International Journal of Soft Computing 2 (1): 103-112, 2007

CPU Secheduling with Fieldbus Systems

[1]Shatha K. Jawad and [2]Munther N. Al-Tikriti
[1]Al-Balqaa Applied University, Faculty of Engineering, Sallt, Jordan
[2]Faculty of Engineering, P O Box1, Philadelphia University, Jordan, 19392

Abstract: In this study, a strategy for an industrial standard, the IEC/ISA Field bus, is proposed to support the intracell and LAN for intercell real time communications. The designed distributed control system consists of autonomous CPUs that work together to make the complete system look like a single computer. So, the microkernel design using two levels of scheduling, the first one for communication and the other for task execution. Level two is proposed by using three routines, guarantee, bidder and decision routines. The Decision routine is implemented using Fuzzy Logic set, neurofuzzy and a mathematical approach. The results obtained, using Neurofuzzy, are approximately the same choice as for fuzzy but it responds faster than it. On the other hand the functional Neurofuzzy is the best algorithm compared with structural and fuzzy. The proposed mathematical approach can be used to implement the latter algorithm. This is because it needs little and simple calculations and no stored data, in making its decision, are needed.

Key words: Fieldbus, neurofuzzy, CUP

INTRODUCTION

In the mid-1980s, two advances in computer technology were developed. The first one was the development of powerful microprocessors and the second was the invention of high-speed computer networks. The Local Area Networks (LANs) allow dozens, or even hundreds, of machines within a building to be connected in such a way that small amounts of information can be transferred between machines in a millisecond or so. Larger amounts of data can be moved between machines at rates of 10 to 100 million bits sec^{-1} and sometimes more. The Wide Area Networks (WANs) allow millions of machines all over the world to be connected at speeds varying from 64 Kbps (Kilo bits per second) to giga bits per second for some advanced experimental networks[1-5].

The result of these technologies is that it is now easy and feasible to put together computing systems composed of large numbers of CPUs connected by a high-speed network. They are usually called distributed systems[6,7].

On the other side, the evaluation of process control systems towards distributed architectures has led in the last few years to the introduction of fieldbus networks, which provide a more efficient interconnection of field devices compared with traditional systems[6,7].

Fieldbus was defined as a totally digital communication protocol that replaces traditional analog wiring and signaling approach[6,7].

Related works: Several works investigated the problem of industrial communication networks, which differ from other types of networks because these applications are responsible for the control and monitoring of physical processes. So, these works describe the advantage of using fieldbus system over the traditional networks[6,8,9-14].

Chih-Che, and Kang[15] proposed a scheme which can provide real-time communication services with both absolute and statistical performance guarantees on multi-access bus networks for given input traffic characteristics and performance requirements.

Some works are directed to develop the relation between Data Link layer (DLL) and Application layer of fieldbus system[16-19]. For example Salvator, C.[16] proposed a method to analyze the communication models currently offered by the application layer and their implementation through the mechanism present in the DLL.

Gonzalo[20] proposed a model that present the Application layer of fieldbus as a part of a real-time distributed system as well as the proposition a protocol with very low network and computational overhead.

Some works directed towards scheduling in fieldbus system[7, 21]. The most widely used strategies consists of drawing up off-line scheduling tables, whose length is equal to the LCM of all process periods, containing this transmission sequence. This strategy is not appropriate for fieldbus because the scheduling table required large memory size, which is excessive for simple field devices. Ali, U.M.[7] found a solution for this problem. The solution based on Boltzmann machine type of Neuro Network was

Corresponding Author: Munther N. Al-Tikriti, Faculty of Engineering, P O Box1, Philadelphia University, Jordan, 19392

developed, which reduces the computational complexity of periodic process scheduling greatly, allowing real-time adaptation of the scheduling table to change in the process control environments.

Networks for factory automation, often named fieldbuses have been developed to fulfill real-time requirements[22-24]. Kunert, O. and Zitterbart, M.[23] described a model for internetworking fieldbuses via ATM networks. A remote bridge is used to span longer distances between fieldbus components in separated fieldbus. Paulo, V.[24] proposed an analytical study of the inaccessibility of CAN and PROFIBUS.

Kang G. S. and Chih-Che, C.[22] proposed a strategy for an industrial standard, the SP-50 fieldbus to support both intracell and intercell real-time communication. This strategy divides the capacity of each link into two parts. The first part is managed by the Local Link Active Scheduler (LAS) for intracell (intralink) communicates. The second part is managed by a proposed global network manager for intercell (interlink) communication.

Hindo, B.K.B.[25] focused on the problem of the Multi-Mastership in the fieldbus when more than one master resides on the same fieldbus segment. Two approaches are used to arrange the access to the bus, the centralized and the distributed approaches.

THE PROPOSED SYSTEM MODEL

In the proposed system the multi-workcells cooperate to perform their tasks by using distributed system instead of centralized system because of its advantages over the latter. However, the connection of any workcell to another one is performed using one of the following two approaches:

- Connecting all LASs found in the process control system by using LAN.
- Connecting the fieldbuses together using bridges.

The advantages of each connection are described latter.

Figure 1 shows the proposed system model. The reason of existence $\overline{PC}$ is to provide redundancy and consequently increasing system reliability. Each PC [LAS] is responsible to perform the following operations:

- Provide control algorithms needed by its workcell (local DLE's).
- Provide control algorithms that may be needed by another workcells found in the same control system.
- Work as a general-purpose computer to solve any problem defined by the user.

