Distribution of neural-based discrete control algorithms applied to home automation with CAN

Bonastre A., Ors R., Capella J.V., Herrero J.
Department of Computer Engineering, Polytechnical University of Valencia

Distributed systems have been demonstrated as one of the best options when implementing industrial control systems due their simplicity and power.
Following this line, our group has proposed Rule Nets (RN) as a HLP over CAN for the implementation of Distributed Expert Systems (at 6th ICC), obtaining excellent results although RN lack of continuous control capabilities.
To provide these new features, in this article, the use of Neural Networks (NN) alongside of RN, as an application level over CAN is proposed, offering an intelligent and hierarchical environment of distributed control, where continuous control loops are being supervised by an expert system.
This technique is now being applied in the integral control of an automated house, where NN implement the control loops (heating /cooling, brightness, etc) and advanced features (voice, fingerprints and shape recognition, etc.) and over them RN supervise the operation of the system, under normal operation conditions and exceptional situations.

Introduction

Distributed systems offer several advantages when implementing industrial control systems, such as scalability, fault tolerance, simplicity and power. In this line, a new HLP over CAN for the implementation of Distributed Expert Systems was proposed [1]. It was based in the so-called Rule Nets (RN), obtaining excellent experimental results [7][8] although RN lack of continuous control capabilities.

New capabilities were required to allow the implementation of continuous control systems. Continuous and discrete control are not incompatible, but complementary, when dealing with complex systems. Combination of both techniques offers new possibilities, such as the implementation of hierarchical control systems. In this scheme, lower level perform continuous control loops while the upper level, based on discrete control techniques, takes decisions over the system, supervises the system function and diagnoses continuous control failures.

When dealing with the implementation of a continuous control system it was needed to decide the most accurate to the characteristics that the previous protocol had achieved, such as:

- To be able to control any system with the same benefits that the control systems already existent.
- Possibility that even non-expert users could design the control system, and even allowing the capacity for the self-learning of the system.
- Possibility to be executed in a distributed way
- Easiness when being transmitted through the net for their execution in generic nodes (possibly very different between them).

After an exhaustive study of different techniques of continuous control, Neural Networks [2][3](NN) have been selected because they meets all the previous conditions.

Indeed, NN not only completes the first condition since they can deal successfully with any control system, but, far beyond, they are able to control systems where no other techniques can be applied.

They also fulfill perfectly the second condition due to their learning characteristics [4], making possible that even non-expert users in the design of control systems can be capable, in a centralized way, to define the desired behavior of the system. NN are capable to learn in an automatic way thought the analysis of a group of samples that reflect the answers expected in real situations of execution. Even more, it is possible in the design phase the simulation of the system operation in front of hypothetical situations with the purpose of checking that it fulfills the desired specifications.

Since the NN are formed by perfectly detachable units (neurons), the distribution of these neurons in different nodes is not very difficult, so the results obtained by each neuron must be spread thought the interconnection network to be used as an input for any other neuron needing it.

Finally, it is possible to characterize a neural network as a group of neurons. All the neurons present a common structure, and they are easily adapted in function of a set of parameters. Therefore, it is possible to locate several generic neurons in the nodes and make a particularization of them in function of the desired control system by means of the transmission of these parameters.

The structure of a generic neuron can be observed in the figure 1. Figure 2 shows a NN where interrelations between neurons are appreciated.

To particularize a generic neuron to fit any one of the NN, the following parameters are required:

- *Position of the neuron* in the NN. To determine this, a **layer number** and an **order number** inside the layer will determine one neuron.
- *Number of neuron inputs*: how many inputs the neuron has.

