AN ADAPTIVE FAILURE DETECTION APPROACH FOR REAL-TIME DISTRIBUTED CONTROL SYSTEMS OVER SHARED ETHERNET

Alírio dos Santos Sá and Raimundo José de Araújo Macêdo
Pos-Graduation Program on Mechatronics[1] - Distributed Systems Laboratoty (LaSiD)
Federal University of Bahia, Campus de Ondina, 40170-110, Salvador-BA, Brazil.
{aliriosa, macedo}@ufba.br

Abstract. *Integration among sub-systems, easy interconnection among devices, reliability, and reduction of operational cost are some of the requirements demanded by modern control systems. Thus, the automation and control industry has used off-the-shelf components such as the Ethernet to reduce costs and increase interoperability. However, the non-determinism nature of such networks brings difficulties for time-critical control applications, especially when fault-tolerant mechanisms (such as failure detectors) are employed to guarantee continuous operation. Following this context, and considering a distributed control system built over an Ethernet network, this paper presents an implementation of an adaptive failure detector based on Artificial Neural Networks. By means of simulation it is shown that the presented approach performs well compared with conventional existing solutions. Additionally, this paper gives evidences that the overhead caused for the use of such detectors does not interfere with the performance of conventional control algorithms, such as PID, in a dedicated network.*

Keywords: *Real-Time Distributed Control Systems, Networked Control Systems, Fault-Tolerance, Failure Detectors, Quality of Service of Failure Detection*

1. Introduction

The advance of computer networks has led the automation industry to adopt standards such as the shared Ethernet in the implementation of distributed control solutions. Such an approach aims at facilitating the integration between systems, easiness of interconnection between devices, reliability, and reduction of the operational costs. However, despite all benefits, the network time delays can influence the control performance and can lead to a more complex system design[Lian et al., 2002]. The communication time delay is related not only with the network *bandwidth*, but also with the size of the messages, number of collisions, message loss probability, etc. [Tanenbaum, 2003].

Further, some distributed control applications require fault-tolerant mechanisms that allow for the continuous operation, taking into account the temporal constraints of control applications. For example, failure detection is a basic service for fault-tolerant mechanisms, either to activate recovery procedures or to allow the reconfiguration of the system [Jalote, 1994]. However, the algorithms related to the fault-tolerant mechanisms in distributed systems impose additional message exchange to guarantee the required *liveness* and *safety* properties [Lynch, 1996]. This message exchange makes the system project still more difficult, since the message traffic increases can result in non-deterministic delays in the *Ethernet*. Thus, it is important to consider the non-deterministic of such networks when designing failure detection mechanisms, so that the detector can be adaptive to the current network load.

Following this context, this paper presents an implementation of an adaptive failure detector based on Artificial Neural Networks for distributed control systems over a *shared-bus-Ethernet* (IEEE 802.3 CSMA/CD) network. By means of simulation, it is shown that the presented approach performs well compared with conventional existing solutions. Additionally, this paper gives evidences that the overhead caused for the use of such detectors does not interfere with the performance of conventional control algorithms, such as PID, in a dedicated *shared-bus-Ethernet*.

The rest of this paper is organized as follows. In section 2.it is discussed related work. In section 3.it is discussed failure detection approaches for conventional distributed systems. The failure detection approach based on Artificial Neural Networks is introduced in section 4. In section 5.it is presented the performance data collected from a series of simulations. Finally, in section 6.it is presented our conclusions.

2. Related Work

In [Lian et al., 2001] it is discussed the performance of shared *Ethernet*, *ControlNet* and *DeviceNet*, regarding the implementation of distributed control systems. In their work, they studied the temporal aspects in the communication between devices of the control systems and verified how these components are affected by network characteristics, such as propagation time on the network medium, maximum and minimum data sizes, medium access protocols, expected time delays etc. In [Lian et al., 2002] it is discussed the Quality of Performance (QoP) of distributed control systems on the network protocols evaluated in [Lian et al., 2001]. They analyzed aspects such as end-to-end transmission time delays, and in [Lian et al., 2002] they demonstrated the shortest and longest sampling times that can be used by distributed

[1]Computer Science and Mechanical Engineering Departments

control systems in order to guarantee acceptable QoP. These works, however, do not consider fault-tolerance mechanisms necessary for some distributed control applications.

