguration (FR) algorithms developed for centralized FTC problems or new ones directly developed taking into account the specificities of the NCS under study.

The proposed approach gave good FDI results for the three tanks benchmark system (Sa da Costa and Mendes, 2006) but revealed the need to develop a computational tool to help the design of the autonomous multi-agents FTC system in a systematic way to make easy to deal with large and complex NCS. Here we present a first version of a Matlab Toolbox to design autonomous multi-agents FTC systems for large-scale and complex NCS.

In this paper Section 2 is devoted to summarize the main steps of the FTC design methodology adopted to deal with large-scale and complex NCS, emphasizing the assumed multi-agents decentralized architecture and the centralized supervision. Section 3 introduces the first version of the toolbox to design Fault Tolerant Networked Control Systems based on Multi-Agent Systems. Section 4 shows the basic steps to be used in a FTC design using the design toolbox for the case of the three tanks benchmark system. Finally, Section 5 gives some conclusions.

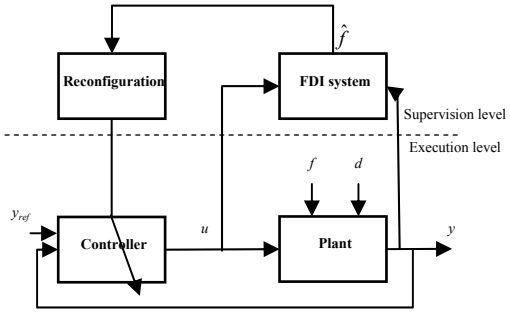

Fig. 1. Schematics of a Fault Tolerant System.

2. FTC DESIGN FOR NCS

The adopted method to design FTC systems for large and complex NCS relies on the inherent properties of these types of systems (Sa da Costa, 2006), i.e. their distributed nature allows us to make a partitioning of the NCS. However, this partition should minimize the number of critical communications needed among parts of the NCS to guarantee safe operation and performance even in faulting conditions in the process itself or in the communication network.

Thus, the adopted design approach relies in two main steps: first, to define an appropriate set of partitions of the NCS; secondly, to assign FTC multi-agent systems to each partition to perform local FTC. Global FTC is obtained by defining a proper communication scheme among the FTC multi-agent systems of all partitions.

This approach allows a large number of degrees of freedom during the design stage, mainly dependent of the NCS considered, the detail we want to reproduce it, and the degree of redundancy we are interested to perform the FTC. The number of partitions depends of the detail the designer consider appropriated to reproduce the NCS. Besides, since partitions can overlap, allowing in this way a certain degree of redundancy, it is possible to consider a hierarchical set of partitions being the FTC performed in a top down manner whenever necessary without the need to perform deep analysis every time.

Redundancy can also be obtained by using different methods (agents) concurrently for FD, FI, FDI and FR in each partition when performing FTC.

2.1 NCS partitioning

Bocănială and Sá da Costa (2004. 2005, 2006), proposed the use of a distributed FDI framework based on optimal partition of the directed graph (digraph) of the overall system that minimizes communication among different subsystems. This approach can also be easily extended to the FTC case.

The causal model of a system may be encoded as a digraph where the vertices represent the available sensors and actuators readings, and edges represent the causal links between these measurements. The complexity of the system is reflected in the complexity of the associated digraph. The partitioning procedure (*i*) considers the digraph as a map, (*ii*) partitioning this map into edge disjoint regions separated by borders formed by vertices, and (*iii*) assigns a dedicated FTC to each region. For step (*ii*), notice that each region may be treated recursively in the same manner as the initial map, therefore inducing a local hierarchy of FTC systems. The local expertise of the agents performing FD, FI, FDI and FR, as well as the interaction among them is used to robustly detect and isolate the faults in the partition and to perform control re-design. The use of this distributed scheme allows maintaining the focus only on those regions of the map that are affected by faults and also allows plug in and plug out parts of the NCS without affecting the FTC operation. Hence, FTC of a complex NCS becomes a tractable problem.