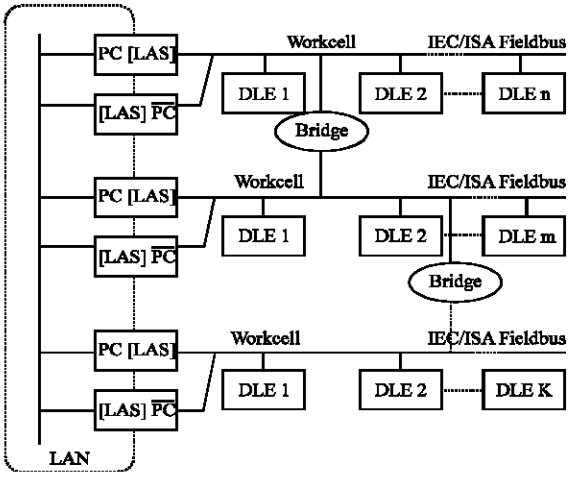

Fig. 1: The proposed system model

- Help the user to monitor and to make any necessary change (e.g. initialization, scaling) for local workcell.
- Perform all jobs of LAS, e.g. perform the required scheduling algorithms to divide its time between tasks in order to obtain better utilization and never exceed the deadline of each task.
- Must provide the following methods for connection:
- Link each DLE to another DLE found in the same workcell, which is known as intracell connection.
- Link any DLE found in a workcell to another DLE in another workcell, which is known as intercell connection.

Each PC contains the same distributed operating system to perform its jobs. This study deals with the microkernel of this distributed operating system by improving the schedulability of the tasks and provides an idea for interprocess communication mechanism in the proposed system model.

INTERPROCESS COMMUNICATION IN THE PROPOSED SYSTEM

A failure to complete a real-time task before its deadline could cause a disaster[22,26]. However, if task arrival is unevenly distributed over PC's in a distributed real-time system, some PC's may become overloaded and thus unable to complete all their tasks in time, while other PC's are unloaded. In such a case, even if the total processing power of the system is sufficient to complete all incoming tasks in time, some tasks arriving at overloaded PC may not be completed in time. One way to alleviate this problem is load sharing, some of those tasks arriving at overloaded PC's are transferred to

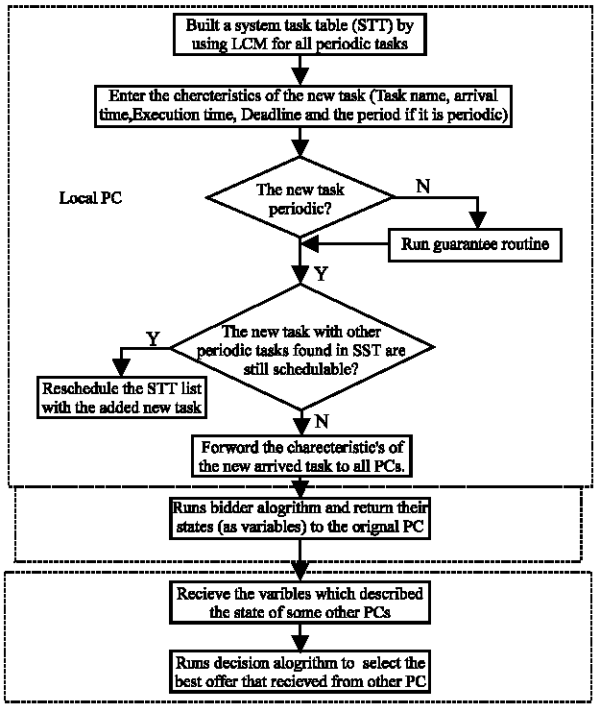

Fig. 2: The forwarding of arrived new task algorithm.

underloaded PC's for execution[26-31]. There is a question here, which task can be guaranteed by local PC and which task must be migrated? To answer the question, first the type of the task must be defined to see whether it is periodic or aperiodic and remember that the PC has an important function that is scheduling the transmission time in local fieldbus. This means that the scheduling mechanism must be introduced to know if a local PC can serve the new task or migrate it to another one.

Because each workcell consists of fieldbus system, then it is required to introduce two level of scheduling for both periodic and aperiodic tasks in each of them:

Level one-fieldbus scheduling: This level of scheduling is to find the instance of transmission by using the 2nd method, which is proposed by Ali[7].

Level two-CPU time scheduling: Scheduling time of the CPU of each PC. This scheduling prepared depending on the transmission instances (found in level one) which represent the deadlines of the tasks. It means that the CPU must complete the required operation for each task before, or as close as possible, to its deadline (transmission instance).

CPU time scheduling: The microkernel, proposed in this work, consists of three algorithms which cooperate together to implement the CPU time scheduling at each PC found in the system, these are: Guarantee Algorithm, Bidder Algorithm and Decision-maker algorithm Fig. 2.

The three above algorithms need many messages to be transmitted between PC's, so a LAN is used to implement this function and to speed up the scheduling algorithm, this is because the designed system is real-time distributed system.So, the fieldbuses whose bandwidth is shared by field devices in advance, will be free from carrying the messages between PC's.

