A Neural Network Approach for Controller Area Network Message Scheduling Control

Chuan Ku Lin, Hao-Wei Yen, Mu-Song Chen, and Chi-Pan Hwang

***Abstract*—In this paper, a novel approach for Controller Area Network (CAN) message scheduling control is proposed. The proposed method utilizes a single hidden layer neural network, i.e. Radial Basis Function Network (RBFN), such that the CAN's bandwidth can be shared properly among different messages. CAN was initially developed for the automotive industry to solve the cabling problems found in some kinds of vehicles. However, CAN is not able to satisfy all the communication needs of the industrial automation and process control scenarios. In particular, the basic protocol in CAN can neither ensure deterministically bounded transmission times for RT (real-time) data exchanges nor a fair share out of the network bandwidth among non-RT application processes. Therefore, CAN must carry both RT messages as well as non-RT messages in order to meet the requirements of a dynamic distributed system. All these messages must be properly scheduled on the network so that RT messages meet their deadlines while co-existing with non-RT messages. To solve these problems, we present a neural network approach to dynamically schedule and distribute the system bandwidth. At the same time, the transmission time can meet the basic requirements. The experimental results clearly show the effectiveness of the proposed technique in solving the message scheduling problems in CAN.**

***Index Terms*—Neural Networks, Controller Area Network, Message Scheduling Controller, Radial Basis Function Network, Backward Through Time**

I. Introduction

CAN [1][2] is a bus protocol with many desirable properties for embedded and real-time systems. It is a priority-based mechanism where collisions are avoided by using priorities for bus arbitration. The priority-based arbitration mechanism requires that different CAN nodes never simultaneously send messages with equal identifiers. According to this mechanism, as soon as the bus is idle, each node competing for the bus begins to send the arbitration field of its message. At the end of the arbitration field, only the node which is sending the message with the lowest arbitration field value, will be transmitting.

CAN must carry both periodic and sporadic real-time messages, as well as non real-time messages. All these messages must be properly scheduled on the network so that real-time messages meet their deadlines while co-existing with non real-time messages. Previous work regarding scheduling such messages on CAN focused on fixed-priority mechanisms [3][4]. The Fixed-priority deadline-monotonic (DM) scheduling [5][6][7]achieves meeting deadlines as guaranteed by an off-line feasibility test for a static system with periodic tasks. Due to the tight relationship between the identifier and the priority of CAN messages, the fixed priority assignment has been applied in the most common CAN-based communication systems implicitly. Although static systems can be scheduled easily by DM scheduling, it does not allow scheduling of dynamic systems, where an offline feasibility test has incomplete knowledge about the future behavior of the system. In general, fixed-priority schemes give lower utilization than other schemes such as nonpreemptive earliest deadline first (EDF) [8]. Livani introduced a mechanism to assign dynamic priorities to CAN messages, in order to achieve an EDF resource access consensus among the participating nodes. Based on the EDF access mechanism, soft real-time communication will be scheduled optimally with EDF approach. To guarantee deadlines of hard real-time communication, calendar-based resource reservation is applied. An application of this scheduling approach in a distributed object-oriented real-time control system has been introduced by [10]. The drawback of EDF is its high overhead which makes EDF impractical for CAN. On the other hand, a scheduling mechanism for the CAN bus, based on a mixture of dynamic and static priorities, has been approached by [9]. However, this approach makes unrealistic assumptions about CAN (10 Mbits/s) and exhibits a rather restricted scheduling ability due to a short time horizon.