In order to comply with the natural requirement for a FTC computational time as small as possible, the previous partitioning is required to satisfy the following conditions: (*i*) the agents should be able to independently asses the state of the system in the assigned area, and (*ii*) the interaction between different agents should be kept as small as possible. The complexity of the interaction between two agents is given by the number of vertices located on the borders between the corresponding regions. The first condition is fulfilled by using the d-*separation* criterion to split the map in separate regions. This criterion offers a parallel between the causal independency and the vertex separation in digraphs. However, the d-*separation* criterion only applies to acyclic digraphs which are not usually the case in controlled system. To solve this drawback, a new methodology that allows cyclic causal models to be transformed into acyclic models has been developed without actually losing the structural and behavioural information given by the digraph loops, (Bocănială and Sá da Costa, 2006). The second condition is fulfilled by using the multilevel hyper-graph partitioning (Karypis, 2002). The analyzed causal model is transformed into a hyper-graph so that the following equivalence holds: the causal model has a minimal number of vertices on the partition borders if and only if the equivalent hyper-graph has a minimal number of hyper-edges cut by the partition borders. For more details and simple examples see (Bocănială and Sá da Costa, 2006 and Sá da Costa and Mendes, 2006).

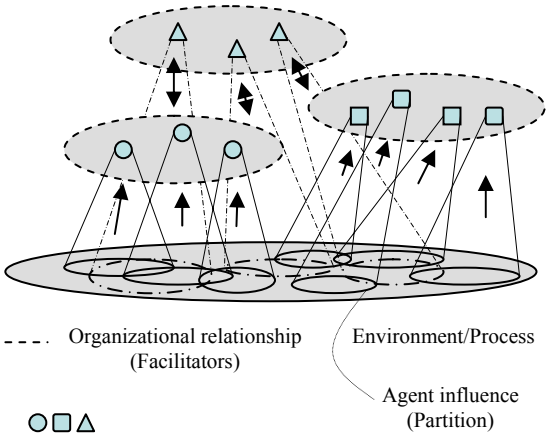

Fig. 2. Multi-agent system with facilitators.

2.2 Multi-agent systems in FTC

Agents are computational entities that are situated in some environment and are capable of autonomous action in that environment in order to meet its design objectives (Wooldridge, 2002).

The FTC framework proposed here is based on agents, or more exactly, in a society of agents usually called multi-agent systems (MAS), (Fig. 2). Several different multi-agent architectures can be found in the literature (Shen *et al.*, 2001). The architecture proposed here for the FTC multi-agent system is a unilateral relation MAS federated architecture coordinated by facilitators (Fig. 3), where one agent depends on the other, but not vice versa. The facilitators deal, respectively, with the unilateral communication among different FTC task agents, manages de corresponding task decisions and informs supervision agents.

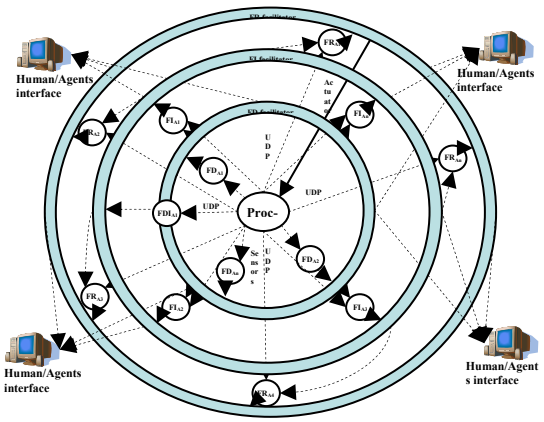

Fig. 3. Multi-agent system with facilitators adopted for FTC.

In this approach the agents are simpler, they have task separation but they are not fully autonomous because they have a facilitator dependency to communicate. A different approach relies on fully competition among the agents to achieve the goals and there is no need of negotiation and coordination between the agents. This last approach will be addressed in future work.

The use of MAS to implement FTC systems will bring several advantages, including: modularity and scalability, adaptivity; concurrency, dynamics, and reliability.