[Hermant and Lann, 2002] discusses how to maximize the covering of safety, liveness, and timely properties of critical real-time systems using conventional algorithms for asynchronous distributed systems models. The authors demonstrate the feasibility of their proposal for fault-tolerant mechanisms in a real-time distributed system using failure detectors and *Ethernet*/*CSMA-DCR*, a determinist variant of the original *Ethernet*/*CSMA-CD* protocol.

Adaptive failure detection mechanisms with QoS have been discussed in some works, such in [Chen et al., 2002], [Bertier et al., 2002], [Macêdo and Lima, 2004], and [Nunes and Jansch-pôrto, 2004]. These approaches, however, deal with failure detection for conventional distributed systems, not discussing the feasibility of the use of such approaches in distributed control applications.

3. Conventional Adaptive Failure Detection Approaches

In order to analyze the efficiency of failure detection, we use in this paper the QoS metrics proposed in [Chen et al., 2002]. These metrics evaluate the speed in fault detection and the ability of the detector in avoiding mistakes. The main metrics are summarized in the following:

- *Detection Time* (T_D) is the time interval between the instant when the component fails and the instant when such a component is suspected by the detector;
- *Mistake Duration* (T_M) measures the time it takes for the failure detector to correct a mistake;
- *Mistake Recurrence Time* (T_{MR}) is the time between two consecutive mistakes;

All approaches discussed here use the *heartbeat* monitoring model, like in [Felber, 1998, Chen et al., 2002]. In this model, it is considered the existence of two components, p and q. The component q has an associated failure detector module and monitors the $crash$ of component p. Every Δ^i time period, p sends to q a message, sequentially timestamped and denoted by *heartbeat*, notifying that it is functioning correctly. For each *heartbeat* (denoted for m_k^{hb}) received, q computes the time interval (Δ^{to}) necessary for the arrival of the next *heartbeat* (m_{k+1}^{hb}). If m_{k+1}^{hb} does not arrive within Δ^{to}, q puts p in its suspected list. In case q receives a *heartbeat* with timestamp equal or larger than the timestamp of the expected *heartbeat*, q removes p from its suspected list.

The component p sends m^{hb} messages in instants denoted by σ, thus: m_k^{hb} is sent in σ_k, m_{k+1}^{hb} is sent in σ_{k+1}, m_{k+2}^{hb} is sent in σ_{k+2} etc. For any two consecutive instants σ_k and σ_{k+1}, $\sigma_{k+1} - \sigma_k = \Delta^i$. The message arrival time of m^{hb} is denoted by A. In other words: m_k^{hb} arrives in A_k, m_{k+1}^{hb} arrives in A_{k+1}, m_{k+2}^{hb} arrives in A_{k+2} and so on. If $delay_k$ is the travel time of m_k^{hb}, so:

$$A_k = \sigma_k + delay_k \tag{1}$$

Hence, each *heartbeat* is obsolete at least $delay$ time units. In an environment where there is no variation in time delays nor there are message losses in the communication subsystem, the time interval in which q can start to suspect of p crash with safety cannot be less than:

$$T_D^{min} = delay + \Delta^i \tag{2}$$

T_D^{min} is the minimum detection time. If variations in time delays are considered, we have:

$$A_{k+1} - A_k = \Delta^i + (delay_{k+1} - delay_k) \tag{3}$$

When the arrival times of messages are unknown and there are no synchronized clocks, the failure detector does not carry out accurate estimation of detection time. For the estimation EA_k carried out by the detector for the *heartbeat* arrival time m_k^{hb}, the more close EA_k will be of A_k, the smaller will be the detention time. The relation $EA_k \geq A_k$ must be satisfied for that detector to avoid false suspicions. Thus, the relation between EA and A is a performance indication of the detector in terms of detection time. When extra variations of delays can provoke sub-estimates, a safety margin (α) is used to compensate possible extra network delays [Jacobson, 1988, Chen et al., 2002]. Then, the new time estimation FP (*Freshness Point* [Chen et al., 2002]) for next *heartbeat* arrival is defined by:

$$FP_{k+1} = EA_{k+1} + \alpha_{k+1} \tag{4}$$

The adaptability of failure detector consists of adjusting EA and, possibly, the safety margin α. Thus, when m_k^{hb} is received in A_k, the more accurate the estimate EA_{k+1} and α_{k+1}, the more close FP_{k+1} will be of A_{k+1}.