Some times, when a task is migrated to another PC the result must be returned to a DLE found in the original PC workcell (the PC that receives the new task from its workcell that was previously overloaded). In this case a bridge is used to perform the transmission of data (e.g. result) between the two corresponding fieldbuses. The time required for data transmission must be reserved in the bandwidth of both fieldbuses, the original fieldbus and the fieldbus which its PC migrated new task. This means that each PC (LAS) must give a token to the bridge connected between the two fieldbus and this can be calculated by using fieldbus scheduling algorithm.

Guarantee routine: To provide the three algorithms above, each PC (LAS) maintains a System Task Table (STT) for all local periodic or aperiodic tasks guaranteed at any point in time. In other words, the table which contains the time slot of any task and any additional information about any one of them (e.g., arrival time, execution time, etc.) known as STT. The STT can be built by using a dynamic priority assignment to implementing the EDF (Earlier Deadline First) rule. This STT is constructed along the least common multiple of the periods of all periodic tasks. Tasks in the STT are arranged in the order of their arriving times and, within each arrival time by their deadlines.

THE GUARANTEE ROUTINE FOR AN APERIODIC TASK

Two types of dynamic guarantee routines that can be successfully used in dynamic scheduling of the proposed systems' tasks are introduced. In both routines it is assumed that tasks are independent.

Each element in the STT list is a data structure called it TASK SLOT; that memorizes the processor sharing among the tasks. Each TASK SLOT is described as a Slice Time (End Time-Start Time) assigned to a task characterized by Arrival Time, Execution Time and Deadline. The time between two tasks represents Available Time, i.e. the time in which the processor is not busy. In other words, the STT list is constructed by simulating the processor behavior in advance. New Task

is a data structure that contains the characteristic of the new aperiodic task and characterized by Arrival time, Execution time and Deadline of this new task.

Algorithm analysis: The two-guarantee routine methods are similar in idea but different in the procedure of calculating the available time. As described before, the available time is used to execute the new aperiodic task.

In the first method, slotted task guarantee routine algorithm it is assumed that the tasks are preemptive, so the execution time of the new task can be divided into slots. The number of slots and their size depends on the available time found between any adjacent tasks with deadline less than the deadline of new task found in stt and the execution time of the new task. The steps of this algorithm are described as follows:

Step 1: Search Start Task in SST list, i.e. the time of TASK SLOT that end before the arriving of the new task arrival and in the same time built a new list that will contain all tasks in SST list and the new task if it is accepted.

Step 2: Search for Available Time (Inter Task Time) in the SST list, the intervals between tasks, i.e. time when the processor is idle. The search is completed when the sum of these intervals is equal to the execution time of the new task or when the deadline (New Task. deadline) is missed. The latter condition is tested by computing End of New Task as a sum of Inter Tasks Time and Tasks with deadlines less than that of the New Task. The tasks with deadlines greater than that of the new task are saved in another list of delayed tasks, let us call it Delayed Tasks List, which are arranged in order according to their deadlines.

Step 3: Check if tasks that are delayed by the new task are still schedulable.

For the second method, *Non-slotted Task Guarantee Routine Algorithm,* it is supposed that the tasks are non-preemptive, so the new task execution time can not be divided into slots. The following steps describe how this guarantee routine works:

Step 1: As described in step1-method one.

Step 2: Searching the STT list for the first task with deadline greater than that of new task. At the same time accumulate the available time and the execution time of all tasks between Start Task and the first task with deadline greater than that of new task. The search is stopped at any time the accumulator time is more than the deadline of the new task

Step 3: Inserts the new task in SST list before the first task with deadline greater than that of new task and add the execution time of the new task to the accumulated time calculated in step 2. If the new adding times go over the deadline then stop and conclude that the new task can not be guaranteed by this PC.

Step 4: Checks if tasks that are delayed by the new task (tasks, which have deadlines greater than that of new task) are still schedulable.

The dispatcher is the module that gives control of the CPU to the task selected from STT. Although the dispatcher should be as fast as possible, its time that is required to stop one task and starts another must be taken in to consideration. So, it is required to include the dispatcher's execution time within every task computation time (execution time). As a result, the conclusion is to use slotted task guarantee routine method, with increasing the execution time of the new task by a factor N, where N is define as follows:

N = time needed for each×slot's number of new task dispatcher

For worst case, the number of slots is equal to the execution time of new task. The time needed by dispatcher is an extra time added to the execution time

The new task is either periodic or aperiodic and must be examined for schedulability soon after it arrives. To facilitate this, both the bidder and the local scheduler tasks are executed as periodic tasks. The period and computation times of these tasks are determined a priori by the nature of tasks.

The above scheme is based on the assumption that there is a communication module, executed on a processor separate from the CPU on which tasks are scheduled. This is responsible for receiving information from local workcells' devices as well as from other PCs. Based on the type of communication, this module stores the received information in an appropriate data structure so that they will be looked at when different tasks are executed.

Bidder algorithm: When the new task, which is received by a local PC within a LAN, can not be scheduled locally its characteristic is broadcasted to all PC's through the LAN. Then each PC (except the local one) runs a bidder algorithm. This algorithm is used to determine incomplete information about the state of each PC running it. Each PC has two bidder algorithms one for periodic and other for aperiodic task.

Bidder algorithm for aperiodic tasks: A bidder algorithm starts to run when it is inspired by a Request For Bid signal (RFB). The local PC broadcasts this signal. The algorithm should return (to the origin PC) the degree to which the PC can guarantee the task.

