

**NEURAL NETWORK FOR SCHEDULING PERIODIC  
INFORMATION FLOW IN FIELDBUS  
COMMUNICATION SYSTEMS**

*A*

*Dissertation submitted to*  
JAWAHARLAL NEHRU UNIVERSITY, New Delhi  
in partial fulfillment of the requirements  
for the award of the degree of

**Master of Technology  
in  
Computer Science & Technology**

**By  
MANGALA PRASAD MISHRA**

Under the Guidance of  
**Prof. P.C. Saxena**  
&  
**Prof. C. P. Katti**



**जवाहरलाल नेहरू विश्वविद्यालय**  
**SCHOOL OF COMPUTER & SYSTEMS SCIENCES**  
**JAWAHARLAL NEHRU UNIVERSITY**  
NEW DELHI 110 067 INDIA

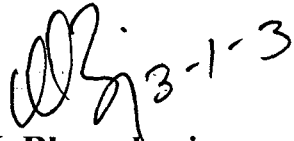


जवाहरलाल नेहरू विश्वविद्यालय  
SCHOOL OF COMPUTER & SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI 110 067 INDIA

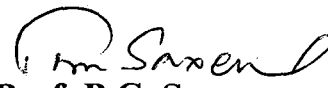
**CERTIFICATE**

This is to certify that the dissertation entitled “Neural Network for Scheduling Periodic Information Flow in Fieldbus Communication Systems” which is being submitted by **Mr. Mangala Prasad Mishra** to the School of Computer & Systems Sciences, JAWAHARLAL NEHRU UNIVERSITY, NEW DELHI for the award of **Master of Technology in Computer Science & Technology** is a bonafide work carried out by him under the supervision of Prof. P.C.Saxena and Prof. C.P.Katti.

This is an authentic work and has not been submitted in part or full to any university or institution for award of any Degree.


 3-1-3

**Prof. K.K. Bharadwaj**  
Dean  
SC & SS, JNU



**Prof. P.C. Saxena**  
Supervisor  
SC & SS, JNU

**Professor K. K. Bharadwaj**  
Dean  
School of Computer & Systems Sciences  
Jawaharlal Nehru University  
New Delhi - 110067

  
**Prof. C.P. Katti**  
Supervisor  
SC & SS, JNU.

## **ACKNOWLEDGEMENT**

I wish to convey my heartfelt gratitude and sincere acknowledgements to my Supervisors Prof. P. C. Saxena and Prof C. P. Katti for his constant encouragement, guidance and affection throughout my work. I am grateful to him for providing me enough infrastructures, through Data Communication and Distributed Computing Group (DCDCG) laboratory to carry out my work. It is great honor for me for being a member of DCDC Group and I am thankful to all the members for their cooperation and understanding.

I would like to thank all my faculty members for their help and useful suggestions during the period of my course work. I am also thankful to all of my classmates for their constructive criticism and useful suggestions.

In last but not least I am thankful to my parents and especially to my wife Mrs. Seema Mishra, without their support it was not possible to reach at the stage of successful completion of this project.

**Mangala Prasad Mishra**

## ABSTRACT

---

Scheduling problem occurs in several application areas where different activities have to be sequentially ordered to perform a pre-established task. A scheduling problem which frequently occurs is to assign a single resource to several users. This kind of scheduling take place in a common-bus computer network.

Scheduling, access to the physical channel in computer network is an important problem in applications such as process control. Process controls are often very critical and delays in delivery may lead to errors in the synchronization between cooperating processes. In common-bus communication networks Fieldbuses are used to connect sensors and control devices to create regulation loops.

The key idea behind this Dissertation is to study the scheduling problem in Single Fieldbus Communication Systems. In this work we have done scheduling of processes with the help of scheduling table containing the transmission instants of the information produced by the processes connected to the communication system. We implemented ERMM, a static priority based scheduling algorithm to schedule the information flow in communication network. We have done analytical study of Hopfieldd Neural Network to be used in finding the optimized solution of the scheduling problem. Lastly we have done comparative study between the performances of classical scheduling solution and scheduling solution emerged by using Hopfield neural network.