3.1 The Jacobson's Algorithm

The Jacobson Algorithm [Jacobson, 1988] is used in TCP networks to predict the instant in which a message must be retransmitted, preventing, thus, unnecessary retransmissions. Such an algorithm is briefly described in the following. Being rtt_k^M and rtt_k^C, respectively, the measured delay and estimated delay in instant k, the algorithm computes rtt_{k+1}^C using rtt_k^C and the importance μ associated to the error ($error_k$) between rtt_k^C and rtt_k^M. Thus:

$$error_k \quad = \quad rtt_k^M - rtt_k^C \tag{5}$$

and

$$rtt_{k+1}^C \quad = \quad rtt_k^C + \mu.error_k \tag{6}$$

Considering the existence of extra variations in network delay, the new estimate for delay (Δ_{k+1}^{to}) in message arrival is:

$$\Delta_{k+1}^{to} = \beta.rtt_{k+1}^C - \phi.var_{k+1} \tag{7}$$

Where ϕ and β are the confidence in the delay variation and in the estimated delay, respectively. var_k represents the variation in the measured network delay at instant k, and can be calculated by:

$$var_{k+1} = var_k - \mu.\left(|error_k| - var_k\right) \tag{8}$$

Finally, it is presumed that the next message will arrive in:

$$FP_{k+1} = A_k + \Delta_{k+1}^{to} \tag{9}$$

3.2 The Bertier, Marin and Sens' Algorithm

In [Bertier et al., 2002] it is proposed an *Adaptive Failure Detector* (AFD), implemented in two layers in such a way that the detection layer (lower layer) can provide failure detectors based on the quality of service metrics presented by [Chen et al., 2002], while the adaptation layer (upper layer) adjusts the detection QoS of the lower layer to conform with the requirements of applications. The message arrival prediction realized in the lower layer is based on an estimation proposed in [Chen et al., 2002], where:

$$EA_{k+1} = EA_k + \frac{1}{n}\left(A_k - A_{k-n-1}\right) \tag{10}$$

During the detection initialization phase, that is, for the n initial messages, the detector is parameterized as follows [Bertier et al., 2002]:

$$U_{k+1} = \frac{A_k}{k+1} * \frac{k * U_k}{k+1} \tag{11}$$

and

$$U_1 = A_0 \tag{12}$$

Thus, it is computed by:

$$EA_{k+1} = U_{k+1} + \frac{k+1}{2} * \Delta^i \tag{13}$$

The safety margin α estimation uses the Jacobson's algorithm. Thus, the authors calculate, respectively, the error and safety margin by:

$$error_k = A_k - EA_k - delay_k \tag{14}$$

and

$$\alpha_{k+1} = \beta.delay_{k+1} - \phi.var_{k+1} \tag{15}$$

3.3 The Mâcedo and Ramon's Detector

In [Macêdo and Lima, 2004] is presented a failure detector based on Artificial Neural Networks, named ANNFD. This detector uses as input parameters variables collected by the Simple Network Management Protocol (SMNP) that characterize the network traffic at each time instant.

After training the neural network, it must compute the message arrival time estimation EA_{k+1}, which is utilized to define the freshness point of m_{k+1}^{hb}:

$$FP_{k+1} = EA_{k+1} + \alpha \tag{16}$$

where α is estimated by experimental analyses.

The authors demonstrate that the ANNFD presents a better performance in comparison with the AFD in load variation conditions of the communication system. In moderated conditions, however, the AFD presents better performance.

4. The Failure Detection Approach for Distributed Control Systems, Based on Neural Networks

In this section, we present the failure detection approach based on an Artificial Neural Network proposed to a real-time distributed control system. We first present the system model utilized and then the failure detection approach developed.

4.1 The System Model

The system model considers the existence of a simple control system with three main devices: a sensor, an actuator, and a controller. Each device has a real-time operating system and is connected to a shared-bus-Ethernet. This scenario composes a simple real-time distributed control system (see figure 1).