Some authors[28,30] implemented schemes that evaluate the available time interval between the arrival of a RFB and the task deadline. They took into account all delays encountered during the process of bidding and the percentage of periodic tasks. Others[26,27] implemented schemes that periodically broadcast the PC state, measured as accumulated computational time or total number of tasks on that PC. These schemes consider only aperiodic tasks. The proposed idea for these schemes is that it is better to make an estimation of the state quicker than to make measurement with overhead[31,32]. In our approach, searching the STT within the interval between the arrival time and the deadline for new aperiodic task only and look for parameters that can influence the schedulability of the new task. The analysis is done in cooperation with the guarantee routine and EDF rule. Among the parameters that can influence the schedulability of a task the following assumption are considered:

- Available Time (AT)-Is the time between the arrival time and the deadline of the new task when the processor is idle.
- TD-Accumulated execution times of already guaranteed tasks that have to be delayed as a result of accepting the new task.
- ND-The Number of already guaranteed tasks that will be Delayed by the new task.

Note: TD and ND assumed to be as scheduling cost.

The bidder algorithm performs only an approximation of the above parameters and sends them to the initiator of RFB (local PC). If any PC in the system can not accept the new task no message will be returned. Because the proposal has two methods of guarantee routine, then two types of bidder algorithms are required. The difference between the two algorithms is: with slotted task guarantee routine method the related Bidder algorithm of any PC returns a message that describes the uncompleted states if the summation of the idle slots time are enough to execute the new task within its deadline. On the other hand and when using non-slotted guarantee routine method, the related bidder algorithm does not care about the pervious described case.

Bidder algorithm for periodic tasks: This algorithm is similar to the bidder algorithm of aperiodic tasks except the parameters that can influence the schedulability of the task which are, number of periodic tasks and number of aperiodic tasks.

To ensure that no PC accept any new task (periodic or aperiodic) until the local PC takes its decision, is to make each PC accept any new task after a fixed interval of time if it is not receive any message from the local PC.

The local PC waits for a fixed interval of time (estimated interval) for bids to come from all PC's. The local PC receives the bids and makes a decision to select the best bid among all bids within the waited interval time and neglects those bids that are arrived after that time.

Any PC does not return its bid to local PC unless it checks its transmission scheduling. It checks if it can re-scheduled the fieldbus transmission instance with the new task and gives token to the bridge at required time that the result must be transmitted beyond it to the DLE which need the answer of the new task found in original workcell.

Decision-maker algorithm: The decision-maker algorithm runs on the initiator of RFB (local PC) and uses information supplied by bidder algorithm from some PCs found in the system. The questions that should be answered by the decision-maker are: "1) Having received the bidding parameters from each PC, 2) Which is the best PC to send the task to?" It is not easy to give a complete answer, but could find partial ones. For example, for aperiodic task, a large available time that verifies the relation available time >execution time of the new task can guarantee the new task. However, it does not contain any information about the delayed tasks. In the same way, a small scheduling cost offers good chances to guarantee the delayed tasks. To handle this qualitative information about the parameters delivered by the PCs and their capacities to guarantee the new task, Fuzzy Sets theory, Neuro Fuzzy algorithm are used to introduce a mathematical approach that can be used to choose the best offer.

Fuzzy sets in decision-maker algorithm: The idea is to make particular use of approximate reasoning to find the qualitative dependence between bidding parameters and the capacity of a PC to accommodate a new task. A set of fuzzy rules is combined with an algorithm for firing them based on the current state of the system. The skeleton of fuzzy rules is written off-line. The parameters that the fuzzy rules built on depending on the type of the task, aperiodic or periodic. So the decision will depend on the type of the new task. If the new task is periodic then the two parameters of each PC are the number of periodic and aperiodic tasks, while for aperiodic new task

the parameters will be available time and scheduling cost. The above parameters, coming from the PCs as responses to RFB, are processed using fuzzy sets. The outcome of the rules is another fuzzy variable, Select, that has a value on the interval (0, 10) with a thirteen intervals (six intervals at each side of number five).

For aperiodic new task, different number of rules (9, 16, 25 and 36) are used with scheduling cost equal to the time of delayed task and number of delayed task. The same number of rules are implemented with periodic new task.

Once the selected PC receives a new task, it executes a guarantee routine for aperiodic new task to check its schedulability with already guaranteed system's tasks. Similarly, the select PC checks the schedulability when the new arrived task is periodic. When the new task fail in the test of schedulability then the selected PC will act as a local PC and start to find another PC in the same way as explained above.

Neurofuzzy in the decision-maker algorithm: The theory of fuzzy logic provides a formal framework to abstract the approximate reasoning characteristics of human decision making and furthermore, conveys an excellent mode of knowledge representation. Neural networks, on the other hand, attempt to replicate the learning capabilities possessed by biological-species, but it is not always possible to extract and interpret the learned knowledge contained within them. However, the use of neural network (NN) offers the following main useful properties such as, parallelism, nonlinearity, generalization, adaptivity, fault tolerance and VLSI implmentation[33-37].

Because of these benefits of NNs and of fuzzy logic, the possibility of integration them has given rise to a rapidly emerging field of neurofuzzy networks that are intended to capture the capabilities and advantages of both neural and fuzzy logic systems.

There are various approaches to combine neural and fuzzy system. These approaches can be classified into Structural and Functional approaches, according to the sought mapping between a fuzzy inference system and an NN[36,37].