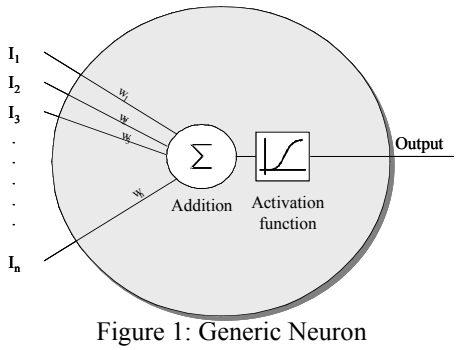

Figure 1: Generic Neuron

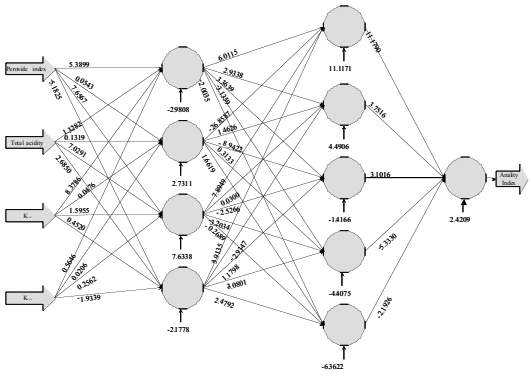

Figure 2: Sample neuronal network

- *Characteristic of the inputs*: The source **neuron** and the **weigh** of each input must be defined. The **weigh** can be explicitly transmitted, and the **neuron** connected with this input is identified by the position in the NN as shown above.
- *Activation function*: The number of activation functions is limited, so it has been proposed a function code that selects the desired one in a set of available functions.

With this parameterization, it is possible to transmit a NN through a distributed system in a simple and efficient way.

NN execution needs some synchronization mechanisms. Several synchronism techniques have been evaluated, and finally the following one has been selected.

The system works in a event-oriented way. Any input neuron (those whose inputs correspond to external inputs) can start the operation of the NN. When a Start condition arises, the input neuron send an NN START message, making that all input neurons read all their input values and evaluate their

outputs. These outputs must be send through the CAN network, while they are been used as input values of subsequent neurons. Any neuron will calculate and spread its output as soon as it gets all its inputs. Finally, output neurons will not apply its outputs until no remaining value would exist. This is accomplished by means of a *fictitious neuron. Fictitious neuron*, uses as input values the outputs of the system, and activates the NN for a new operation, allowing output to be applied. Synchronous operation is granted by this procedure.

The fault tolerant characteristics of the system must be highlighted. When being a hierarchical system, the superior level takes charge from the topic of fault tolerance. When a failure in the system is detected, the upper level takes the control of the system, evaluating the failure and deducing from the characteristics of it the correction actions that should be carried out for the reconfiguration of the system (operation in degraded way) or those guided to drive the system into a safe failure state.

CAN network [5] has been selected for the execution of the Distributed Neural Network because of its particular characteristics. First of all, CAN is a diffusion network, providing that all neurons that need a value produced by another neuron get the value with only one message. Even more, these messages are labeled (in the identifier field) with the neuron that produced it, so avoiding any overload. Also, non-destructive contention allows a limited response time. Is it also possible to fix the priority of any messages, guaranteeing that values proceeding from latter neurons will be sent before those from earlier ones. Finally, CAN offers several advantages, such as low cost, fault tolerance, great variety of products, etc.

Communication protocol

The new protocol ICCAL (Intelligent Control CAN Application Layer) is an extension of the protocol DESCAL [1], updated in [6]. A new level has been included for the implementation of continuous control systems by means of neural networks. System architecture is shown in figure 3.

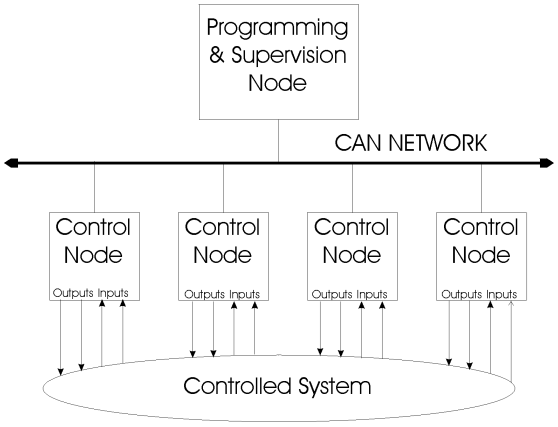

Figure 3: System architecture

The PSN (*Programming and Supervision Node*), which will also support the specification functions, training and distribution of the neural networks, and CN´s (*Control Nodes*) also incorporate news features in order to NN execution. The protocol use addresses in CAN extended format (CAN 2.0B). Therefore, the new characteristics of the protocol only will be available in CN implemented with extended CAN devices, although passive CAN extended devices could belong to the system, but restricted to Rule Nets execution.