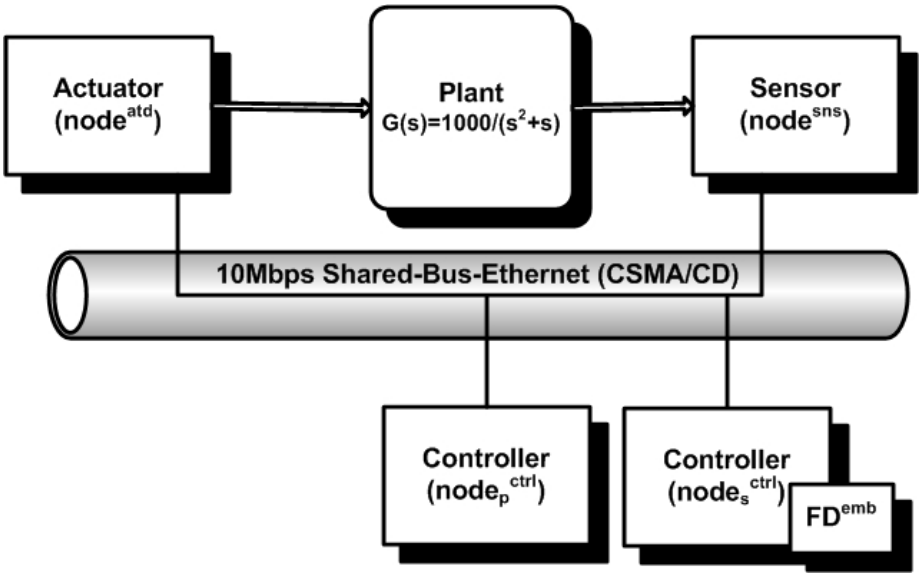

Figure 1. Distributed control system with failure detector

There is a periodic task (τ^{sns}) for data acquisition associated to the sensor device. Every collected sample by τ^{sns} is sent to the controller device. In the controller, a control task (τ^{ctrl}) is activated at each received message from the sensor. τ^{ctrl} executes a simple Proportional, Integral and Derivative (PID) control algorithm (see [Ogata, 1990]) and sends the control information to the actuator device. The task τ^{atd} is activated upon the reception of a control message. Finally, the actuator task acts over the plant. The scheduling policy of the tasks is based on fix priorities.

The controlled object is a single DC-Servo, like in [Henriksson and Cervin, 2003], described by the continuous-time transfer function:

$$G(s) = \frac{1000}{s(s+1)} \tag{17}$$

We assume that the controller device may crash, thus, this device is replicated to tolerate a single failure. One controller is said the primary controller and the other one, the secondary controller. The primary controller receives messages from the sensor, executes the algorithm for state consistent with secondary controller device, and sends a control action to the

actuator device. There is a failure detector associated with the secondary controller, whose aim is to detect the crash of the primary controller. If the primary controller crashes, the secondary controller takes over to guarantee the continuity of the plant operation.

4.2 The Artificial Neural Network Implementation

The artificial neural network implemented is a Feed Forward Multi-Layer Perceptron (FF-MLP) [Haykin, 1994] with four layers: three neurons in the input layer, one neuron in output layer, and two hidden layers with thirty and ten neurons, respectively. In each neuron it is utilized the hyperbolic tangent sigmoid transfer function [Haykin, 1994]:

$$tanh(\frac{x}{2}) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{18}$$

The training of neural network uses the resilient propagation algorithm proposed by [Riedmiller and Braun, 1993]. This algorithm realizes the update of synaptic weights using an adaptive learning rate, as in equation 19, given below.

$$\Delta_{ij}^{k} = \begin{cases} \eta^{+} * \Delta_{ij}^{k-1} & , \quad if \ \frac{\partial E^{k-1}}{\partial w_{ij}} * \frac{\partial E^{k}}{\partial w_{ij}} > 0 \\ \eta^{-} * \Delta_{ij}^{k-1} & , \quad if \ \frac{\partial E^{k-1}}{\partial w_{ij}} * \frac{\partial E^{k}}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{k-1} & , \quad if \ \frac{\partial E^{k-1}}{\partial w_{ij}} * \frac{\partial E^{k}}{\partial w_{ij}} = 0 \end{cases} \tag{19}$$

where:

- w_{ij} is the synaptic weight of the neuron i to neuron j;
- $\frac{\partial E}{\partial w_{ij}}$ is the partial derivative error;
- Δ_{ij} is the synaptic weights correction factor w_{ij};
- η is correction factor of Δ_{ij}, being that η^{+} is utilized when the partial derivative w_{ij} is positive, while that η^{-} is used when partial derivative is negative, η^{+} and η^{-} must satisfy the following relation $0 < \eta^{-} < 1 < \eta^{+}$;

The rationale behind the equation 19 is the following. When the partial derivative of w_{ij} changes between negative and positive, it means that the weight updating was too large, then the value of Δ_{ij} is decreased by η^{-}. If the partial derivative keeps its sign, Δ_{ij} must be lightly increased by η^{+}. This procedure guarantees a fast convergence.

In the training phase, the Artificial Neural Network (ANN) was utilized with the following initial configurations:

- synaptic weights correction factor $\Delta_{ij} = 0.1$;
- $\eta^{-} = 0.5$ and $\eta^{+} = 1.2$;
- training epochs = $5 * 10^{4}$;
- minimal error = 0.0;
- minimal gradient = 10^{-12}

The input variables for the neural network are:

- *delay between arrivals*, i.e., the last delay observed between two consecutive *heartbeats* ($delay_k = A_k - A_{k-1}$);
- *variation of delays between arrivals (*$delay_k - delay_{k-1}$*)*;
- *heartbeat rate*(Δ^{i});

Using these variables, the ANN computes the estimated value for the interval (Ω) between consecutives *heartbeats*. Thus, the estimated arrival time for m_{k+1}^{hb} is (see figure 2):

$$EA_{k+1} = EA_k + \Omega_{k+1} \tag{20}$$

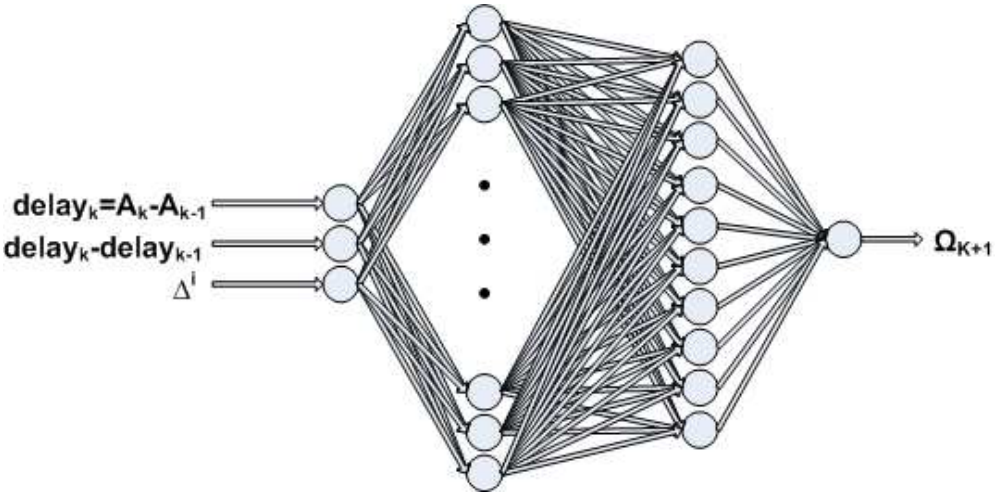

Figure 2. Artificial Neural Network Predictor

Table 1. Especification of system tasks

Task	Device	Activation type	Event	D	T
Control (τ_{ctr})	Controller	Sporadic	Reception of message sent by τ_{sns}	$6ms$	-
Aquisition (τ_{snr})	Sensor	Periodic	-	$10ms$	$10ms$
Actuation (τ_{atd})	Actuator	Sporadic	Reception of message sent by τ_{ctrl}	$4ms$	-

5. Simulations

We evaluated our adaptive failure detector using the TrueTime, version 1.13 [Henriksson and Cervin, 2003], a *Toolbox* for real-time distributed control system simulation available in the Simulink/Matlab Tool [The Mathworks, 2004]. In this simulation tool, it is possible to model and simulate real-time environments with different operating system scheduling policies, varied computer network protocols, and distinct models of task activation. For our experiments, we setup in TrueTime a 10Mbps shared bus Ethernet/CSMA-CD with data frames with 64 bytes. During the simulations, the tasks were configured as presented in table 1. In this table D is the deadline and T is the task period.

