Hierarchical and Distributed Embedded Control Kernel


*Departamento de Ingeniería de Sistemas y Automática (DISA)
**Departamento de Informática de Sistemas y Computadores (DISCA)
Universidad Politécnica de Valencia, Camino de Vera, s/n. 46022 Valencia (Spain)
(e-mail: rausifer@ai2.upv.es, jacopa@ai2.upv.es, jsimo@disca.upv.es, pblanes@disca.upv.es)

Abstract: This paper presents how to get a high control performance and a reliable operation, by means of a suitable combination of several Embedded Control Systems. For this purpose, a hierarchical and distributed control model is proposed. The model holds a set of activities that should be executed on it, such as change, switch and delegate new code of controllers into embedded nodes. All of these activities are managed by a middleware component following the control kernel concept. This model was tested on real processors interconnected in a CAN network, using a XScale microcomputer with a real time operating system (RTLinux) running a high level controller (GPC) and a dsPIC microcontroller for signal acquisition and delivering of control actions.

Keywords: Embedded Systems, Control Kernel, Distributed Control, General Predictive Controller, Real Time Control.

1. INTRODUCTION

Distributed Control Systems (DCS) are used in industrial and civil engineering applications to monitor and control distributed equipment with remote human intervention. Moreover, these systems use a network to interconnect sensors, controllers, operator terminals and actuators.

While computing power of embedded systems is increasing over time, the networking technology trend is to move from control-specific networks to Ethernet based infrastructures and wireless communication. In this scenario the control strategy should be tolerant to variations in the message delays as well as bandwidth availability. Some work has been done in communication protocols with traffic characterisation, bandwidth allocation and clock synchronisation (Coronel et al., 2005). But this kind of cyber-physical interaction motivates a big amount of innovations in many Information Technology related fields including DCS architectures and controller design (Lee, 2006).

The DCS architecture assumed in this work is based on the “control kernel” concept (Albertos et al., 2006). These principles of organisation are intended to the automatically distribution of control code in a DCS insuring safe operation. The functionalities are provided by a specific middleware responsible of information and code distribution over the communication infrastructure.

Section 2 presents a distributed control model and gives an overview of the main components of the architecture. In the section 3, the main elements of the experimental platform are presented. Afterward, in section 4, two experimental cases studies are presented using the developed experimental platform depicted in section 3. The first experiment illustrates the controller switching mechanism while the second experiment present the implementation of a more complex hierarchical controller that uses a predictive controller (GPC) to develop a safe control system. Finally, section 5 summarizes the conclusions of the work.

2. AN APPROACH TO DISTRIBUTED EMBEDDED CONTROL

Safety is a crucial issue in embedded control systems. Independently of the number of variables to be controlled by the same processor, the systems with hard real time requirements must ensure the delivering of control actions to all actuators. The quality of the delivered signal can depend on the processing level: used data, the computational algorithms, resources availability, among other, but always must ensure the safe performance of system (Albertos et al., 2006).

Apart from components malfunction, in complex DCS, safety can be affected by the variation of the controlled system dynamics that requires controllers switching, missing execution deadlines, loosing messages and variation of communications delays. In this context, for running control applications in a safe mode, the following activities should be taken into account:

- Communication links with other activities should be activated.
- Some data should be recorded, displayed, stored and updated.
- It exist at least one controller that computes the control action based on available data at each time instant and using the predefined algorithms.
- According to the system behaviour, it must advance actions such as: disconnect and switch controllers. Controllers are parts of code that run spread in a distributed environment.

1 This work is part of the KERTROL project DPI2005-09327-C02-01/02 under the sponsorship of the Spanish Ministry of Education and Science.
• If the control action has not been delivered by the current controller on time, a safe control action should be delivered at time required to the process. This signal may be the result of a simple calculation (but sufficiently safe) an emergency shutdown or simply a safe back-up response such as: keep unchanged. Note that this operation can be interpreted as a controller switching.

For this purpose, a distributed embedded control model can be defined as composed by two node types: light nodes and service nodes (Fig. 1). Service nodes are powerful embedded computers running a full featured RTOS and complete networking with I/O capabilities. Light nodes are small and low power consumption SoC processors with limited computing and networking capabilities but complete I/O features.

**Service node**

![Service node](image)

**Light node**

![Light node](image)

---

**3. EXPERIMENTAL PLATFORM ELEMENTS**

### 3.1 Embedded Nodes

The light and service nodes are based respectively on a dsPIC microcontroller and a XScale microprocessor.
The Microchip dsPIC (Microchip 2006) combines the huge computation speed of a Digital Signal Processor (DSP) with a powerful 16-bit microcontroller (MCU), to produce a tightly coupled single-chip single-instruction stream solution for embedded systems design. This dsPIC device achieves speeds of up to 30 MIPS, is efficient for C programming and has Flash program memory, data EEPROM, data SRAM, powerful peripherals and a variety of software libraries.