# CONTENTS

<u>Chapters</u>	<u>Page No<sub>s</sub>.</u>
1. INTRODUCTION	1
1.1 Fieldbus	1
1.2 Short History of Fieldbus	2
1.3 Fieldbus Standard	3
1.4 Scheduling in computer network	3
1.5 Artificial Neural Networks and process scheduling	5
1.6 Outline of this study	6
2. FIELDBUS COMMUNICATION SYSTEMS	7
2.1 Fieldbus Classification	7
2.2 Fieldbus Architecture	9
2.2.1 ISO/OSI Reference Model	9
2.2.2 OSI Layers for Fieldbus	10
2.2.3 Comparison Between ISO/OSI and Fieldbus Models	12
2.3 Advantages of Fieldbus	14
3. NEURAL NETWORK TECHNOLOGY	15
3.1 What are Artificial Neural Networks?	15
3.1.1 Analogy to Brain	15
3.1.2 Artificial Neurons and their Working Process	16

3.1.3 Artificial Neural Network Operations	18
3.2 Training Artificial Neural Networks	20
3.2.1 Supervised Learning	20
3.2.2 Unsupervised Learning	21
3.2.3 Learning Rates	21
3.2.4 Learning Laws	22
3.3 Major Components of an Artificial Neuron	23
3.4 Neural Network Selection	26
3.5 Uses of Neural Network	27
3.5.1 Language Processing	27
3.5.2 Signal Processing	28
3.5.3 Character Recognition	28
3.5.4 Image (data) Compression	28
3.5.5 Financial	29
3.6 Neural Network Present and Future	29
3.6.1 Development Systems	30
3.6.2 Hardware Accelerators	30
3.6.3 Dedicated Neural Processor	30
3.6.4 Next Developments Bill Be	31
<b>4. HOPFIELD NETWORK IN PROCESS SCHEDULING IN FIELDBUS COMMUNICATION SYSTEMS</b>	<b>33</b>
4.1 Process Scheduling	33
4.2 Scheduling in a Single Fieldbus Environment	34
4.3 Monocycle Polling Scheduling	37
4.4 Multicycle Polling Scheduling	37
4.5 Extended Rate Multicycle Polling Scheduling	38
4.6 Determination of Scheduling Sequence	41

4.7 Hopfield Neural Network and Scheduling	
Problem in Single Fieldbus Systems	42
4.8 Definition of the Neural Architecture	43
4.9 Neurons Specification	44
4.10 Specification of the surrounding Conditions of the	
The Scheduling Problem to be solved	45
4.10.1 Correctness of the Scheduling of Processes	45
4.10.2 Formalization of Surrounding Conditions	46
4.10.3 Number of Transmissions	47
4.10.4 Correctness of Scheduling of a Single Process	47
4.11 Determination of Weigh and Biases	48
4.12 Tuning of Neural Model	49
4.13 Eample of Neural Scheduling	51
4.14 Neural Versus Classical Scheduling	52
4.14.1 Programming Complexity	52
4.14.2 Computational Complexity	54
4.14.3 Comparison between Computational	
and Programming Complexity	55
CONCLUSION	56
REFERENCES	57

# CHAPTER 1

## INTRODUCTION

---

Scheduling problem occur in our daily life, when we perform different activities by giving suitable time slot to do these activities. Scheduling problems often occur in several applications in computer science like CPU scheduling, Process Scheduling, Disk Scheduling, and Scheduling of resources in Communication Network. Whenever we need to perform different sequential pre-established activities scheduling comes into existence. For example if we have to assign a single resource to several users, in this case the main aims are to maximize the use of resources and satisfy the requirements of the various users.