From the experiments we analyzed the failure detectors performance using the QoS metrics (T_D, T_M, and T_{MR}). Every metric was evaluated in terms of the mean value (T_D^{mean}, T_M^{mean} and T_{MR}^{mean}) and standard deviation (T_D^{std}, T_M^{std} and T_{MR}^{std}). Besides the QoS metrics, we also analyzed the number of mistakes (N^{fs}) made by the detector. The analyses were realized in environments where the network is utilized only to transfer data between the control system devices. We analyzed the [Jacobson, 1988] and [Bertier et al., 2002] algorithms, and these algorithms were compared with the neural network approach suggested in this paper.

In the experiments, we set $\Delta^i = \{50ms, 100ms, 150ms, 200ms, 250ms, 500ms\}$, $\alpha_0 = 0ms$ and the number of messages $N = 520$ for each fixed Δ^i(where 52 messages were used in initialization of the algorithms). In [Bertier et al., 2002] and [Jacobson, 1988] algorithms, it was utilized $\beta = 1$, $\phi = 2$ and $\mu = 0, 1$, as in their original proposals.

The table 2 shows that the *ANN* algorithm possesses a better performance in all evaluated metrics. The algorithm of *Jacobson* presented a bad precision, realizing much more false suspicions. However, The Jacobson and *Bertier* algorithms corrected their suspicions very quickly (≈ 0.00). The algorithm of *Bertier* presented the worst detention times, however, had a better performance when compared with the algorithm of *Jacobson* in terms of T_{MR}^{mean} and N^{fs}. The large values of T_D^{mean} presented by the algorithm of *Bertier* in the experiments was due the estimates produced during the detector initialization.

Although, the Neural Network predictor obtained better performance, its computation time is significantly larger than the others ($7.40ms$). However, such a computational time does not compromise typical control applications as we show below.

We measured the impact in the control by calculating the extra time delay d^{atd} incurred to control actuation with the usage of the failure detector. Thus, given t_k and t_k^{fd}, the message arrival times in the actuator without and with the detector, respectively, we calculated :

$$d^{atd} = \frac{1}{N}.\sum_{N}^{k=1}(\left|t_k^{fd} - t_k\right|) \tag{21}$$

Table 2. Results in milliseconds for QoS metrics obtained in realized experiments

Δ^i	$A_{k+1} - A_k$		Detector	T_D^{mean}	T_D^{std}	T_{MR}^{mean}	T_{MR}^{std}	T_M^{mean}	T_M^{std}	N^{fs}
	$mean$	std								
50.00	50.00	0.00	Jacobson	50.06	0.25	370.00	415.33	0.00	0.00	26
			Bertier	80.06	107.11	-	-	-	-	1
			ANN	50.01	0.00	-	-	-	-	0
100.00	100.00	0.00	Jacobson	100.11	0.50	857.14	1075.44	0.00	0.00	22
			Bertier	160.51	215.67	1040.00	2101.90	-	-	6
			ANN	100.00	0.00	-	-	-	-	0
150.00	150.00	0.00	Jacobson	150.17	0.75	1810.00	2064.81	0.00	0.00	16
			Bertier	240.97	324.25	150.00	0.00	0.00	0.00	7
			ANN	150.00	0.00	-	-	-	-	0
200.00	200.00	0.00	Jacobson	200.23	1.00	1575.00	2304.86	0.00	0.00	25
			Bertier	321.42	432.82	-	-	-	-	1
			ANN	200.00	0.00	-	-	-	-	0
250.00	250.00	0.00	Jacobson	250.28	1.25	-	-	-	-	1
			Bertier	401.88	541.39	-	-	-	-	1
			ANN	250.00	0.00	-	-	-	-	0
500.00	500.00	0.00	Jacobson	500.56	2.50	51500.00	66468.00	0.00	0.00	3
			Bertier	804.15	1084.26	-	-	-	-	1
			ANN	500.01	0.00	-	-	-	-	0

By analyzing d^{atd} (table 3), we observe that the detection mechanism in this environment does not impact on the performance of the control actuation, as the additional time delay in the actuation is insignificant and grows lightly with the decrement of Δ^i.