In this paper, we propose a controller-plant model to schedule different types of messages on CAN. These messages are classified into three broad categories, (1) hard real-time (HRT) messages, (2) soft real-time (SRT) messages, and (3) non real-time (NRT) messages, respectively. The allocation of the higher priority class of HRT messages must be made for the time-critical events and the lower priority class for the NRT messages. The message buffer, containing messages in queue, can be regarded as a plant to be controlled by controller. We take advantage of the learning capability of neural networks and present a messenger scheduling controller (MSC), which is implemented by Radial Basis Function Network (RBFN) [11]. The MSC can be treated as a message dispatcher, which decides a proper message type to be sent. Moreover, a novel on-line Backward Through Time (BTT) algorithm is suggested such that the supervised learning method can be applied to RBFN. Fig. 1 shows the complete framework of the proposed controller-plant model. In the following, we introduce the basic structure of RBFN and its modeling training technique in section II. However, the supervised learning method is

(Advance online publication: 13 February 2007)

inadequate to apply for the CAN message scheduling. We, therefore, devise a BTT algorithm to extract the desired outputs for supervised training in section III. In section IV, simulation result is demonstrated by applying RBFN and the BTT algorithm. Finally, the conclusions are drawn in section V.

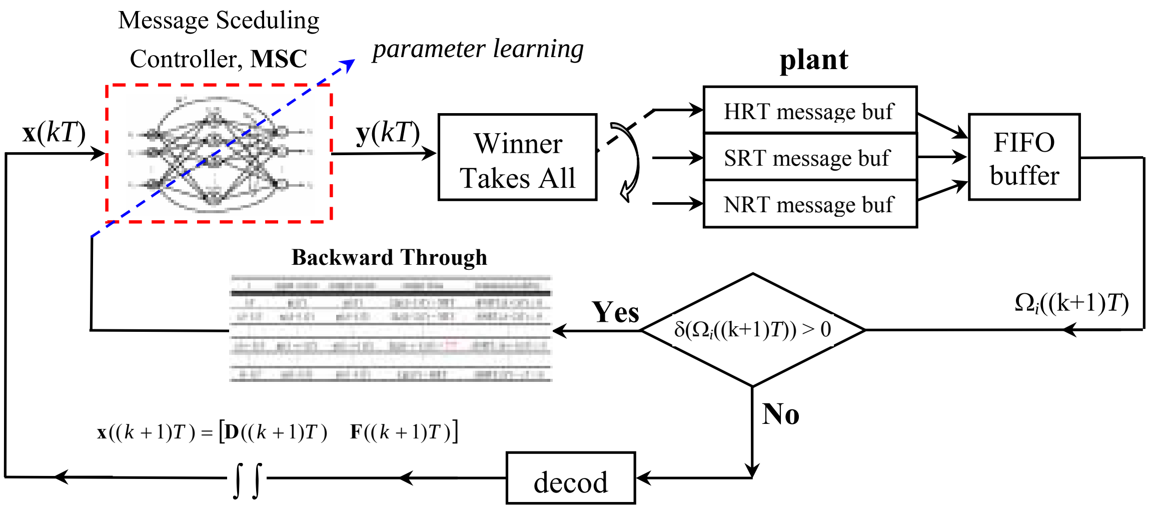

Fig. 1. The proposed controller-plant model for CAN message scheduling.

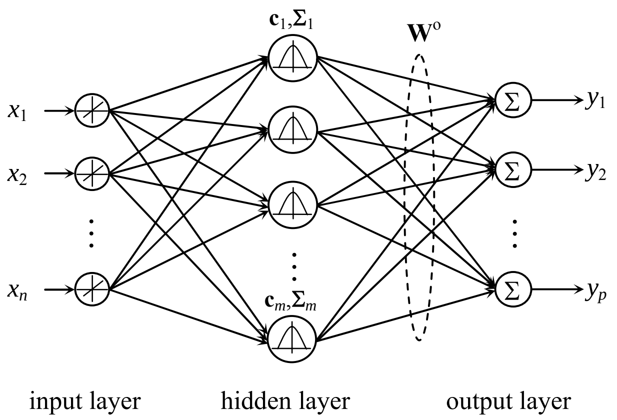

Fig. 2. A multi-input multi-output Radial Basis Function network.

