COMMUNICATIONS IN DISTRIBUTED CONTROL ENVIRONMENT WITH DYNAMIC CONFIGURATION

Raúl Alves, Miguel A. García

Dpt. of Systems Engineering and Automatic Control
University of Valladolid, Spain

Abstract: Distributed Control Systems (DCS) are based on assignments of control loops and supervision points to different units. In a first approach, these assignments are made statically and following given distribution criteria. This paper presents a proposal of communications system over a DCS which allows the interchanging of assignments between the control units. The objective is to provide a frame for dynamical load balancing in the system with reactivity against meaningful variables in the system such as changes in the process, traffic demands or computing load in every processor.

Keywords: Distributed control, dynamic configuration, protocols, load balancing

1. INTRODUCTION

The assignment of control loops to the control units of a Distributed Control Systems (DCS) is a main topic in which the first attempt is based on the knowledge of the plant and their plausible subsystems. This design is made off-line and has not into account performance criteria. This paper presents an implementation of a DCS with capabilities of dynamical changing of assignments. It is based on the DCS SICODI (García,1999) with modifications for including the new functionality. One related topic for this purpose is the load balancing in the system.

The DCS SICODI executes in several PCs over Windows NT connected with an Ethernet network between them and with a simulation of the process in the benchmark case. For this kind of case, the concept of generic Local Control Unit (LCU) allows to consider the same executable both for control and supervision units, and even for combinations of them and over the same type of standard platforms. The deploy of functionalities is based on the operation of configuration of every LCU, in which depending on the selections done the executable applies data acquisition and control algorithms, or interface elements to allow operators interaction.

Load balancing is a well known topic in multiprocessor homogeneous systems, where heuristics and algorithms for static and dynamic, off-line and on-line solutions have been developed. The proposed case of study supposes a generalisation to heterogeneous multi-platforms case from the point of view of DCSs.

The assignment of loops in a DCS is made traditionally off-line due to its association with the division of the factory in sections. The personnel in charge of this task are the process engineers as the persons with the best knowledge of the process and the interactions between its parts. Even the back-up policies are applied to the basic configuration for complete substitutions and with final results equivalent to the initial situation. Nevertheless, general theoretical approaches to the topic of distribution can be found, such as the Decomposition-Coordination methods (Singh, 1978) based on the off-line minimisation of interactions and the presence of a coordination module at a higher level. In the other hand, cases of massive migrations are likewise a well-known problem in cases of falling of any of the LCUs, and normally covered by the presence of back-up units associated to every one of them.

However, yet another cases can be considered covering partial re-distributions in relation with dynamic conditions of the systems. This type of decisions can apply fine adjustments to the assignments in the system, although in some cases can derive in important modifications in the long term once the results of progressive changes are evaluated.

Anyway, the main case of study from the load balancing point of view is one of uncertainties in the assignment of control, and of changes in the control structure and distribution. These cases can appear when the definition of sections and parts of a given plant is not an immediate task, and multiple choices must be evaluated on-line in order to obtain and compare their results. Besides, changing circumstances of, for instance, raw material, arrival conditions, failures or disturbances along the process, can be taken into account by commuting between different control policies and possibly transferring the control between alternative LCUs. Redundant actions from LCUs in flexible policies in which any of them can assume the assignment depending on availabilities are also related to this study.

Precedents of these cases can be found in changing environments in which control or communication conditions appear. (Grupen, 2001) presents a team of mobile robots devoted to perimeter surveillance tasks in which distributed control takes the form of location set points for each robot. The changing conditions consist in this case in possible invasions of the zone under observation or modifications of the zone itself, being the reaction a new assignment of set points for the robots if it is considered more appropriated than the original one.

The change of assignments can be focused from the communications point of view. So approaches taking into account explicitly this aspect are likewise very promising, with cases in which direct manipulations of the communication traffics can obtain new and optimized relationships between the nodes.