Structural neurofuzzy approach: In this study, one is looking for a structure mapping from a fuzzy reasoning system to an NN, resulting in a localized implementation.

Functional neurofuzzy approach: In this study, fuzzy rules are represented in a distributed fashion by the connection weights and local processing elements in the network.

In this study the same Decision-maker algorithm, which was built using fuzzy logic will be performed using Neurofuzzy approach in two manners.

Fig. 3: The proposed structural neurofuzzy decision-maker algorithm.

Structural neurofuzzy approach: The aim of using structural neurofuzzy is to find the correct fuzzy select action. The correct fuzzy select actions are progressively learned by operating the system repeatedly. The block diagram of the proposed system is shown in Fig. 3 The overall system is composed of three components:

- Fuzzifier to convert the scaled input into its fuzzified values in the predefined Universe Of Discourse (U.O.D) using the triangular equation form.
- Back-propagation Neural Network (BNN) Knowledge base representing the fixed Fuzzy Production Rules (FPRs).
- Center of gravity defuzzification approach to map the fuzzy select action vector ($\overline{U}$) into a crisp select value (U).

The variables shown in Fig. 3 are defined as follows:

x = Available Time (AT) or number of APeriodic Tasks (APT).
y = Scheduling Cost (SC) or number of Periodic Tasks (PT).
X = Scaled x.
Y = Scaled y.
$\overline{\mu_x}$ = Vector Membership of input x.
μ_y = Vector Membership of input y.

A given twenty five FPRs for periodic new task and for aperiodic new task can be used in the learning process of the BNN. The already defined five fuzzy sets for both X and Y will give five memberships for each one, these are $\overline{\mu_x}$ and $\overline{\mu_y}$, respectively which are the output of fuzzifier and the input to the FPRs BNN.

Figure 4 shows a BNN with 10 nodes input layer receiving the membership functions related to X and Y, a 21 nodes hidden layer (many trails were made to get this suitable size) and output layer of 13 nodes (number of intervals found in the universe of discourse). The 13 nodes of output give the select action vector $\overline{U}$ for each input pattern.

Functional neurofuzzy approach: There are two points must be taken into consideration when using a Structural Neurofuzzy approach in the Decision-maker algorithm that shown in Fig. 4, these are:

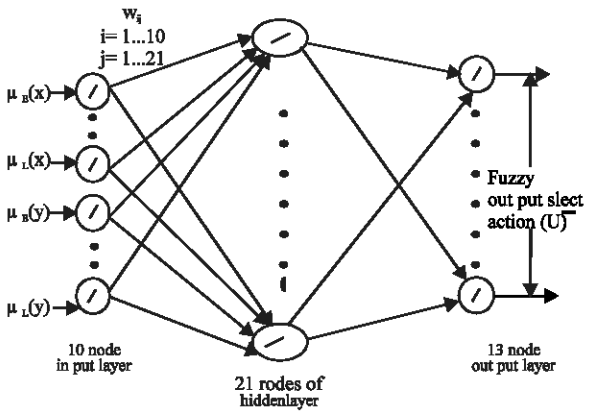

Fig. 4: BNN used to learn 25 FPRs for periodic new task during 1370 iterations with $\eta = 0.3$, $\alpha = 0.8$, $\theta_o = 1$ and $E_{total} = 0.01$ and for aperiodic new task during 2698 iterations with $\eta = 0.3$, $\alpha = 0.8$, $\theta_o = 1$ and $E_{total} = 0.009$.

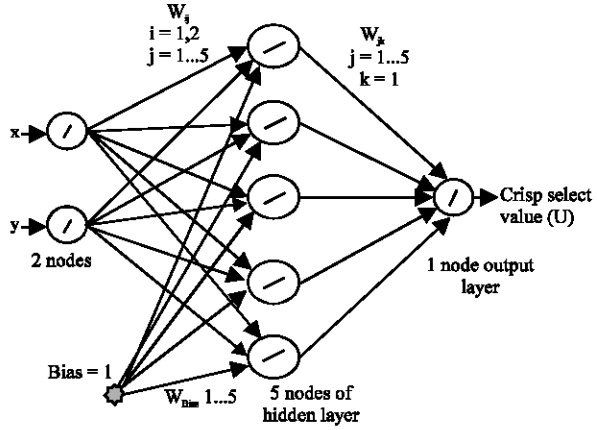

Fig. 5: BNN used to learn 169 pattern for periodic new task during 17675 iterations with $\eta = 0.3$, $\alpha = 0.8$, $\theta_o = 1$ and $E_{total} = 0.5$ and for aperiodic new task during 18850 iterations with $\eta = 0.3$, $\alpha = 0.8$, $\theta_o = 1$ and $E_{total} = 0.5$.

- Huge architecture is required for real-time implementation.
- It is too difficult to be implemented in a hardware form if needed.

To overcome the above drawbacks, the Functional Neurofuzzy approach is used with new idea of implementation. Because of using a fixed range of inputs [0-10] to the fuzzifier part used to implement the fuzzy decision-maker algorithm, then all possibility of the input values can be known before running the system. Now if all possibilities are known in advance, then the corresponding possibility of crisp output (U) can be calculated in advance and OFF-LINE.

Referring to the proposed system, the number of intervals which are used are equal to 13 intervals. So, the possibility of any input value will be in one of these intervals. Also, because of the number of inputs is equal to two (X and Y) then the number of all possibilities are 13×13 (i.e., 169).