3. TOOLBOX FTNCS-MAS

In this first version of the toolbox to design Fault Tolerant Networked Control Systems based on Multi-Agent Systems (FTNCS-MAS) only the design of the multi-agent system is included. The partitioning algorithm will be integrated in the next version and the respective user interface to deal with digraphs still under development.

A new computational platform for FTC design using MAS must be in compliance (or at least with minimal requirements) with some standard rules for agent platforms (Bellifemine *et al.*, 2003). In this case, the standard is FIPA (Foundation for Intelligent Physical Agents) (FIPA, n.d.), which define a standard model of an agent platform, as shown in Figure 4.

Several multi-agent platforms exist (Vrba, 2003; Bellifemine *et al.*, 2003; Garneau and Delisle, 2003) to deal with different problems but none

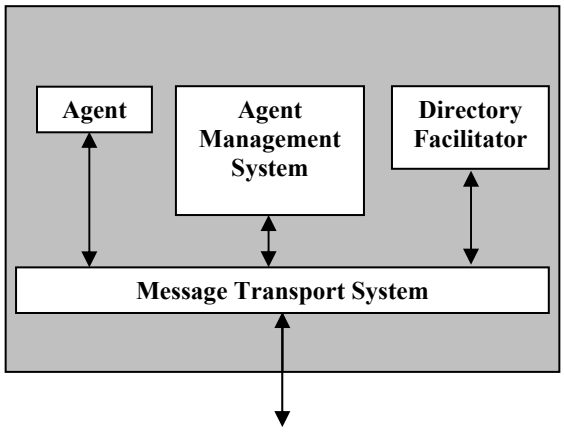

Fig. 4. The standard FIPA agent platform.

of them to deal with control systems tolerant to faults using Matlab/Simulink® environment, which is in our days the scientific tool to FDI/FTC research. Thus, the building of the FTNCS-MAS toolbox proposed here has being developed in MATLAB®, using Simulink®, xPC Target, and the Distributed Computing Toolbox. This last toolbox plus the MATLAB's Distributed Computing Engine (MDCE) ensure the services of a FIPA compliance MAS platform.

Simulink® is the environment for developing all platform components, namely: agents, facilitators, processes and supervisors. Data communication among the agents at this moment is done with the xPC Target's UDP blocks. Although, it may be an unreliable way of data transfer, it is the fastest standard data transmission available on this environment. The MDCE and the respective Distributed Computing Toolbox are an easy, fast and reliable way for remote deployment of the MAS components (agents and facilitators). The parallel job is used, thus giving a secure way for data transfer among agents, for cases that require starting, pausing, stopping and/or changing agents or facilitators. This is especially good for ensuring that a future implementation of an automatic platform reconfiguration can run without losing data in data transfers. An MDCE worker is assigned to each agent and to each facilitator.

The whole backbone of the platform has been developed. More specifically, a toolbox for MATLAB® has been developed, named FTNCS-MAS Designer Toolbox. Figure 5 shows the Simulink library blocks available. At the moment, the imposed limitations are: each agent can only receive data from the main process and/or one facilitator, and only send to one facilitator; a facilitator can only receive data from agents and supervisor; the supervisors can receive data from any facilitator and from the process; the process only has one input, but can have as many outputs as necessary.

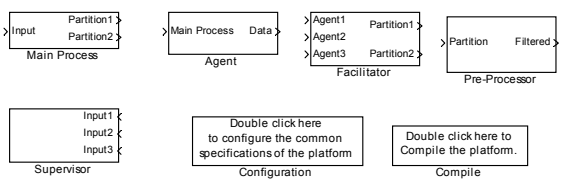

Fig. 5. FTNCS-MAS Designer Toolbox (basic blocks).

The amount of data, to be sent via UDP, has to be predefined manually, since the dimensions of the outputs are dependent from the variables to be used in the FTC system.