### 1.1 Fieldbus

Fieldbus is a generic-term, which describes a new digital communications network that is to replace the existing 4 - 20mA analogue signal. Fieldbus Networks are a special form of local area network dedicated to applications in the field of data possession and the control of sensors and other control devices. Fieldbus Networks typically operate on low cost twisted pair cables. The network is a digital, bi-directional, multidrop, serial-bus, communications network used to link isolated field devices, such as controllers and sensors. Each device will be able to execute simple functions on it's own such as diagnostic, control, and maintenance functions as well as providing bi-directional communication capabilities. These devices make able to communicate with other field devices. Fieldbus will replace centralized control networks with distributed-control



networks. Therefore field bus is much more than a replacement for the 4 - 20mA analogue standard.

With fieldbus technology it can be promised to improve quality, reduce costs and increase in efficiency. These promises made by the fieldbus technology are achieved partly from the fact that information, which a field device is required to transmit or receive, can be transmitted digitally. This is a great deal more accurate than transmitting using analogue methods, which were used previously. Each field device is also a 'smart' device and can carry out it's own control, maintenance and diagnostic functions. As a result it can report if there is a failure of the device or manual intervention is required, this increases the efficiency of the system.

## **1.2 Short History of Fieldbus**

In the 1960's, the 4-20 mA analogue signal standard was introduced for instrumentation. Despite this standard, various signal levels were used to suit many instruments, which were not designed to the standards specification.

The development of digital processors in the 1970's gave rise to the use of computers to monitor and control a system of instruments from a central point. The computers have been used in automation earlier also but they became common not until the *Distributed Control Systems* (DCS) became commercially available in the late 1970's. These DCS's were based on the independent, task-oriented microcomputers. DCS's were also connected to each other by a digital communication link. . In the 1980's smart sensors began to be developed and implemented in a digital control, microprocessor environment. This raise the need to integrate the various types of digital instrumentation into field networks to optimize system performance. While the "if it works then use it" mentality progressed, it became necessary that a fieldbus standard was required to formalize the control of smart instruments.

### **1.3 Fieldbus Standard**

The decision to provide an international standard is in the hands of the Instrument Society of America (ISA), the International Electrotechnical Commission (IEC), Profibus (German national standard) and FIP (French national standard). They form the IEC/ISA SP50 Fieldbus committee. The committee decided the standard to be developed must include issues like

- a. It must integrate the enormous range of control instruments.
- b. Provide them with interfaces to operate various devices simultaneously.
- c. Set a communication protocol to support them all.

The SP50 committee decided to concentrate on four layers for the fieldbus solution:

- Physical Layer: This defines the media that communication occurs over and could be viewed as the 4 - 20mA standard replacement.
- Data Link Layer: This monitors the communications-taking place among the various devices and detects errors.
- Application Layer: This formats the data into messages, which all devices connected, to the network can understand and provides the services for process control, supplying them to the user layer.
- User Layer: This connects the individual plant areas and provides an environment for applications. It is implemented using high-level control functions.

### **1.4 Scheduling in computer network**

Scheduling of physical channel in a common-bus computer network is a very important problem in applications such as process control. These applications are very critical delay in delivery in may lead to errors in the synchronization between cooperating processes. If there id delay and deadline are not met, it may cause a serious̄ problem in

system operation, which may lead to wrong result or a disaster. Common-bus communication systems Fieldbusses are used to connect sensors, control devices and actuators with regulation to create regulation loop.

Fieldbuses are characterized by centralized access control, a Particular Master station manage access to the physical channel by sending various stations authorization for the transmission of information. Figure 1.1 given below shows the Fieldbus communication model.