II. Radial Basis Function Network

Radial Basis Function network (RBFN) [12] is a special neural network architecture that consists of three layers, i.e. input layer, hidden layer, and output layer. Fig. 2 depicts a multi-input multi-output RBFN. The values of the input variables formulate an input vector, which is forwarded from the input layer to the hidden layer. The hidden layer is comprised of a number of nonlinear processing nodes, which are characterized by the center locations and the nonlinear radial basis function. Each hidden node receives the input vector, calculates the (Euclidean) distance $\|\mathbf{x} - \mathbf{c}_i\|$ between the center vector $\mathbf{c}_i$ and the input vector $\mathbf{x}$, and finally performs a nonlinear transformation of the distance, using the radial basis function. The output of each hidden node is then multiplied by a particular synaptic weight, while the final output of the network is a simple summation of all the weighted hidden node activations. In this paper, we employ Gaussian function as the basis function

$$\psi_i(\mathbf{x},\mathbf{c}_i,\mathbf{\Sigma}_i) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_i)^{\mathrm{T}}\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{c}_i)\right) \qquad (1)$$

where $\mathbf{c}_i$ and Σ_i are center vector and covariance matrix, respectively. In Fig. 2, $\mathbf{W}^{\mathrm{o}}$ are connection weights from hidden layer to output layer. In this way, the free parameter vector in RBFN can be defined by

$$\mathbf{\Theta} = \left\{ W_{ij}^{O}, \mathbf{c}_i, \mathbf{\Sigma}_i \right\}, \quad \forall\ i, j \qquad (2)$$

Instead of solving all the network parameters $\mathbf{\Theta}$ in one step, the optimization problem is divided into two phases, where the locations of the RBF's center and covariance matrix are computed first, and the weighting connections are calculated in the second phase.

With a predefined error function (3)

$$E = \frac{1}{2}\sum_j (z_j - y_j)^2 = \frac{1}{2}\sum_j e_j^2 \qquad (3)$$

$\mathbf{\Theta}$ in (2) can be modified according to gradient descent algorithm iteratively, i.e.

$$\theta(t+1) = \theta(t) + \eta\Delta\theta(t) = \theta(t) - \eta\frac{\partial E}{\partial\theta(t)} \qquad (4)$$

where $\theta \in \mathbf{\Theta}$. In (5) and (6), we summarize the iterative formula of the steepest descent method

$$\frac{\partial E}{\partial W_{ij}^{O}} = -e_j\psi_i \qquad (5)$$

$$\left\{\begin{array}{l} \dfrac{\partial E}{\partial \mathbf{c}_i} = -e_j W_{ij}^{O}\psi_i\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mathbf{c}_i) \\ \dfrac{\partial E}{\partial \mathbf{\Sigma}_i} = -e_j W_{ij}^{O}\psi_i \begin{bmatrix} \frac{(x_1-c_{i1})^2}{\sigma_{i1}^3} & 0 & \cdots & 0 \\ 0 & \frac{(x_2-c_{i2})^2}{\sigma_{i2}^3} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{(x_n-c_{in})^2}{\sigma_{in}^3} \end{bmatrix} \end{array}\right. \qquad (6)$$

In the next section, we will discuss message scheduling control and how the training data can be generated for supervised learning.

III. Message Scheduling Control and Backward Through Time (BTT) Algorithm

In this section, we present a complete framework for online adaptation of RBF networks that can be used for dynamical message scheduling control. In our CAN model, the transmission time $D(\Omega_i)$

$$D(\Omega_i) = T_E(\Omega_i) - T_S(\Omega_i) \qquad (7)$$

is defined as the time which elapses from an application process making a transmission request (i.e. $T_S(\Omega_i)$) to the reception of the last bit of the frame by the intended receiver(s) (i.e. $T_E(\Omega_i)$). The subscript i of Ω_i denotes one of predefined messages, $\Omega_i \in \{\text{HRT,SRT,NRT}\}$. If each class of message Ω_i has its own predefined maximum predefined transmission time $D_{\max}(\Omega_i)$, then the most effective index to evaluate the performance of message scheduling is the transmission delay $\delta(\Omega_i)$, defined as the time elapsing from the transmission request to the successful transmission of the first bit of the frame over the network. In formula,