On the other hand, the XScale embedded architecture has been chosen as development platform for our service node due to their low power consumption, their high performance and their low cost. All the generations of XScale are 32-bit ARM v5TE processors manufactured with a 0.18 μm process technology. This processor support changes of core frequencies between 100 MHz and 400 MHz for optimization of power consumption with a top of computing power of 700 MIPS.

Service node uses the Real Time Operative System “RTLinux” (Yodaiken and Barabanov, 1996) which offers characteristics of a hard real-time system in a multi-threaded real-time kernel and which can be used as embedded O.S. But, for this it was necessary to make a porting to adapt the original RTLinux source code to XScale architecture. We have developed an RTLinux version with support to XScale architecture, available on http://rtportal.upv.es/

3.2 Communication protocol

Although there is a great variety of real time buses, CAN (Controller Area Network) (CiA, 1996) is one of the preferred solutions to communicate distributed real time systems (Coronel et al., 2005). Therefore, for our experimental case, the communication middleware will use CAN as infrastructure to interconnect the two units.

In order to incorporate this communication protocol into XScale node, it has been designed an expansion board with a PIC microcontroller which has an embedded CAN chip. This PIC communicates with the XScale through a double port RAM memory. Moreover, the respective CAN drivers have been developed for RTLinux. On the other hand, the dsPIC node already has an embedded CAN chip.

4. EXPERIMENTAL WORK

In order to test the characteristics and capabilities of the proposed distributed control kernel model (Crespo et al., 2006), two cases study will be presented in this section: first, we evaluate the ability to switching simple controllers located on different computation nodes interconnected through a shared communication channel, and after that, we take advantage of the distributed computing availability to run a predictive control algorithm to control a real process and provide fault tolerance to communication sporadic error.

4.1 Case Study 1. Switching of Process Controllers

The switching of controllers is one of the key features in the control kernel model to run control applications in a safe mode. For this case study the light node is directly connected to a simulated process and it send information about process state to the service node through the CAN communication bus.

4.1.1 Description of the Simulated Process

The light node has been connected through a DAQ card to a PC running a simulated system in MatLab with Simulink and Real-Time WorkShop toolboxes (MathWorks 2006). The system is a simulation of the real HUMUSOFT CE 152 Magnetic Levitation educational scale model. The simulated model transforms the error signal into a real analog one through the DAQ analog output. An analog input of the DAQ is used to get the control or feedback signal.

As shown the figure 3, the error signal \( (e(k)) \) is directly sampled by the dsPIC and its value is transmitted to service node by means of the communication bus. The light node also applies directly the final feedback signal \( (u_f) \) on the control process, whose \( u_f \) can be the \( u_o \) or \( u_s \) signal. This last depend of the switching mode.

\[
\begin{align*}
    u_s(k) &= q_0 \cdot e(k) + q_1 \cdot e(k - 1) \\
    u_o(k) &= u_o(k - 1) + q_0 \cdot e(k) + q_1 \cdot e(k - 1) + q_2 \cdot e(k - 2)
\end{align*}
\]

where \( e(k) \) is the error in the instant “k”, and \( u_o(k) \) is the calculated control signal to apply to the simulated system.

Several tests have been made on a simulated environment to obtain the optimal coefficient values for the given process. With that result a local control for the simulated plant could be done at 5ms control cycles.

The result is sent from the light node to the system by means of the communication bus. The service node applies directly the final feedback signal \( (u_f) \) on the control process, whose \( u_f \) can be the \( u_o \) or \( u_s \) signal. This last depend of the switching mode.
monitors and analyzes the sensory data and the control action $u_{o(k)}$ received through the CAN bus.

For the first situation, if $u_{o(k)}$ is not received or has some delay, then, the light node (dsPIC) will apply his calculated control action ($u_{o(k)}$) (see figure 2). Thus, if an internal timer is up to a critical time delay, then CKM Runtime switches to the local PD controller into the dsPIC. A communication delay may represent that the light node is not working properly or the communication network is busy or down. This switching ensures that always exists a control action ($u_{o(k)}$) to be sent to the process. The figure 4 shows that the switching no affect the evolution of the process.

For the second situation, the control action produced by the light node ($u_{o(k)}$) is analyzed to determine if it is suitable to control the system. When $u_{o(k)}$ is detected as wrong, immediately the signal $u_{o(k)}$ is switched to a safe signal $u_{o(k)}$ (see figure 2). The present and accumulated error signal values is analysed on the dsPIC, and if the error parameter exceeds a programmed value, the regulator changes to the service node controller. In the figure 5, the regulation is working successfully until the second 5 a lost data is simulated, then the process begins to be unstable, and therefore the regulator switches to service node controller to command it.

These nodes get the sensory data and deliver the control actions to the system through the communication middleware.