Both in this kind of cases the techniques that have been used are based on the reinforcement learning theory. It consists on the dynamic application of tests in lines of exploration based on gradient-based approaches. The estimation of goodness of results can be made with probabilistic measures of success, giving place to finish results when they are considered good enough. So these techniques can be considered as learning by doing in dynamic cases, with the possibility of adaptive solutions in changing environments and specific results for each case.

Application of reinforcement learning to batch processes is also possible as can be seen in (Martinez, 2000), where the time and conditions of evolution to new states in the sequence are improved in successive attempts until the arrival to solutions according to ranges of specification.

For this balancing to be possible in DCSs, the transference of control loops from one machine to another must be solved. This is made by designing communication protocols between the LCUs, in such a way that dialogs are established between them for this purpose. Related requests are in charge of getting system and platforms information for their on-line comparison.

SICODI has been adapted in order to achieve this objective, creating a set of communication protocols and functionalities allowing the transfer of control loops and the monitoring of the whole system. Such a monitoring is made from a server application made ad hoc, from which it is possible to integrate supporting units for partial load distribution by using the atomic functionality of transference of control loops without loose of control during the transition.

The rest of this paper is structured as follows: Section 2 is devoted to the architecture of the system, while in Section 3 the levels of communication developed are also shown in detail. Section 4 explains the details of the protocols for implementing the transference of loops and section 5 explains how the basic cases of load balancing can be configured in SICODI. A little final section comments the future work to be made.

2. SYSTEM ARCHITECTURE

This Section describes the system over which the load balancing functionality is added. The working environment is one of training simulation and the functionalities are implemented in the communication protocols of the DCS.applied to the system.

2.1 Working environment

The starting point is a Training Simulator (García, 1999 II) composed by the DCS SICODI consisting of a set of interconnected LCUs; a dynamic simulation of the different plant sections; and an Instructor Console (IC) from which an instructor imposes changes to the operators in front of the LCUs and evaluates their responses. Fig. 1 shows the system architecture.

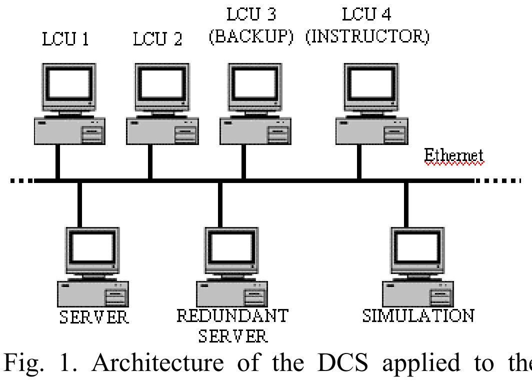


Fig. 1. Architecture of the DCS applied to the Training Simulator

All the applications in the Training Simulator have been developed for Windows NT operating system over PC platforms. The simulation is implemented in ACSL language and the IC is in fact based on a LCU plus extra functionality deployed under configuration. Only one of the

LCUs can act as IC at a time, acquiring special access to the simulation in order to obtain more information and the ability of imposing changes hidden to the rest of LCUs but for their effects.

2.2 Distributed Control System

The DCS SICODI applied to the Training Simulator consists on several units in which the concept of generic LCU is developed by configuration. This is possible due to the homogeneous nature of field communications with respect to the ones between LCUs, and the not special hardware needs at field level.

Communication among the LCUs is made through a server application in charge of the delivery of packets between them, the collecting of the status and the metrics of the communications, and the execution of the backup and load balancing mechanisms. In the other hand, communication with the simulation is provided by a link application for all the LCUs in their accesses to the simulated plant.

Fig. 1 shows a typical configuration of the DCS, with several LCUs in charge of control and supervision, a backup LCU covering possible failures, another one configured with IC properties, and primary and secondary servers for providing communications services.