Under these circumstances, new protocol is fully compatible with the old DESCAL protocol.

Protocol messages

The protocol DESCAL used four types of messages. ICCAL contemplate two new messages and adds new control messages.

Control messages: To control the NN operation, the same messages that appeared in DESCAL have been used. It is also necessary to add a GO message, that applies the outputs of the NN and activates all neurons for the next operation.

To distinguish RN and NN commands, those related to the NN will be sent in extended CAN messages.

Identification messages: Several updates have been introduced from DESCAL. After the beginning of the identification phase, each node transmits a message where it announces its presence in the network, informing of its capacities, followed by the transmission of a list of its inputs and

outputs. In this identification message a bit has been added (NNC) to indicate that the node offers the possibility to execute NN.

Load messages: They make possible the transmission of the variables and matrix alongside the CAN network. To differentiate the messages of the RN matrix transmission and those of the NN, the use of CAN extended messages for latter ones has been selected. In this way, only extended CAN CNs will be affected by the new messages.

Updating Variable messages: they allow the diffusion of the new values in the RN and in the NN. To differentiate both cases, the distinction between standard CAN and extended CAN has been used, Extended CAN messages being reserved to indicate the change of values referred to the NN.

Stages of the protocol

The protocol is structured in three phases or states:

Initialization

During this phase all the available CN's in the system are identified, informing to the PSN of their characteristics. The new version adds at the beginning of this phase, the transmission of an identification message for each CN, in which is indicated, alongside other characteristics, if it is able to execute NN.

3	8	7	1	8	8
001	*Node*	*0000000*	*NNC*	*Ports*	*Speed*
Identifier	Data				

Where:

Node: Number of node that is identifying itself.

NNC (Neural Network Capability): This bit must be set to "1" to indicate that this node is able to execute neural networks.

Ports: This field indicates how many input and output ports are available at the CN.

Speed: it indicates what transmission speeds supports the node. The assignment of each bit is the following one:

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	*0*	*0*	*0*	*1 Mbps*	*500 Kbps*	*250 Kbps*	*125 Kbps*

Since all nodes should contemplate at least the speed of 125 Kbps, the minimum value of this field will be 1. This field is used as a guide for the selection of the speed of the system.

The rest of the identification messages related to the input and output ports, are numbered consecutively by means of the field *count* of 7 bits, existent in the first byte of the field of data.

Design & Distribution

During the design stage, the user introduces the Rule Net (RN) and Neural Net (NN) in the PSN, assigning the logical variables from the Rule Net and Neural Net to the physical inputs and outputs and even creating the necessary internal variables.

Also the powerful graphical environment of the PSN will allow to the user train the neural network given a collection of samples.

The distribution of the neuronal net is user-guided, so that he can select the most accurate node for the execution of each neuron. Several automatic distribution techniques in function of the requirements of the system and the user are being studied.

As first step in the particularization of the generic neurons to the desired control system, it is necessary the selection of the activation function and the data format used in the representation of the numeric values weight and results). To achieve this goal, the following message is send to each neuron:

3	8	16	2	8	8
000	Node	NI	00	AFC	NRSC
Identifier					Data

Where:

Node: Node where the neuron resides. Thanks to the presence of this field it is simple the definition of reception masks in the CAN controllers.

Neuron Identifier (NI): it allows the identification of a neuron. The first five bits indicate their level and the last eleven indicate the ordinal number inside the level.

Activation Function Code (AFC): This field selects among the set of activation functions

which one should be used in the calculations of the neuron.

Numerical Representation System Code (NRSC): The code of the numeric representation system that will be used to transmit the results of each neuron is indicated in this field.

For every neuron input, the distribution will require one of the following messages:

3	8	16	2	16	8	40
110	*Node*	*NI*	*00*	*INI*	*NRSC*	*Weight*
Identifier					Data	

Where the fields **node**, **NI** and **NRSC** correspond with those previously described. Also, other two fields exist:

Input Neuron Identifier (INI): This field reflects the identifier of the neuron that will send the value, in function of their level and number inside the level, that will be contained in the identifier of the CAN message that diffuses the value through the network.

Weight: Weight associated to the corresponding input.