4.2 Case Study 2. Supervising control with local compensating

In this case the idea is to use the distributed computing availability to run a predictive control algorithm. This piece of control will provide future control actions that can be used by the local processor to feed actuators in the case of unexpected communications delays or missing data.

The predictive control algorithm is a Generalized Predictive Control (GPC). Next the developed strategy is explained.

4.2.1 Developed Strategy

The developed structure basically involves a distributed control system made by a service node, with a supervisor control GPC, and a light node. The service node is a system with wide capacity in computation and communication resources, whereas the light node is an embedded system with limited computation resources.

![Fig. 4. Process signals evolution when some control CAN messages are lost and the system controller is switching from the service node to the light node.](image1)

The light node takes the control

![Fig. 5. Evolution of the process signals when a control error is detected and the controller is changed from the light node to the service node.](image2)

The service node takes the control
In order to minimize these problems, in case of using the postulated control actions calculated by the GPC, the light node makes modifications in the propose control actions, considering the discrepancy between the output calculated trajectory and the real output is applied.

\[ e_y = \hat{y}(k + i|k) - y_{real}(k + i), \quad i = 1, ..., N \] (10)

If \( i < N_u - 1 \), then \( u = u(k + i|k) + K_{gain} \cdot e_y \) (11)

If \( i \geq N_u - 1 \), then \( u = u(k + N_u - 1|k) \) (12)

where \( \hat{y}(k + i|k) \) and \( u(k + i|k) \) are elements of the Y and U vector, respectively. \( K_{gain} \) is the local compensator gain that is due to determine of empirical method, studying the influence of error “\( e_y \)” in the final control action. The propose strategy is shown in figure 6.

Fig. 6. Distributed control structure.

The advantage of these systems is that service node can make the supervision of several light nodes.

Considering that the maximum horizon calculated by the GPC corresponds to sampling instant \( k+N_u \), it is necessary to design this horizon so that all the system dynamics is included, thus is case of missing all data makes sure that the output arrive at the reference.

In case of massive missing data, further the designed control horizon, it is necessary to apply a safe control strategy, that according to the controlled process it can consist of constant control action application or an emergency shutdown (Crespo et al., 2006).

Finally, it is necessary to consider that when recovering the communications between light and service node, the supervisor control GPC must known the control action applied during missing data and the real output, for recalculate the trajectories with the real data.

4.2.5 Implementation Example

The plant tested is an electronic process of second order with a stable overshoot response in open loop. The transfer function is:

\[ G_p(s) = \frac{6.818}{0.1021s^2 + 0.9588s + 7.818} \] (13)
The parameters of the GPC are:

\[ N_1=1, \ N_2=8, \ N_u=3, \ \gamma=0.1, \ T(z)=1 \] and \[ T_s=0.05 \text{ seconds} \]

Following the figure 6, a dsPIC microcontroller act as light node, it has been directly connected to process. As service node a XScale embedded computer has been used, it computes the GPC algorithm. The interconnection and data exchange between the nodes are carried out through the CAN communication bus (Coronel et al., 2005). The main characteristics of these nodes are described in (Martinez et al. 2007).

Basically, light node gets, via CAN, the control values computed by the GPC on service node, and apply them to the electronic process with its analogue outputs. The data values sent to light node are the present control action, two future predicted actions, and eight predicted output trajectories based on the GPC process model.

### 4.2.6 Results

The main advantage of this distributed system approach is its fault tolerance for sporadic error communication. As shown in the figure 8, when the control action is not received at the light node (data lost), it has to apply the predicted actions at the next sample times. If the communication errors are longer than control horizon, the light node has to apply the last control action value according to the predicted trajectory data. This is shown in the figures 8 and 9, where the data is lost during five sample periods (50 milliseconds).

Both figures are from the same experimental test, figure 8 shows the evolution of the process (system signal tries to follow the reference value), whereas figure 9 displays the control signal applied, the control value computed by the GPC and the difference between them.

![Fig 8. Process Evolution when some data is lost](image)

The higher difference between the applied control value and the GPC computed value occurs when the data is lost just within the transitory response (third second in figure 9). Although this is the worst situation for loosing data, the control in the light node overcomes with the process evolution as it is shown in figure 8. The control in the light node corrects the action values considering the discrepancy between the calculated output trajectory and the present output, minimizing the output error.

![Fig 9. Control Signal Applied, GPC Control Value computed and their discrepancy when data is lost](image)

5. CONCLUSIONS

In this paper a distributed control kernel model to complex control applications has been presented. Furthermore, from the implementation point of view, the proposed control model permits to perform a set of basic activities to ensure a safe operation of the system under control. A realization of a control algorithm GPC is described. The proposed distributed control scheme, through a combination of embedded systems, permits to ensure a safe and suitable operation of the system under control. This work is intended to be a proof of concept of some of the characteristics that are going to be implemented in the middleware kernel of the project KERTROL.

REFERENCES