3. LEVELS OF COMMUNICATION

The set of modules taking part of the Training Simulator requires a connection between them in order to support the communication needs. The basic primitives for this purpose are sockets over Ethernet. Different channels are devoted to every case of communication attending their meaning and importance. The three designed cases can be considered levels taking into account the type of interested modules and the relationship established between them, as can be seen in Fig. 2. Server parts of the sockets are located in the link application for the first level, and in the server application for the second and third levels.

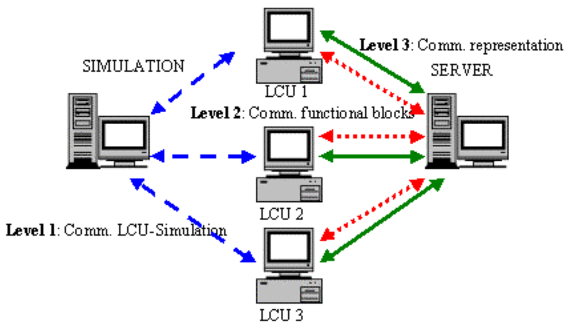

Level 3: Comm. representation
SERVER
SIMULATION
LCU 1
Level 2: Comm. functional blocks
LCU 2
Level 1: Comm. LCU-Simulation
LCU 3

Fig. 2. Scheme of the system with the levels of communication.

3.1 Level 1: Communication LCU-Simulation

This level of communication is devoted to the communication of every LCU with a link application in charge of administering the accesses to the simulation. Information in this channel consists on controlled variables taken from the simulation for the LCU to which they are assigned, and manipulated variables received from the LCUs for being ordered and passed to the simulation as control results every sample period. Likewise, information to and from the Instructor Console uses this channel. This communication level has high priority as it is expected for the nature of the information in it.

3.2 Level 2: Communication between function blocks

The interchange of data from function blocks is provided by this level of communication. It is based on standards for function blocks such as the IEC 61499 for distributed control (Christensen, 1999) or the Fieldbus Foundation specification. The type of information transferred depends on the type of connected blocks, covering the data acquisition for being treated remotely, control structures between LCUs (cascade, feedforward), and alarms treatment and acknowledging through the system. Besides, it allows the design of event-driven sequences in which data from remote blocks are evaluated or modified. The priority of this communication level is high as corresponding to control data flow, although a distinction can be considered in favour of the level 1.

3.3 Level3: Communication for representation

This level of communication is in charge of the communication between the LCUs for interchanging of data in order to be represented under requests of LCUs in which they are not assigned. The types of representations that can be configured are tables, bar diagrams, synoptics and trends including past values. Besides, this level of communication supports the remote modification in cases such as the change of set point, toggle between auto and manual, or the change of manipulated value in manual, for loops not assigned to the LCU from which the changes are imposed. The data received for these purposes in the LCUs are maintained in dynamic storage with a life linked to the interval in which the requests are active. With all these services and a correct complementary configuration of the LCUs, the objective of a unified representation in the system independently of assignments can be achieved. The priority for the communication in this level is low, thus avoiding conflicts with the information in the other levels and taking into account the lower importance of this information in comparison with the one in the others.

4. TRANSFERENCE OF LOOPS

A set of protocols has been implemented with the objective of interchanging the workload between LCUs in a wide sense. Basically, this workload consists on control loops assigned to the different LCUs, together with the event-driven sequences associated to them.

The loops assigned to a LCU appear with all their parameters in the configuration file of the LCU and, together with the process values obtained during the execution, in the corresponding part of the distributed database. The server maintains the information of assignments in the system, distinguishing the cases of transferable loops (most of them in cases such as simulated plants and even in real plants with fieldbuses) and fixed ones associated to specific hardware.

The implemented protocols cover the functionalities of transference of loops between LCUs and the monitoring from the server together with information about the available LCUs, loops assigned to them at every moment, and even the configuration of the criteria for the load balancing. The protocols are grouped in a set of actions that must be taken in the appropriated order, as explained in the next paragraphs.