$$\delta(\Omega_i) = D(\Omega_i) - D_{\max}(\Omega_i) \qquad (8)$$

According to the definition of transmission delay $\delta(\Omega_i)$, message Ω_i is said to be timely transmitted if $\delta(\Omega_i) \le 0$ (or $D_{\max}(\Omega_i) \ge D(\Omega_i)$). If $\delta(\Omega_i)$ is greater than zero, however, message Ω_i can't guarantee its deadline requirement or is notified of the deadline failure. In Fig. 3(a), message Ω_i can't complete its transmission until $T_E(\Omega_i)$, which exceeds the predefined limit for message Ω_i. In this case, message Ω_i fails to fulfill the requirements of timely transmission. This causes performance degradation and even instability of real-time application systems interconnected into the CAN bus. On the contrary, Fig. 3(b) is the case of timely transmission.

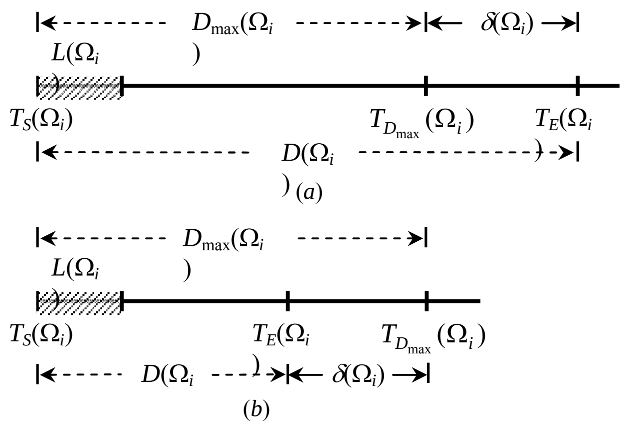

Fig. 3. (a). Message Ω_i can't complete its transmission before $T_{D_{\max}}(\Omega_i)$ (b). case of timely transmission.

To avoid starvation for the low-priorities objects, we did a experiment by forcing a fixed delay $L(\Omega_i)$ after any transmission requests for different priority classes. This method, however, simply limits in advance the bandwidth allocated to each type of message, thus, decreasing the wasted network bandwidth. In this way, the delay vector is $\mathbf{L} = [L(\text{HRT})\ L(\text{SRT})\ L(\text{NRT})]^{\text{T}} = [n_1T\ n_2T\ n_3T]^{\text{T}}$. Moreover, it requires also a certain degree of *a priori* knowledge of the maximum allowable transmission time and the network traffic load. Therefore, we assumed that $\mathbf{D}_{\max} = [D_{\max}(\text{HRT})\ D_{\max}(\text{SRT})\ D_{\max}(\text{NRT})]^{\text{T}} = [100T\ 150T\ 200T]^{\text{T}}$ and the traffic loads are $F(\Omega_i) = 30$ for $\Omega_i \in \{\text{HRT,SRT,NRT}\}$. Table 1 and Table 2 illustrate the transmission time under different conditions of vector $\mathbf{n} = [n_1,n_2,n_3]^{\text{T}}$. In Table 1, all messages are timely transmitted (i.e. $\delta(\Omega_i) < 0$). However, too large of the value of n_2 in Table 2 is notified of the deadline failure for SRT message (i.e. $\delta(\text{SRT}) > 0$). The simulation results reveal the fact that values of n_i affects the performance of message scheduling significantly. We therefore reasonably assume that there exists

Table 1. Transmission time v.s (n_1,n_2,n_3)=(60,85,0).