This allows an easy development of any FTNCS-MAS, since it automatically sets up the communications among agents (UDP/IP) and all of the, ready to use, files for executing the platform. To the developer of a particular platform is left the task of defining specific configurations, like the computers IP addresses and additional files to use; the micro-level architecture of the agents, facilitators and supervisors have to be done manually, since there are infinite possible configurations for each. Also, for each agent and facilitator is possible to define the worker specific script to be executed. This script handles the execution of the model and can communicate with other workers. A library of scripts are already made and prepared for receiving commands to pause/continue, end or reconfigure each model. To give a better view of what has been done, Figure 6 is a good example of a platform design. All arrow lines indicate UDP connections. Notice that the process and the supervisor blocks are to be executed outside of the MDCE worker group, thus allowing better control over what is going on. More agents can be easily added, as simple as copy-paste-change. The same goes for the facilitators and supervisors.

After the platform is designed, double clicking the available Compile block will activate the compiler function developed for this toolbox, which will: check if the design is done accordingly to the imposed limitations; afterwards will create the final models for each agent; for the process, it gives the send and receive blocks ready to be used. It also creates and copies the necessary files for the platform execution. Once compilation is complete, the platform is ready for deployment.

The UDP connections done by the compiler assigns to each send port block an UDP port; when the data to be sent is less than or equal to 512 byte, data broadcast mode is used (IP address 255.255.255.255), but when it is more than 512 byte, dedicated UDP send blocks are automatically added.

The toolbox is already prepared for a future possibility of an online full system reconfiguration,

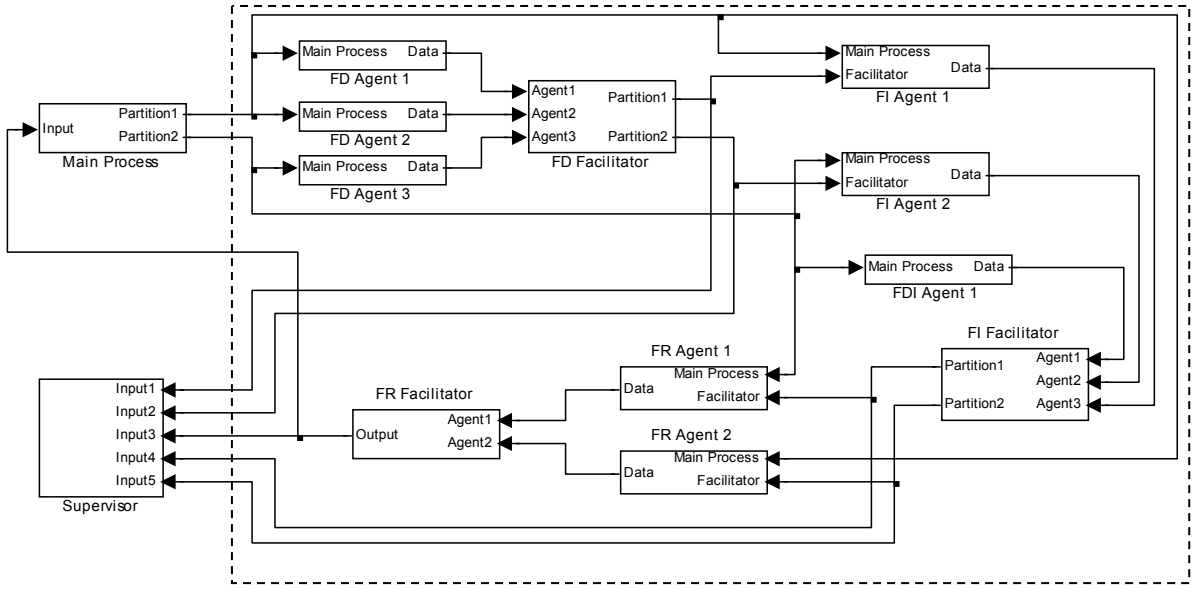

Fig. 6. Example of a MAS architecture layout of the FTNCS-MAS Designer Toolbox before Compilation.

requiring only that the model reconfiguration algorithm is done and called from the already made scripts assigned to workers.

4. THREE TANKS BENCHMARK

The AMIRA DTS200 Three Tank Process is used to help the development and the testing of the FTNCS-MAS Designer Toolbox.