Table 3. Results in milliseconds of control actuation

Δ^i	50.00	100.00	150.00	200.00	250.00	500.00
$delay^{atd}$	$5.68 * 10^{-13}$	$2.87 * 10^{-13}$	$1.82 * 10^{-13}$	$1.50 * 10^{-13}$	$1.22 * 10^{-13}$	$6.79 * 10^{-14}$

6. Conclusions and Future Works

This paper presented a proposal for failure detectors based on neural networks for a real-time distributed control system. Through the experiments, we have shown the advantages of this proposal in relation to conventional approaches for distributed systems. The results demonstrated that in an environment of moderate traffic, the implementation of the detector does not influence in control system performance. All algorithms (*ANN*, traffic analyzer, *PID*, and failure detectors) have been implemented using the *MatLab script* language. As future work, it is intended to evaluate the impact of the detector presented (and related fault recovery mechanisms) in control systems with multiples sensors, controllers and actuators.

7. References

Bertier, M., Marin, O., and Sens, P. (2002). Implementation And Performance Evaluation Of An Adaptable Failure Detector. In *Proc. Of The International Conf. On Dependable Systems And Networks*, pages 354–363, Washington, Dc.

Chen, W., Toueg, S., and Aguilera, M. K. (2002). On The Quality of Service of Failure Detectores. *IEEE Transactions On Computer*, 51(2):561–580.

Felber, P. (1998). The Corba Object Group Service : A Service Approach to Object Groups in CORBA. Tese de doutorado em informática, DÉPARTEMENT D'INFORMATIQUE, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE.

Haykin, S. (1994). *Neural Networks; A Comprehensive Foundation*. MACMillan, New York, 1st edition.

Henriksson, D. and Cervin, A. (2003). Truetime 1.13 - Reference Manual. Technical Report Isrn Lutfd2/Tfrt--7605--se, Department Of Automatic Control, Lund Institute Of Technology.

Hermant, J. and Lann, L. (2002). Fast Asynchronous Uniform Consensus in Real-Time Distributed Systems. *IEEETC: IEEE Transactions on Computers*, 51.

Jacobson, V. (1988). Congestion Avoidance And Control. *ACM Computer Communication Review; Proc. Of The Sigcomm '88 Symposium In Stanford, Ca, August, 1988*, 18, 4:314–329.

Jalote, P. (1994). *Fault Tolerance In Distributed Systems*. Prentice Hall, New Jersey.

Lian, F., Moyne, J. R., and Tilbury, D. M. (2001). Performance Evaluation Of Control Networks: Ethernet, Controlnet, And Devicenet. *IEEE Control Systems Magazine*, 21:66–93.

Lian, F., Moyne, J. R., and Tilbury, D. M. (2002). Network Design Consideration For Distributed Control Systems. *IEEE Transactions On Control Systems Technology*, 10:297–307.

Lynch, N. A. (1996). *Distributed Algorithms*. Morgan Kaufmann, San Francisco, California.

Macêdo, R. J. A. and Lima, F. R. L. (2004). Improving The Quality of Service of Failure Detectors with SNMP and Artificial Neural Networks. In *Proc. of Anais do 22o. Simpósio Brasileiro de Redes de Computadores.*, pages 583–586, Brazil, Gramado.

Nunes, R. C. and Jansch-pôrto, I. (2004). Qos of Timeout-based Self-tuned Failure Detectors: The Effects Of The Communication Delay Predictor And The Safety Margin. In *Proc. of International Conf. On Dependable Systems And Networks*, pages 753–761.

Ogata, K. (1990). *Modern Control engineering*. Prentice Hall, Englewood Cliffs, 2nd edition.

Riedmiller, M. and Braun, H. (1993). A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proc. of 1993 IEEE International Conf. on Neural Networks*, volume 1, pages 586–591, San Francisco, California. IEEE/INNS. U. Karlsruhe.

Tanenbaum, A. S. (2003). *Computer Networks*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 4th edition.

The Mathworks (2004). *Simulink Reference:Simulation and Model-Based Design*. The Mathworks Inc., Nantick, USA.