$D(\Omega_i)$	Min	Mean	Max	$\delta(\Omega_i)$
D(HRT)	$61T$	$61.45T$	$90T$	$\delta(\text{HRT}) < 0$
D(SRT)	$86T$	$102.33T$	$120T$	$\delta(\text{SRT}) < 0$
D(NRT)	$1T$	$124.07T$	$180T$	$\delta(\text{NRT}) < 0$

Table 2. Transmission time v.s (n_1,n_2,n_3)=(50,140,0).

$D(\Omega_i)$	Min	Mean	Max	$\delta(\Omega_i)$
D(HRT)	$51T$	$51.37T$	$80T$	$\delta(\text{HRT}) < 0$
D(SRT)	$141T$	$154.36T$	$200T$	$\delta(\text{SRT}) > 0$
D(NRT)	$1T$	$122.53T$	$162T$	$\delta(\text{NRT}) < 0$

We strongly believe that there exists some implicit relationships between traffic load $\mathbf{F}$, delay vector $\mathbf{L}$, and $\delta(\Omega_i)$. If $\mathbf{n}$ or $\mathbf{L}$ is chosen properly in advance, $\delta(\Omega_i)$ can be completely decided. In fact, vector $\mathbf{n}$ or $\mathbf{L}$ can only be selected by trial-and-error. Moreover, the network traffic load is usually time-varying, values of (n_1,n_2,n_3) has to be changed accordingly. It implies $\mathbf{F}$ and $\mathbf{L}$ are time-dependent, $\mathbf{F} = \mathbf{F}(t)$ and $\mathbf{L} = \mathbf{L}(t)$. If we consider the competing time between heterogeneous and homogeneous classes, the implicit function between $\mathbf{F}$, $\mathbf{D}$, and $\delta(\Omega_i)$ can be reasonably expressed as

$$\Phi(\mathbf{F}(t), \mathbf{D}(t), \delta(\Omega_i)) = 0 \qquad (9)$$

In (9), $\mathbf{F}(t) = [F(\text{HRT},t)\ F(\text{SRT},t)\ F(\text{NRT},t)]^{\text{T}}$ and $\mathbf{D}(t) = [D(\text{HRT},t)\ D(\text{SRT},t)\ D(\text{NRT},t)]^{\text{T}}$. Unfortunately, the function Φ is highly complex and the explicit solution is almost impossible to find or doesn't exist. To optimally utilize the system resources for different types of messages while guaranteeing timely transmission, we are not intended to find the exact solution. Instead, we propose a neural network approach for message scheduling control. The controller is realized by several radial basis functions and forms a radial basis function network. The purpose of the RBFN is to reveal the implicit relationship between the input vectors $\mathbf{x}(t) = [\mathbf{D}(t)\ \mathbf{F}(t)]^{\mathrm{T}}$ and a proper output vector $\mathbf{y}(t)$, which features resource reservation as well as dynamic message scheduling such that an appropriate class of message to be served in the next time instant. Since the proposed MSC is implemented by the RBFN network, in the rest of the paper MSC and RBFN will be used interchangeably.

At any time instant t, the input vector $\mathbf{x}(t) = [\mathbf{D}(t)\ \mathbf{F}(t)]^{\mathrm{T}}$ of the MSC consists of transmission time and the current network flow rate, where $\mathbf{D}(t) = [\mathrm{D}(\mathrm{HRT},t)\ D(\mathrm{SRT},t)\ D(\mathrm{NRT},t)]^{\mathrm{T}}$ and $\mathbf{F}(t) = [F(\mathrm{HRT},t)\ F(\mathrm{SRT},t)\ F(\mathrm{NRT},t)]^{\mathrm{T}}$. Consequently, the inferred output $\mathbf{y}$ after WTA (Winner Takes All) identifies the message to be served in the next time instant. For optimal message scheduling, the parameters of the MSC of the RBFN have to be designed by the so-called supervised learning. The learning employs two levels of adaptation, namely, adaptation of the connection weights between the hidden layer and the output layer and adaptation of the parameters of the radial basis functions. Unfortunately, the optimal message scheduling is impossible to predict in advance. Consequently, without complete knowledge about the target vector $\mathbf{z}$, the parameter learning method in (5) and (6) can't proceed. Moreover, in many applications, it would also be desirable for an online training algorithm to be able to provide stable outputs, whenever the system dynamics change.