Figure 7 shows the Simulink block diagram of the AMIRA DTS200 Three Tank Process. The plant consists of three plexiglas cylindrical tanks T1, T2 and T3. These are connected serially in pairs with each other by pipes at lower end. Located at T2 is a nominal outflow valve. The outflowing liquid (usually distilled water) is collected in a reservoir, which supplies the pumps B1 and B2. The necessary level measurements are carried out by piezo-resistive difference pressure sensors (PS1, PS2 and PS3). The control objective is to maintain the level in T2 by acting on pumps B1 and B2, whatever the disturbances or faults that can occur.

From Bocănială and Sá da Costa (2006) the causal graph model of the three tanks process can be partitioned in two regions with a certain degree of overlapping as shown in Figure 8. See Table 1 for details.

The resulting setup for FTNCS-MAS for this benchmark with two partitions is represented in Figure 9. Host 1 is connected to the process to acquire and pre-process the data from the sensors, sends (and archives) this data and receives control data (references, parameters tuning, etc.) through the Ethernet. Hosts 2 and 3 deal with the FTC-MAS in regions 1 and 2, respectively. Host 4 mainly supervises the all plant and the FTC procedure.

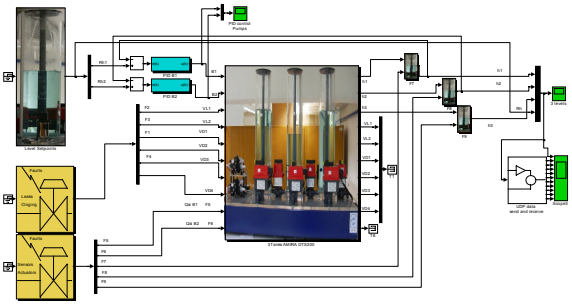

Fig. 7. Simulink block diagram of the Three Tank Process.

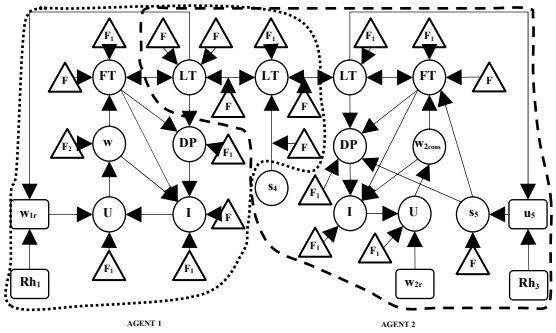

Fig. 8. Graph partition of the three tanks process.

Table 1: List of variables on the Graph partition

Variable	Description
Rh_1	level reference for tank T1
Rh_2	level reference for tank T2
h_1	level in tank T1
h_2	level in tank T2
h_3	level in tank T3
Dp_1	pressure difference on pump B1
Dp_2	pressure difference on pump B2
Q_1	flow through pump B1
Q_2	flow through pump B2
w_1	”speed” of rotation of pump B1
w_2	”speed” of rotation of pump B2
VD_2	Output - Tank T3 discharge valve
U_1	voltage on motor in pump B1
I_1	current on motor in B1
U_2	voltage on motor in pump B2
I_2	current on motor in B2
w_{1r}	reference speed on pump B1
w_{2r}	reference speed on pump B2

Each Host has an Intel C2D 3GHz processor and 2 GB of RAM with Windows XP_{prof}, Mat lab/Simulink®, Java virtual machine and MDCE service installed. Host 4 has additionally the Distributed Computing Toolbox® and Host 1 the Real Time Toolbox® from Humusoft®. This last toolbox allows working in real time, with the Simulink simulation environment.