When performing all these possibilities of inputs to the Decision-maker algorithm implemented by fuzzy rules then 169 crisp output values could be obtained corresponding to them OFF-LINE. The BNN shown in Fig.5 can be used to learn about these inputs as well as their corresponding output OFF-LINE. The use of this taught neural net ON-LINE will speed up the selection of the better offer (select PC) in addition to the collection of advantages obtained using fuzzy and neural at the same time.

It is clear from Fig. 5 that the BNN consists of:

- 2 nodes input layer that receive the two inputs x and y after scaling them in the range of [0-10]. The inputs x and y represent the Available Time and Scheduling Cost respectively when the new arrived task is aperiodic. On the other hand, they represent Number of APeriodic tasks and Number of Periodic tasks respectively if the new arrived task is periodic.
- 5 nodes hidden layer with a bias equal to 1 (many trails were made to get this suitable size).
- Output layer of node 1, which gives the crisp output value (U) for each input pattern.

The mathematical approach: While trying to use a neural net in implementing the decision algorithm an idea to put a threshold for each layer in their neurons, as those found in the transmission scheduling, was raised. Then another idea was introduced, if the conditions for thresholds are found then why not use it directly without neural net?.

So the implementation of these ideas was started and the beginning point was to find the conditions to make the correct selection, in the way a mathematical approach was introduced to make the required decision without needing to fuzzy, neurofuzzy, genetic or neural.

The description of the mathematical approach for aperiodic new task is shown below:

- Find maximum Available Time (AT) from all offers as follows:
  Max. AT = max (AT_0,…, AT_{n-1})
  Where: n= numbers of PCs which offer bids.
- Find maximum Scheduling Cost (SC) from all offers as follows:
  Max. SC = max (SC_0,…, SC_{n-1})

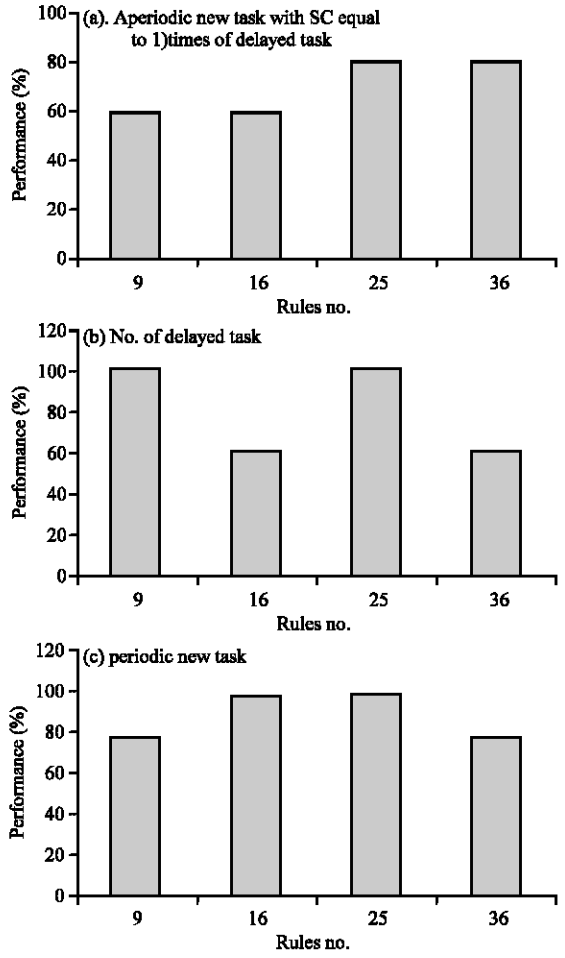

Fig.6: The Performance of the fuzzy decision algorithm with different numbers of rules

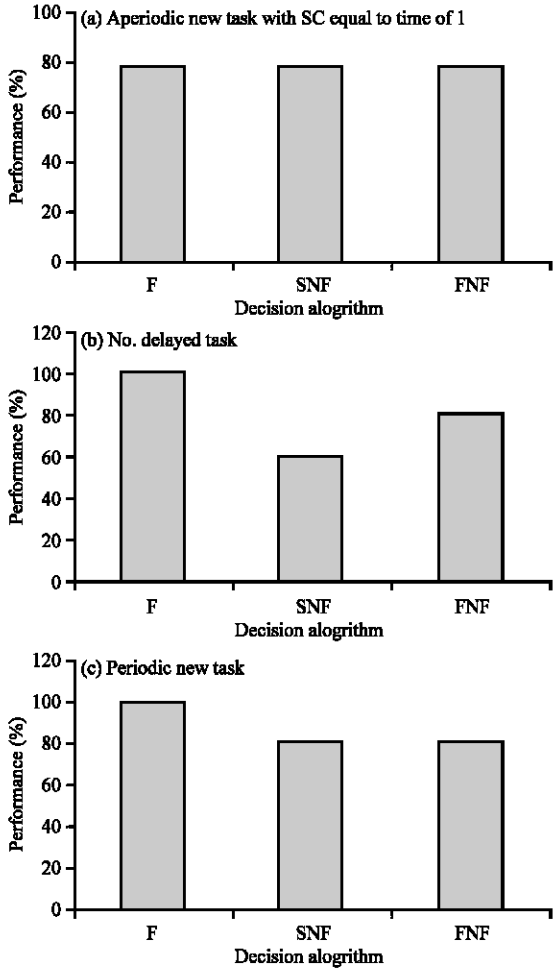

Fig.7: The Performance of the decision algorithm with different approaches (F = 25 fuzzy rules, SNF = Structural Neurofuzzy and FNF = Functional Neurofuzzy)

- Find a new value for each SC, let it be called Complement Offer (CO), as follows:
  CO_I= Max. SC - SC_I
  Where: i= 0, 1,…,n-1
- Find maximum Complement Offer from all offers as follows:
  Max. CO = max (CO_0,…, CO_{n-1})
- Find the best anticipation offer (B):B= Max. AT+Max. CO
- Find the competing number (CN) for each offered PC as follows: CN_I = AT_I+CO_I
- The selected PC, is that node which has CN_I closest value to B.