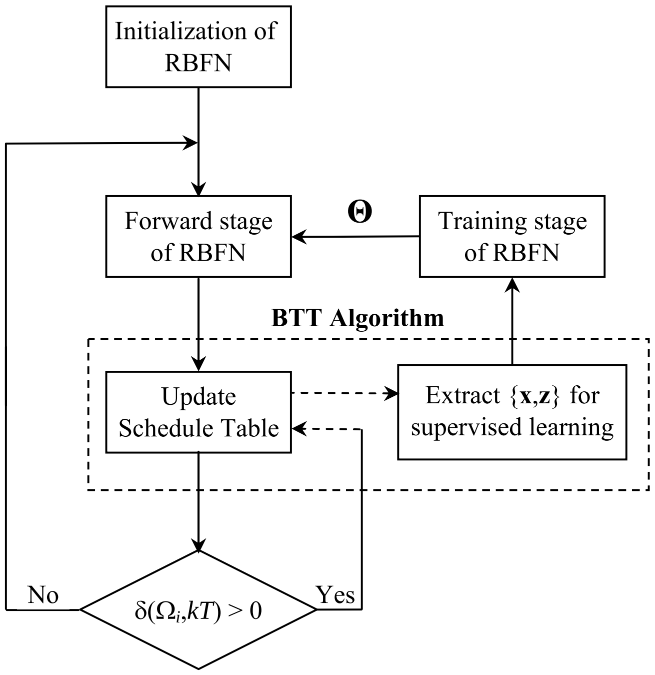

Fig. 4. On line Backward Through Time (BTT) algorithm.

Therefore, in this paper we devise an on-line Backward Through Time (BTT) learning algorithm which continuously monitors $\delta(\Omega_i,t)$ for message Ω_i and constructs a schedule table sequentially as shown in Table 4. In the beginning, the schedule table records both input vector $\mathbf{x}$ and the corresponding vector $\mathbf{y}$ (and Ω_i), based on the default parameter $\boldsymbol{\Theta}$ in the RBFN. If at any time instant t, $\delta(\Omega_i,t) = rT$ is greater than zero, it implies that the transmission time for the corresponding message is unacceptable or the bandwidth allocation in the previous time instant is improper. At this point, we retrieve the vector $\mathbf{x}(t-(r\text{-}1)T)$ from the schedule table and reassign its corresponding output class (or target) by the current type of message. Consequently, the desired target vector $\mathbf{z}$ can be extracted. This vector $\{\mathbf{x}\ \mathbf{z}\}$ sets up the necessary information for on line supervised learning and the parameters of RBFN can be tuned accordingly. Fig. 4 shows the complete framework for online BTT adaptation algorithm of RBF networks.

In the following, we conduct some experiments to evaluate the performance of the proposed BTT method in CAN message scheduling.

IV. Experimental Results

In the simulation of message scheduling, we assume the number of radial basis functions for the MSC is fixed and known in advance. The messages generated at each node by the user are inserted according to a FIFO (First In First Out). The message is generated randomly in a predefined range and is defined as the traffic load $F(\Omega_i)$. Table 3 summarizes these necessary information, including $D_{\max}(\Omega_i)$, $F(\Omega_i)$, and the number of RBFs.

Table 3. Simulation parameters.