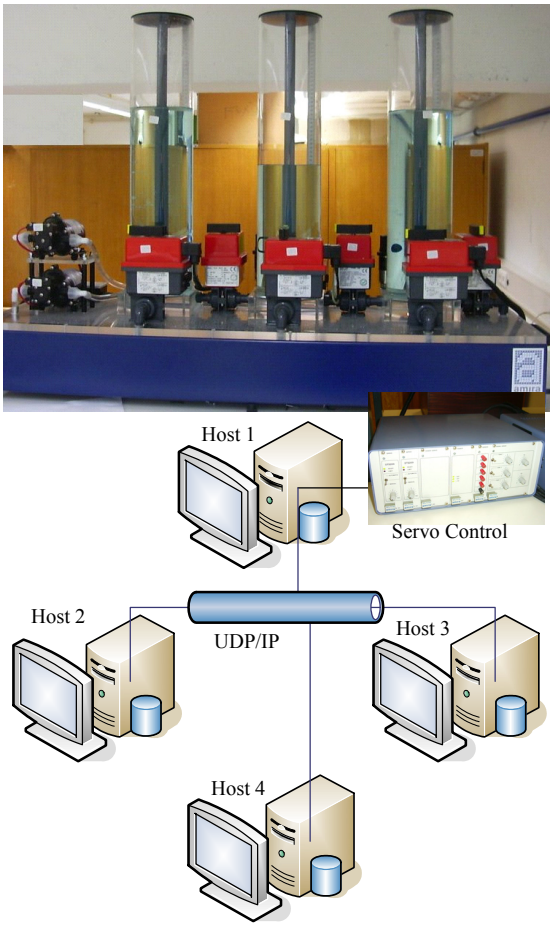

Fig. 9. Three tanks process and hardware used for developing and testing the FTNCS-MAS Designer Toolbox.

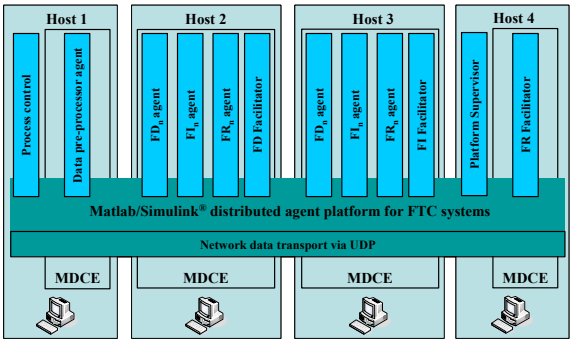

Fig. 10. FTNCS-MAS Designer platform for the three tanks process.

The FTNCS-MAS platform developed allows the agents to be distributed within the hosts, e.g. as shown in Figure 10. Since the hosts have core duo processors, other configurations could be used to deal with this benchmark, namely with only 3 or even 2 hosts. At the present time the backbone created has the basis for running 4 FD agents plus its Facilitator and 2 FI agents plus its Facilitator. This sums it up to eight workers. The link between the four computers is being done via Ethernet LAN. With three hosts, tests have already been done to determine the lag for a full lap (process→$FDAn_i$→FD facilitator→$FIAn$→FI facilitator→process). One of the tests was: in real time and with host 1 controlling the process, a sine wave (with a 0.01 s sample time) was generated in the host 1 Simulink control model and then, with the workers running agents in free infinite simulation, approximately 1.1 s lag was the result of the loop. The time analysys shows that the main bottleneck was the UDP receive at the process's end. Since it runs in real time, if the sample time of the UDP receive (in host 1 process control model) is decreased to 0.001 s, the lag also comes down to one tenth of the previous, i.e. 0.11 s.

5. CONCLUSIONS

This paper proposed a first version of a Matlab toolbox to design Fault Tolerant Networked Control Systems based on Multi-Agent Systems (FTNCS-MAS).

The multi-agent systems can be a valuable software engineering solution for the development of distributed FTC systems in complex distributed processes, where the complexity and the distribution of the processes ask and needs new approaches. In addition, the wide adoption of the Internet as an open environment, the increasing communication capabilities and the increasing popularity of distributed systems make the adoption of multi-agent technology a feasible and a very interesting solution. More tests, with this new FTNCS-MAS toolbox applied to larger NCS, must be done to confirm these first promising results.

ACKNOWLEDGEMENT

This work was partially supported by project POCTI/EME/59522/2004 and POCI-SFA-10-46-IDMEC, co-sponsor by FEDER, POCI 2010, FCT (MCTES), Portugal.

REFERENCES

[1] Bellifemine, F., G. Caire, A. Poggi and G. Rimassa (2003). Jade - a white paper. Exp in search ofinnovation 3(3), 6–19.