Execution

During this stage each node executes its Rule and Neural nets, communicating through the net the changes produced in the global variables and the results of each neuron.

We will distinguish among **input neurons**, those that their entrances correspond with physical input of the system, **output neurons,** those that their exit correspond with one physical output of the system, and the rest of neurons will be **intermediate neurons**, because their inputs come from other neurons and their outputs are needed by other neurons.

Initially all the neurons are active in state WAITING (see figure 4) for its input data. The process will be initiated by an input neuron through a start condition, diffusing by means of a START message the beginning of the computing process to the other input neurons. The START message has the following format:

3	26	8
000	Broadcast	START
Identifier		Data

Once initiated, when all inputs are available the neuron perform their calculations and diffuses its output by the network and changes its state to DISABLED (see figure 5). The obtained values are sent through messages with the following format:

3	16	11	<= 64
111	NI	*1010101010*	Result
Identifier			Data

Where the field **NI** correspond with those previously described. And the other field is:

Result: Value calculated by the neuron, according to the established representation

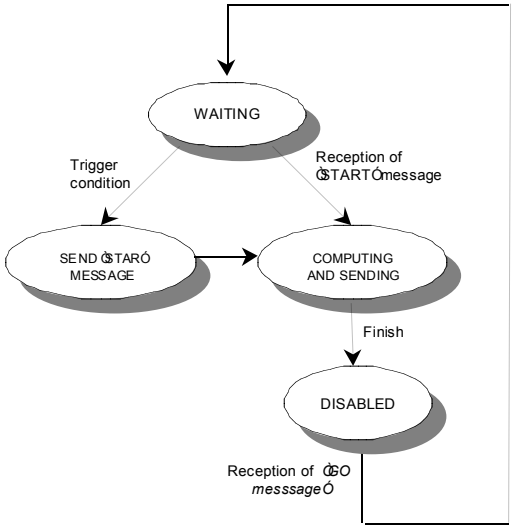

Figure 4: Input neuron state diagram

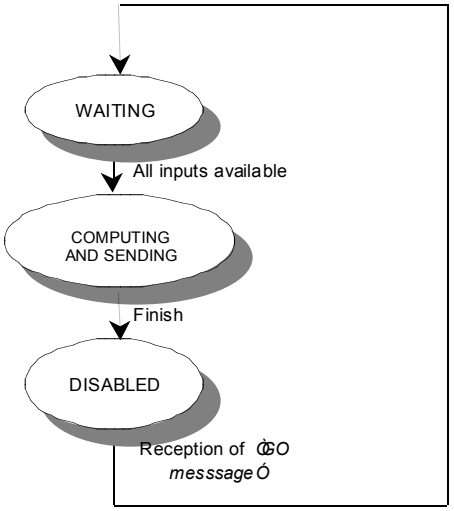

Figure 5: Intermediate neuron state diagram

system.

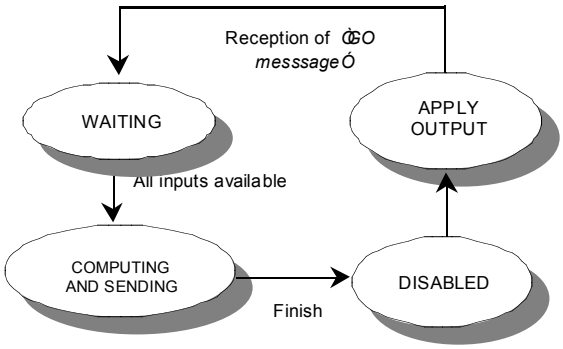

Figure 6: Output neuron state diagram

Finally, when output neurons finish their operations, they will send its results to a so-called *fictitious neuron*. This neuron, which is added automatically for the system, will receive the values of all the outputs. When all the values are available, the fictitious neuron sends a GO message that forces the output neurons to apply their results to the physical output (see figure 6). This procedure guarantees that all the outputs will be applied at the same time.
Additionally all neurons in DISABLED status will pass to WAITING, so a new computing process could start when an start condition rises
The format of the GO message is:

3	26	8
000	Broadcast	GO
Identifier		Data

Test application: Home integral control by means of a Neural Network
As experimental environment for the test of the new protocol capabilities, the integral control of a house is being implemented in a laboratory scale model, very interesting partial results being obtained at the moment.
The system structure can be seen in figure 7. The system presents a hierarchical structure where upper level deals with the discrete control, the supervision of lower layers and must perform the adequate actions in case of failure. Middle layer takes into account the continuos control loops, including hardware supervision. Finally, lower layer consists of the system to control and a group of sensors and actuators that carry on the actions defined by upper layers.