Maximum transmission time	$D_{\max}(\mathrm{HRT}) = 120$, $D_{\max}(\mathrm{SRT}) = 200$, $D_{\max}(\mathrm{NRT}) = 400$
Traffic load	$F(\mathrm{HRT}) = 30\sim50$, $F(\mathrm{SRT}) = 30\sim50$, $F(\mathrm{NRT}) = 30\sim50$
# of RBFs	7

In the beginning, the parameters of MSC are initialized randomly. The MSC shares out the system bandwidth among three different messages fairly and all messages are timely transmitted. As time goes on, the HRT messages can only transmit using the residual bandwidth that is left after SRT and NRT messages have been allocated. Therefore, HRT messages become less and less competed with others since HRT messages have the smallest allowable transmission time. Consequently, the transmission time of HRT message can not fulfill with the requirement of timely transmission. On the contrary, the transmission time of SRT and NRT are far below $D_{\max}(\mathrm{SRT})$ and $D_{\max}(\mathrm{NRT})$. However, in this time interval, MSC continuously activate the on line BTT algorithm as mentioned in section III. The behaviors of the RBFN intend to give a higher priority to the HRT messages and allocate more system bandwidth for HRT messages while keep the SRT and NRT messages meet the basic requirements. Fig. 5 reveals this fact.

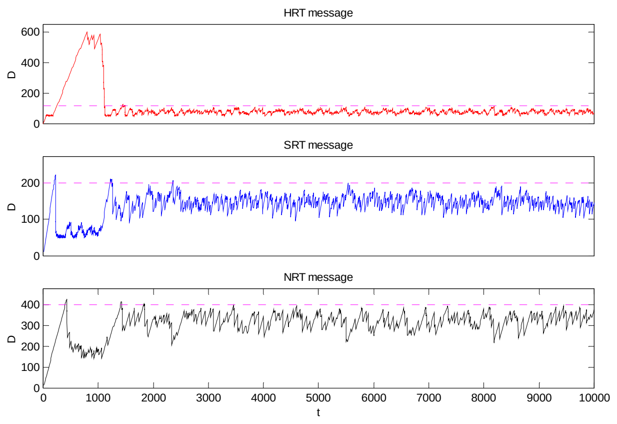

Fig. 5. Simulation results of message scheduling under the conditions of Table 3. The dotted lines denote the maximum allowable transmission time.

We also calculate the ratio $\gamma(\Omega_i)$ of message Ω_i that fail to accomplish its transmission before $D_{\max}(\Omega_i)$ versus total number of that type message being transmitted. In Fig. 6, $\gamma(\Omega_i)$ is recorded from Fig. 5. As was expected, the curves of $\gamma(\Omega_i)$ are gradually improved and become level off when the system is stable.

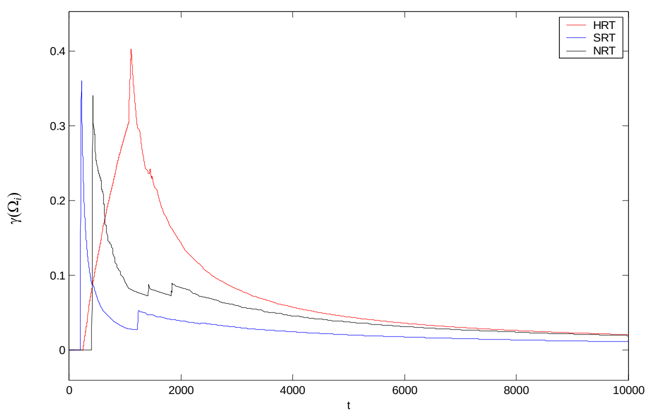

Fig. 6. Curves of $\gamma(\Omega_i)$ versus *t*.

V. Conclusion

The CAN is a network protocol that supports communication among field devices in the control and distributed systems. This study introduces a bandwidth allocation algorithm in the CAN bus such that different types of messages, hard real-time, soft real-time, and nonreal-time messages, can satisfy their basic requirements for timely transmission. The proposed MSC is implemented by radial basis function networks which dynamically decide the proper type of message to be served in the next time instant. Simulation results demonstrate the effectiveness of the proposed method. In out current work, the number of RBFs is fixed during training phase. For modeling dynamical time varying systems, the number of RBFs should be adaptively changed in order to capture the dynamics of the CAN bus system. That means the proposed algorithm should not only tune the parameters of RBFN but also add or delete some RBFs. In this way, the MSC or the RBFN can retain a reasonable size, but at the same time describe well of the system at any time instant. This involves the so-called structure learning strategy with the parameter learning method. This research topics is still under investigation and will be our future works.