[2] Blanke, M., M. Kinnaert, J., Lunze and M. Starowswiecki (2003). Diagnosis and Fault-Tolerant Control. Springer.

[3] Bocănială, C.D. and J. Sa da Costa (2006). Causal Models for Distributed Fault Diagnosis of Complex Systems. Computational Intelligence in Fault Diagnosis, Eds. V. Palade, C.D. Bocănială and L.C. Jain. Series: Advanced Information and Knowledge Processing. Springer-Verlag.

[4] Bocănială, C.D. and J. Sa da Costa (2005). Novel methodology for partitioning complex systems for fault diagnosis purposes. In: Proc. of the 16th IFAC World Congress, Prague, Czech Republic.

[5] Bocănială, C.D. and J. Sa da Costa (2004). Novel Framework for Using Causal Models in Distributed Fault Diagnosis" In: Proc. Workshop on Advanced Control and Diagnosis, Karlsruhe, Germany, pp. 142-147.

[6] Ding, S., P. Zhang (2006). Observer-based monitoring of distributed networked control systems. In: Proceedings of the IFAC Symposium SAFEPROCESS'06, pp. 337-342, Beijing, P.R. China.

[7] Fang, H., H. Ye, M.Zhong (2006). Faul diagnosis of networked control systems. In: Proceedings of the IFAC Symposium SAFEPROCESS'06, pp. 1-12, Beijing, P.R. China.

[8] Garneau, Tony and Sylvain Delisle (2003). A new general, flexible and java-based software development tool for multiagent systems. In: Proceedings of the International Conference on Information Systems and Engineering (ISE 2003), pp. 22–29, Montreal, Canada.

[9] Gertler, J. (1998). Fault Detection and Diagnosis in Engineering Systems, Marcel Dekker, New York.

[10] FIPA (n.d.). The foundation for intelligent physical agents. Available: http://www.fipa.org/.

[11] Karypis, G. (2002). Multilevel Hypergraph Partitioning, Technical Report 02-25, Department of Computer Science and Engineering, University of Minnesota, USA.

[12] Sá da Costa, J. and M.J.G.C. Mendes (2006). FDI/FTC for complex networed control systems based on multi-agents. In: Proceedings of the 2nd Intern. Workshop on Networked Control Systems: Tolerant to Faults, Rende, Italy.

[13] Sauter, D., T. Boukhobza and F. Hamelin (2006). Decentralized and autonomous design for FDI/FTC of networked control systems. In: Proceedings of the IFAC Symposium SAFEPROCESS'06, pp. 163-168, Beijing, P.R. China.

[14] Shen,W., D. H. Norrie and J.P. A. Barths (2001). Multi-Agent Systems for Concurrent Intelligent Designand Manufacturing. Taylor and Francis. London, England.

[15] Vatanski, N., J.P. Georges, C. Aubrun and S.L. Jämsä-Jounela (2006). Control reconfiguration in networked control system. In: Proceedings of the IFAC Symposium SAFEPROCESS'06, pp. 349-354, Beijing, P.R. China.

[16] Vrba, Pavel (2003). Java-based agent platforms evaluation. In: Holonic and Multi-Agent Systems for Manufacturing. pp. 47–58. Springer Verlag. Berlin Heidelberg.

[17] Walsh, G.C., Y. Hong and L.G. Bushnell (2002). Stability Analysis of Networked Control Systems. IEEE Transactions on Control Systems Technology, 10(3):438–46.

[18] Wooldridge, M. (2002). An Introduction to Multiagent Systems. John Wiley & Sons, LTD. Chichester, England.

[19]Zhang, P. and S. X. Ding (2006). Fault detection of networked control systems with limited communication. In: Proceedings of the IFAC Symposium SAFEPROCESS'06, pp. 1135-1140, Beijing, P.R. China.

[20]Zheng, Y., F. Wang, J. Qin S, X. Yang and Y. Chen (2006). Fault detection for MIMO networked control system. In: Proceedings of the IFAC Symposium SAFEPROCESS'06, pp. 1141-1145, Beijing, P.R. China.