The description of the mathematical approach for periodic new task is shown below:

- Find minimum periodic number (PT) from all offers:
  Min. PT = Min (PT_0, …, PT_{n-1})
- Find minimum aperiodic number (AP) from all offers:
  Min. APT = Min (APT_0,…, APT_{n-1})
- Find the competing number (CN) for each offered PC:
  CN_I = PT_I+APT_i
- Find the best anticipation offer (B):
  B = Min. PT+Min. APT
- The selected PC, is that node which has CN_I closest value to B.

Comparison between the mathematical approach and the others: The mathematical approach has the following specification:

- It needs little and simple calculation to find the better offer received from PCs.
- It is faster in finding its selection than the fuzzy and neurofuzzy approaches.

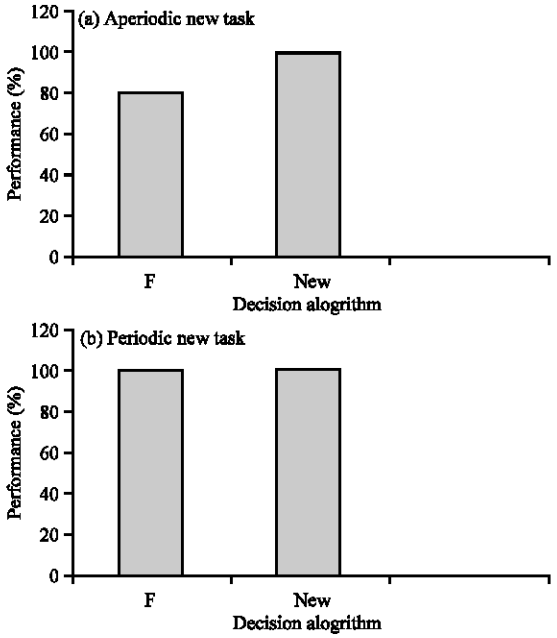

Fig.8: The Performance of 25 Rules fuzzy decision algorithm in comparison with the mathematical approach.

- It needs no temporary stored data that is necessary in calculation to find the best offer as in the case of neurofuzzy systems.
- In the mathematical approach, there is a high possibility to have more than one best offers (equals to CN).
- The experience of the designer and the flexibility to choose some parameter can be added to the fuzzy and neurofuzzy facilities when needed to implement the decision algorithm, while the mathematical approach is a mathematical solution only and the final decision are not affected by the conditions above.

Performance evaluation: After running a twenty different case of systems for each methods used to implement the decision algorithm (Fuzzy with different number of rules, Functional Neurofuzzy, Structural Neurofuzzy and the Mathematical approach), the final results is obtained as shown in Fig. 6-8.

The results represented by performance of the decision algorithm are related to the percentage of the better choice accuracy.

CONCLUSION

Based on the results obtained in the previous chapter some conclusions can be deduced as follows:

- Using the method based on slotted task guarantee routine, the execution time of the new task must be increased by the factor N (where N = time need for dispatcher×number of slots of new task). For worst case, the number of slots is equal to the execution time of a new task. While when non-slotted task guarantee routine is used the time needed by dispatcher added to the execution time of the new task one time only. On the other hand, using slotted task guarantee routine will give more chance to the new task to be accepted compared with non-slotted task guarantee routine.
- From the results obtained when implementing the decision algorithm by fuzzy rules Fig. 6, it can be seen that there is not much difference between using different number of rules but for better decision, the 25 rules are preferred. In addition, the results show that when using odd number for rules (9 and 25 Rules) the decision is better than using even number of them (16 and 36 Rules).
- The neurofuzzy net (Structural and Functional) can make a decision after learning about how the fuzzy rules work. So it has its own decision, which has approximately the same decisions as that obtained by fuzzy approach but on the whole is acceptable.
- It is better to choose the functional neurofuzzy approach over the structural neurofuzzy and fuzzy sets because of the following reasons:
  - It is a very small net in comparison with structural neurofuzzy and consequently needs a very small size of memory.
  - A small size of net has a very high speed to obtain the selected node number in comparison with the structural neurofuzzy which has a huge number of nodes and in comparison with Fuzzy approach which makes all it's calculations ON-LINE.
  - The results show that the decision obtained from functional neurofuzzy is the best.
- The mathematical approach that is used in this work to implement the decision algorithm, need little and simple calculations and needs no stored data for making its decision.
- The designed system is real-time distributed system, so the speed to reach a decision must be very high i.e. very short time. The mathematical approach is faster in finding its selection than the fuzzy and neurofuzzy approaches as the results of simulation show.
- The mathematical approach is a mathematical solution and not affected by the conditions that supports the Fuzzy and Neurofuzzy algorithms to get the best decision.