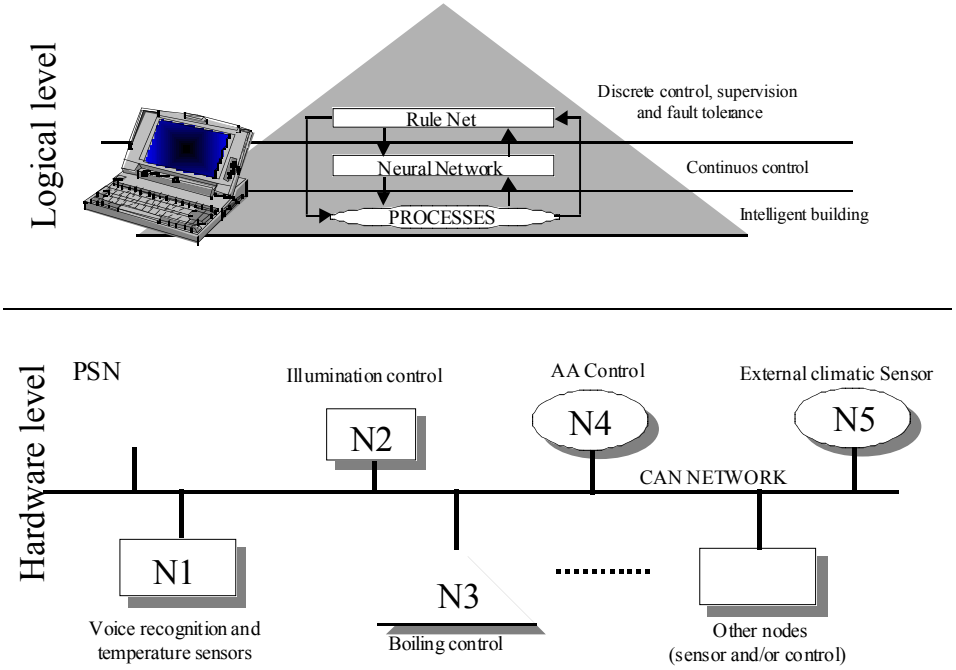

Figure 7: System structure

Physically, the system consists on at least six nodes, an PSN and five CN,
interconnected by means of a CAN network
Practical evaluation of proposed protocol has been performed by means of a test system, described below. There has been built an illumination control loop based on two CNs; first one (N1) is implemented by means of a PC with a USB-CAN communication device, that holds a Neural Network able to recognize simple voice commands [9]. This NN actualizes the value of some variables of the Rule Net, that controls the illumination. This RN is located in both N1 and N2 nodes. N2 is implemented by means of a CANary processor, able to handle all the illumination devices at the whole house. The collaboration between both networks, NN and RN, has proved to be very effective and successfully resolved. The diagram of interrelations can be seen in the figure 8. Voice commands are interpreted by the NN, indicating the wishes of the user. The presence is also detected by means of adequate sensors and optical barriers at doors, but these inputs are handled by the RN. As a result of all these inputs, RN updates the control light signals.
By the other way, to verify the distributed