4.1 Stopping the system

The first step for proceeding to carry any transfer involving the configuration of control in the LCUs out is to stop the simulation time and the treatments of control. This is possible thanks to the simulation environment in which it is applied, although once the transferring mechanisms are implemented and optimised, their use in real systems and conditions are applicable with minimal interferes or loose of control during the transient interval. Thus stopping the system is a particular advantage in simulation environments which can be exploited for acquiring knowledge and tools useful for any kind of distributed system.

A protocol has been developed in order to allow the server to put a request of stopping in the system. The different LCUs execute the stopping and confirm their status to the server, as shown in the chronogram of Fig. 3. The five steps of this process are:

- Step 1: The server decides to stop the system and sends a request packet to the LCUs.
- Step 2: The LCUs receive the stopping request and send to the server an acknowledgement packet.
- Step 3: Once the server has received the acknowledgement from all the LCUs, it sends the stopping packet to them. The LCUs make their arrangements for cancelling their control timer.
- Step 4: Once the LCUs end their internal actions, they send a ready packet to the server.
- Step 5: Once the server has received all the answers from the LCUs, it considers the system stopped and allows the beginning of transfers between the LCUs in stopped system conditions.

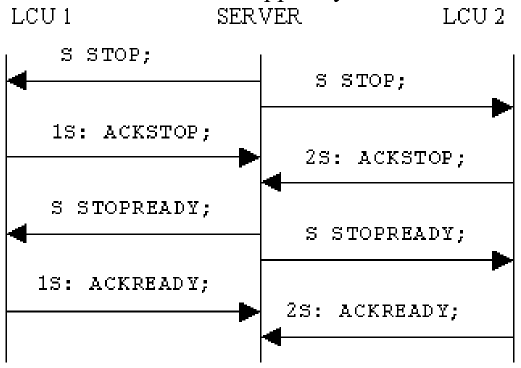


Fig. 3. Chronogram of the stopping of the system

4.2 Transference of loops between LCUs

The transference of loops in a DCS is the action by which the configuration of assignments is modified. As corresponding to the distributed environment in which it takes place, the transference of loops takes the form of a set of communication packets derived from a protocol that covers and order the successive actions.

When a control loop is assigned to a LCU, a set of configuration parameters are deployed locally in it in order to apply its control capabilities appropriately. In case of transference of the control loop to another LCU, actions must be taken for the control loop to finish its execution in such a LCU and for it to be applied from an alternate LCU. For such a purpose the configuration parameters must be sent from the original LCU to the one in charge of the assignment from that moment on, together with the elimination commands in the previous LCU and the activation commands in the new one.

In the same way that in the case of the stopping of the system, all these objectives are implemented as a communication protocol with which the LCUs and the server establishes a dialog between them until all the changes are propagated and accepted by the nodes taking part of the operation. As an example, Fig. 4 shows a simplified chronogram of the necessary steps for transferring the loop marked with the ordinal 35 from the LCU no. 1 to the LCU no. 2. In outline, these steps consist on:

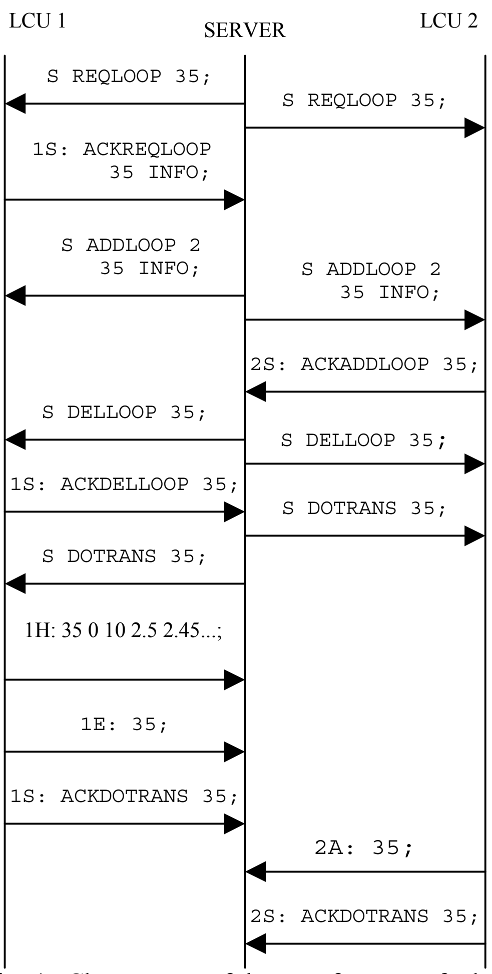


Fig. 4. Chronogram of the transference of a loop from one LCU to another one.

- Step 1: The server decides to transfer one loop (or a set of them) from one LCU to another one and puts in the system the request of transference. A set of lists maintains a centralised updating of the loops to be transferred, from and to what LCU must be made, and the status of such a communication. In fact, this set of lists must be ready even before of the stopping of the system. After the stopping, a call is made to the function that puts in the system the request packet for the loops under transference according to the items in the lists.
- Step 2: The LCUs receive the request, being ignored by the LCUs with no assignment of the loops under transference. In case of positive assignment, a packet is made with the configuration parameters of the loop under transference in a due form, being sent immediately after its packing.
- Step 3: The packets are received in the server, being merged and modified for their sending to the LCUs together with the information of the selected LCU for the new assignment.
- Step 4: Once the message for adding loop arrives the LCUs, it is partially unpacked for obtaining the number of the selected LCU for the transferred loop. If the LCU is the selected one, it unpacks the rest of the message and gets the configuration parameters of the loop. In order to avoid inconsistencies and the risk of more than one LCU trying to have the loop assigned, the new LCU does not begin to carry the new loop on. Instead of this, the LCU adds its availability in the list of pending actions, notifies the server and waits its licence for beginning the assignment.
- Step 5: Once the server receives the notification of the LCU trying to obtain the assignment, it sends to the rest of the LCUs the request of elimination of such a loop from their own assignments.
- Step 6: The LCUs receive the request of elimination. If they have not the associated loop assigned, they ignore it. In case of assignment, the LCU begins the action by adding it to the list of pending actions as the LCU with the new assignment made with its availability. Once this adding is made, the LCU sends an acknowledgement message to the server.
- Step 7: With all the confirmations collected by the server, it begins the actions for the new assignment to be effective, notifying to the LCUs related to the assignment that it is allowed.
- Step 8: The rest of LCUs receive likewise the notification and get the number of the corresponding loop. All of them (including the LCUs involved in the assignment) update the lists of representation requests in order to redirect them (or to eliminate them in case of self-assignment). Just before it, the LCU with the old assignment prepares all the data of past values of operation and collected in its corresponding part of the distributed database (supported in its platform) in order to send them to the server. Once this is made, the LCU with the old assignment modifies its configuration file taking into account the elimination, and sends two messages to the server: one for the acknowledgement of the actions made, and yet another for the updating of the inner lists of assignments in all the LCUs, in which the searching of information under request is based.
- Step 9: The arrival to the server of both the acknowledgements of the action of transference from the LCUS with the old and the new assignment are considered the successful signal for the whole operation from the server.

4.3 Re-starting the system

The last part in all the operations of reassignment is the new starting of the system after the transferences and acknowledgements. Basically, the starting consists on the release of the time variable and their consequent operations of acquisition and control each sample period. These operations must be coordinated between the LCUs in the system together with the simulation itself. The chronogram of the re-starting is shown in Fig. 5, and consists on three steps:

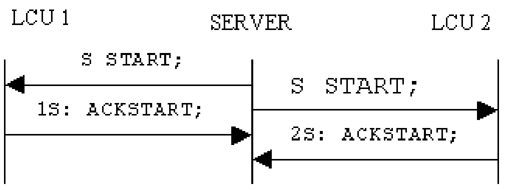


Fig. 5. Chronogram of the re-starting of the system

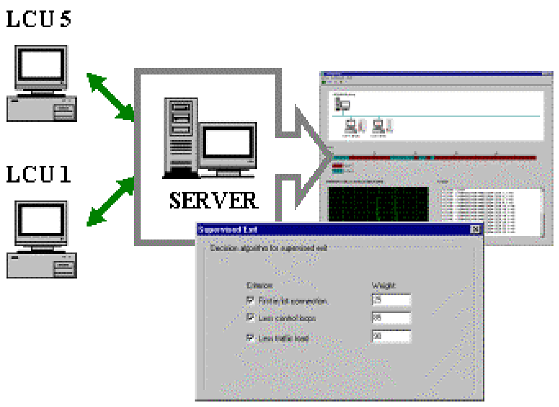
- Step 1: The server takes the decision of re-starting and sends a packet to the LCUs.
- Step 2: The LCUs receive the packet of re-starting and activate the timers in such a way that they recover their normal operation. Once this is made, they send to the server a packet of acknowledgement.
- Step 3: Once the server has received the acknowledgement of re-starting from all the LCUs, the system is considered in on-line mode again. Synchronisation protocol is in charge of the new adjusting of the LCUs to a common time.

5. BASIC IMPLEMENTATION

The case of SICODI applied to the Training Simulator is specially appropriated for this kind of techniques, due to the homogeneity of field level and the ease of access from all the LCUs to the simulation. Changes in the control structure can be applied on-line, supported during transient intervals by the level of communication between blocks and finally derived to a reassignment of loops in the LCUs of the system. Besides, for the cases of hybrid configurations with both assignments and representation, traffic conditions are subjected to relevant changes due for instance to requests for that representation. Under given circumstances and if another criteria are not against it, reassignments for balancing the requirements of traffic can be an option to be considered. The dynamic nature of these cases appears in the changing conditions of the execution and in the event-driven requests applied by the operators from their sites. All these properties together with the changes in the simulated processes give place to a platform of development and evaluation for different load balancing criteria.

Criteria for on-line load balancing and selection of LCU for assignments are already available in SICODI, although based in simple decisions such as the number of assignments or the traffic in every level. These criteria complement the basic one for backups, which is based in priorities of the LCUs forcing correspondences between groups of LCUs and corresponding backups, with the particular case of one-to-one associations. The algorithms implementing these criteria are located and centralized in the server application, from which the decisions are taken.

The monitoring of the system from the server, based on representations of the LCUs together with their assignments and loops under request between them, is extended to the possibility of on-line configuration of the different criteria, even with weighted combinations of them as can be seen in Fig. 6.



Decision algorithm for supervised exit

Criterion	Weight
☑ First in list connection	25
☑ Less control loops	35
☑ Less traffic load	30

Fig. 6. Monitoring from the server and on-line configuration of load balancing criteria

6. FURTHER RESEARCH

The benchmark system is already working with the simulation of sections of sugar plants and applying basic criteria of load balancing. The implementation of criteria based on reinforcement learning is under study with promising results.

REFERENCES

Christensen J. H. (1999). *Basic Concepts of IEC 61499*.

García M.A., Prada C. (1999). SICODI: *A Configurable Real-Time Distributed Control System*. 7th IEEE International ETFA'99.

García M.A., Acebes F., Prada C.(1999 II). *A Dynamical Training Simulator for Beet-Sugar Factories*. CITS Procee Antwerp. Ver Bartens

Grupen R. (2001). *Task-Level Design for Concurrent Control Interactions – Distributes Multi-Robot Strategies in Open Environments*. Univ. of Massachusetts, CS Dpt., int. report.

Martinez E.C. (2000). *Batch Process Modeling for Optimization Using Reinforcement Learning*. Computers&Chemical Engineering.

Singh M.G., Titli A. (1978). *Systems: Decomposition, Optimisation and Control*. Pergamon Press.