REFERENCES

1. Anderew, S.T., 1995. Distributed Operating Systems, Prentice Hall.

2. Abraham, S. and B.G. Peter, 1998. Operating Systems Concept, Addison-Wesley Publishing Company.
3. James, E.G. and T.R. Phillip, 2000. Local Area Networks, John Weley and Sons, Inc.
4. William, B., 1999. Mastering networks, Macmillan Press Ltd.
5. Jean-Dominique, 1993. A survey on industrial communication network, ANN. Telecommunication, pp: 9-10.
6. Henneth, R., 1998. Data Network Handbook, Galgotia Publications (P) Ltd.
7. Ali, U.M., 1999. Fieldbus scheduling using neural networks, Ph.D. Dissertation, Control and Computer Engineering Department, University of Technology, Baghdad, Iraq.
8. Andrew, B., 1996. An enabling technology for fundamental change, IEE Fieldbus Supplement.
9. Michael, L., 1996. Go with the right bus, Fieldbus Supplement.
10. Juan, R., 1989. Communication architectures for fieldbus networks, Control Engineering, pp: 74-78.
11. Alan, R., 1989. Fieldbus moves up another gear, C and I, pp: 135-139.
12. Alan, R., 1988. Final stop for industrial fieldbus, C and I.
13. Matt, K., 1987. Fieldbus: New Standard, Control Engineering, 2nd October.
14. Gialuca, C., D. Luca and V. Adriano, 1995. Standard fieldbus networks for industrial applications, Computer Standards and Interfaces, pp: 155-167.
15. Chin- Che, C. and G.S. Kang, 1997. Statistical real-time channels on multi-access bus networks, IEEE Transactions on Parallel and distributed systems.
16. Salvatore, C., 1996. Exploiting data link layer features to realize communication models in fieldbus system, Computer Standards and Interfaces, pp: 139-158.
17. Salvatore, C., S. Antonella, Di and M. Orazio, 1993. Optimization of a cyclic bandwidth allocation exploiting the priority mechanism in the fieldbus data link layer, IEEE Transaction on Industrial electronics, pp: 297-306.
18. Al- Isterbadi, K.I.K, 2000. A proposal for a fieldbus protocol standard, M.Sc. Thesis, communication and Electronic Engineering Department, College of Engineering, University of Baghdad, Baghdad, Iraq.
19. Salvatore, C., S. Antonella, Di and M. Orazio, 1994. Mapping application layer functionality's to DLL mechanisms in process control environment, IEEE Proceeding IECON '94.
20. Gonzalo, U., 1991. Fieldbus Application Layer and Real-Time Distributed Systems, IEEE Proceeding IECON '91, pp: 1679-1683.
21. Salvatore, C., S. Antonella, Di and M. Orazio, 1995. Pre- run- time scheduling to reduce schedule length in the fieldbus environment, IEEE Transaction Software Eng., pp: 865-887.
22. Kang, G.S. and C. Chih-Che, 1996. Design and Evaluation of real- time communication for fieldbus based manufacturing systems, IEEE Transactions on Robotics and Automation.
23. Kunert, O. and M. Zitterbart, 1997. Interconnecting Fieldbuses Through ATM, Proceedings of the 22nd IEEE Conference on Local Computer Networks.
24. Paulo, V., 1997. How Hard Is Hard Real- Time Communication on Field-buses?, Proceedings of the 27th IEEE International Symposium on Fault-tolerant computing.
25. Hindo, B.K.B., 2000. Implementation of multi- master fieldbus processor, M.Sc. Thesis, Control and Computer Engineering Department, University of Technology, Baghdad, Iraq.
26. Kang, G.S. and C. YI-Chieh, 1989. Load sharing in distributed real-time systems with state-change broadcasts, IEEE Transactions on Computers.
27. Krithivasan, R., A.S. Jhon and Wei Zhao, 1989. Distributed scheduling of tasks with deadlines and resource requirements, IEEE Transactions on Computers.
28. John, A.S., R. Krithivasan and C. Shengchang, 1985. Evaluation of flexible task scheduling algorithm for distributed hard real-time systems, IEEE Transactions on Computer.
29. Marin, L., C.I. Traian and L. Jesus, 1998. Dynamic task scheduling in distributed real-time systems using fuzzy rules, Microprocessors and Microsystems, pp: 299-311.
30. Krithivasan, R. and A.S. Jhon, 1984. Dynamic task scheduling in hard real-time distributed systems, IEEE Software, pp: 65-75.
31. John, A. S., 1989. Decentralized decision making for task reallocation in a hard real-time system, IEEE Transaction on Computers.
32. Cherkassk, V. and F. Mulier, 1998. Learning from data, Wiley- Inter Science Population.
33. Kung, S.Y., 1993. Digital neural networks, PTR Prentice Hall.
34. Hunt, K.J., D. Sbarbaro, R. Zbikowski and P.J. Gawthrop, 1992. Neural networks for control system Automatica.
35. Haykin, S., 1994. Neural networks, A comprehensive foundation, Macmillan College Publishing Company Inc., UK.
36. Grossberg, S., 1988. Neural networks Principles, Mechanisms Architectures, Neural Networks, pp: 17-61.
37. Ali, M.M., 1998. Neurofuzzy controller design, Ph.D. Dissertation, Control and Computer Engineering department, University of Technology, Baghdad, Iraq.