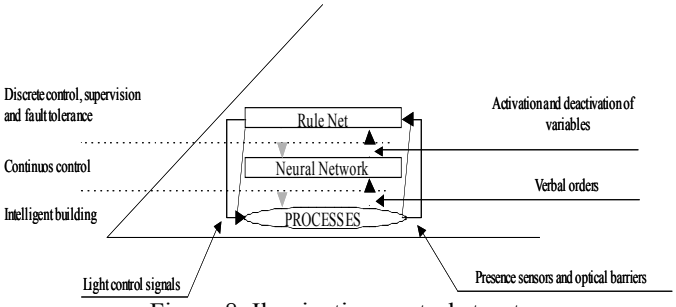

Figure 8: Ilumination control structure

functioning of the NN, another control loop that controls the habitability conditions of the house, has been established. This system is composed by four nodes. Node N3 is in charge of the boiler, and it is implemented by means of a CANary processor with a digital output. Node N4 controls the conditioned air of the room, and it is also implemented with a CANary processor, using a digital output (ON/OFF) and an another that, by means of a triack, controls the fan speed. Finally, node N5 is located outside the house. Node N1 is also needed in this subsystem, with a temperature sensor inside the room. Our NN uses the indoor and outdoor
temperatures as inputs, and is able to indicate the RN if the utilization of any temperature control systems (boiler and air conditioner) is necessary. If AA is needed, NN controls the analog fan speed, as shown

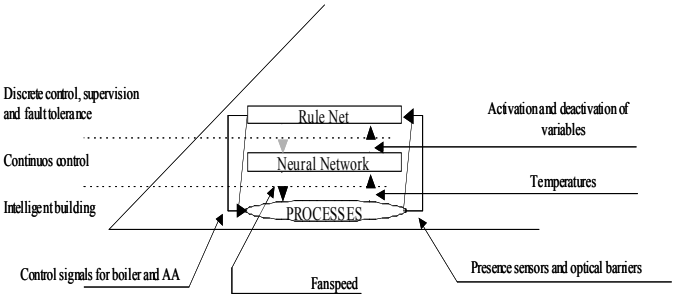

Figure 9: Air Conditioning structure

in figure 9.

After a design process to determine the best topology of the NN, and the appropriate training, the distribution of the NN was performed by the user. The protocol performance was completely successful. Some design and training matters of the NN should need further revision.

Finally, both control systems (RN and NN) work together in a perfect manner, guaranteeing the correct function of the proposed objectives. Nowadays, new specifications are being introduced, such as temperature control by means of voice commands and self-learning.

Conclusions and future work

Given the excellent results obtained, two new approaches are being applied to the system. One of them consists on the implementation of a more powerful PSN software that integrates all options and tools (design, training, distribution, etc.), and

could take carry on an automatic distribution of NN. This distribution algorithms should take into account the user restrictions (minimum number of messages on the network, maximize parallelism, fault tolerance by means of redundancy, etc.)
Other interesting question to study is the response time of the proposed system, in order to its application to Real Time control systems, and the impact of fault tolerance in this situations.
Finally, this methodology is being applied to other control systems, such as the integral control of a Diesel engine, substituting current strategies (cartographic methods) with NN based ones. Over them an expert system is responsible of the supervision and control of normal and anomalous operation of the system.

References

[1] "A CAN Application Layer Protocol for the implementation of Distributed Expert Systems" Bonastre, A., Ors, R, Peris, M. Proceedings 6th International CAN Conference. Noviembre 1999.
[2] S. Y. Kung, Digital Neural Networks, Prentice Hall, 1993.
[3] S. Pal, S. Mitra, Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing, John Wiley & Sons, New York, 1999.
[4] M. Anthony, P. Bartlett, Neural Network Learning: Theoretical Foundations, Cambridge University Press, 1999.
[5] "CAN specification, Version 2.0 Robert Bosch GmbH, Stuttgart, 1991.
[6] " Industrial Local Area Network: A new architecture for control distributed systems." Ph. D. on computer science. Bonastre, A., 2001.
[7] "Distributed expert system for the monitoring and control of chemical processes" Peris, M., Bonastre, A., Ors, R. Laboratory robotics and automation Ed. Wiley & Sons, Julio 1998.
[8] "Distributed Expert Systems as a new tool in Analytical Chemistry" Bonastre, A., Ors, R, Peris, M., Trends in Analytical Chemistry vol. 19 Abril 2000.
[9] “Geometric pattern recognition techniques for acoustic-phonetic decoding of Spanish continuous speech” Castro, M., Prat, F., Aibar, P., Casacuberta, F. EUROSPEECH’95, September 1995.

Contact Info
Bonastre Pina, Alberto
Polytechnic University of Valencia
Camino de Vera, 2
46071 Valencia, Spain
Phone: +34 96 387 9703 / 387 7577
Fax: +34 96 387 7579
Email: bonastre@disca.upv.es
Homepage: http://www.disca.upv.es/gstf/