Acknowledgment

This research was supported by the National Science Council under contract number NSC94-2213-E-212-036.

References

[1] Robert Bosch, "CAN Specification Version 2.0", Bosch, Sep. 1991.

[2] Farsi, M., Ratcliff, K., and Barbosa, M, "An overview of controller area network," Computing & Control Engineering Journal Vol. 10(3), P. 113-120, June 1999.

[3] Zuberi, K.M., & Shin, K.G., "Non-preemptive scheduling of messages on controller area network for real-time control applications," *Proc.of the first IEEE real-time technology and applications symposium*, pp. 240-249, 1995.

[4] K. Tindell, A. Burns, and A. J. Wellings, "Calculating controller area network (CAN) message response times," Contr. Eng. Practice, vol. 3, no. 8, pp. 1163–1169, 1995.

[5] Audsley, N.C., A. Burns, M.F. Richardson, A.J. Wellings (1991). Hard Real-Time Scheduling : The Deadline Monotonic Approach. Proceedings of 8th IEEE Workshop on Real-Time Operating Systems and Software.

[6] Tindell, K. and A. Burns (1994). Guaranteeing Message Latencies on Control Area Network (CAN). First International CAN Conference 94.

[7] Tindell, K., A. Burns, A. Wellings (1995). Calculating Controller Area Network (CAN) Message Response Times. Control Engineering Practice, 3(8):1163-1169.

[8] Livani, M.A. and J. Kaiser (1998). EDF Consensus on CAN Bus Access for Dynamic Real-Time Applications. Lecture Notes in Computer Science 1388 (Jose Rolim Ed.), pp. 1088-1097, Springer Verlag Berlin, 1998.

[9] Zuberi, K.M. and K.G. Shin (1995). Non-Preemptive Scheduling of messages on Controller Area Network for Real-Time Control Applications. Proc. Real-Time Techn. and Appl. Symposium.

[10] Kaiser, J. and M.A. Livani, "Invocation of Real-Time Objects in a CAN Bus-System," First Int'l Symposium on Object-Oriented Distributed Real-Time Computing Systems, Kyoto, 1998.

[11] Uykan, Z., Guzelis, C., Celebi, M.E. and Koivo, H.N, "Analysis of input-output clustering for determining centers of RBFN" IEEE Trans. on Neural Networks, Vol. 11(4), pp. 851-858, July 2000.

[12] S. Haykin. Neural Networks: A Comprehensive Foundation. Macmillan College Publishing Company, New York, 1994.

Table 4. Schedule Table

t	input vector	output vector	output class	transmission delay
kT	$\mathbf{x}(kT)$	$\mathbf{y}(kT)$	$\Omega_i((k+1)T)$ = NRT	δ(NRT,$(k+1)T) \leq 0$
$(k+1)T$	$\mathbf{x}((k+1)T)$	$\mathbf{y}((k+1)T)$	$\Omega_i((k+2)T)$ = SRT	δ(SRT,$(k+2)T) \leq 0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(k\text{-}r\text{-}1)T$	$\mathbf{x}((k\text{-}r\text{-}1)T)$	$\mathbf{y}((k\text{-}r\text{-}1)T)$	$\Omega_i((k\text{-}r\text{-}1))T)$ = ???	δ(SRT, $(k\text{-}r\text{-}1))T) \leq 0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(k\text{-}1)T$	$\mathbf{x}((k\text{-}1)T)$	$\mathbf{y}((k\text{-}1)T)$	$\Omega_i(kT)$ = HRT	δ(HRT,$kT) = rT > 0$