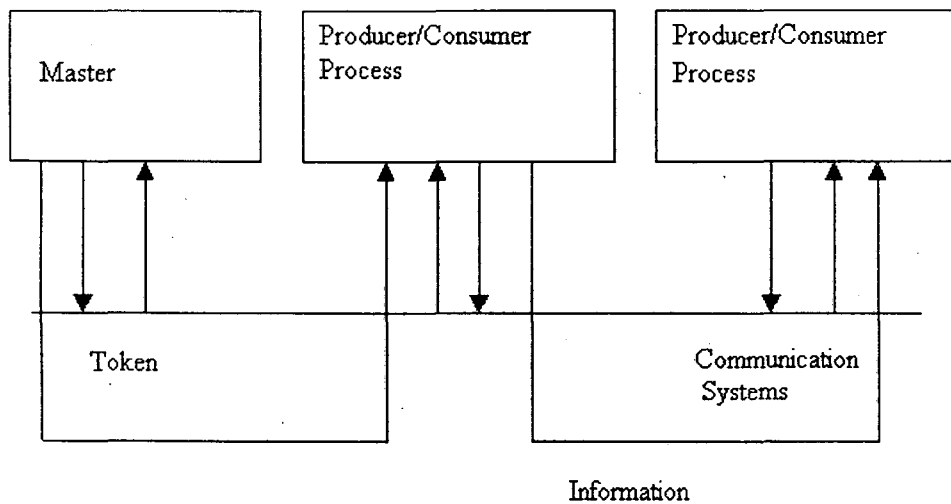


Figure 1.1 Producer Consumer Systems

Here Master station uses a scheduling table containing the transmission instants of the information produced by the processes connected to the communication system. The transmission instants are determined in such a way that the time constraints of various control processes are met. By using the table information, the Master authorizes the various transmissions contained in it.

Sometime the size of the scheduling table may be extremely large. It demands large size of memory to store it. This create a problem in control systems in which simple devices, with little memory are used.

## 1.5 Artificial Neural Networks

Today ANNs are being applied to an increasing number of real- world problems of considerable complexity. They are good pattern recognition engines and robust classifiers, with the ability to generalize in making decisions about inaccurate input data. They offer ideal solutions to a variety of classification problems such as speech, character and signal recognition, as well as functional prediction and system modeling where the physical processes are not understood or are highly complex.

ANNs may also be applied to control problems, where the input variables are measurements used to drive an output actuator or to handle some control device , and the network learns the control function. The advantage of ANNs lies in their resilience against distortions in the input data and their capability of learning. They are often good at solving problems that are too complex for conventional technologies.

There are multitudes of different types of ANNs. Some of the more popular include the multilayer perceptron which is generally trained with the back propagation of error algorithm, learning vector quantization, radial basis function, Hopfield neural model which is very suitable for solving optimization problems.

Hopfield network was first used to solve the well-known Traveling Salesman Problem (TSM). Later it was used to solve number of optimization problems including, An A/D Converter, Path Determination in Graph and Shortest Path Computation and Routing in Computer Networks.

For scheduling the information flow (Process scheduling) in Fieldbus communication systems, scheduling methodology based on a Hopfield neural model can be developed to get optimized solution (to get a reduced scheduling table).

## **1.6 Outline of This Study**

In this report we have discussed first in chapter 2, the Fieldbus classification which explain about different type of Fieldbuses , Fieldbus architecture and its comparison with ISO/OSI model and in last advantages of Fieldbuses. In chapter 3 we have discussed , ANN and its analogy to brain ,how ANN works ,what are ANN operations. In this chapter we have discussed also the training of ANN, which include supervised & unsupervised learning, learning rates. A brief history of ANN, uses of ANN are also given in this chapter .In chapter 4 we have explained the implementation of a priority based Fieldbus scheduling known as ERMM it is an static scheduling algorithm. In this chapter we have done analytical study of proposed Neural Network solution to the scheduling problem. At the end of this chapter we have done comparative study of performance analysis of both classical and neural network solution to the scheduling problem. In the last we have concluded the work done in previous chapters.

## CHAPTER 2

# FIELDBUS COMMUNICATION SYSTEMS

---

### 2.1 Fieldbus Classification

Different kind of fieldbuses can be classified into three classes according to their main purposes. These classes are given below and they can be visualize in figure 2.1.

1. Fieldbus,
2. Devicebus and
3. Sensorbus.

**Fieldbus** is a bus that is basically planned to use for process automation. Fieldbus is also to connect intelligent field devices. The communication is based on data blocks with varying length. It has all the elements needed in distributed automation applications. Function blocks can make distributed applications. These function blocks are flexible and easy to use and configure without traditional programming.

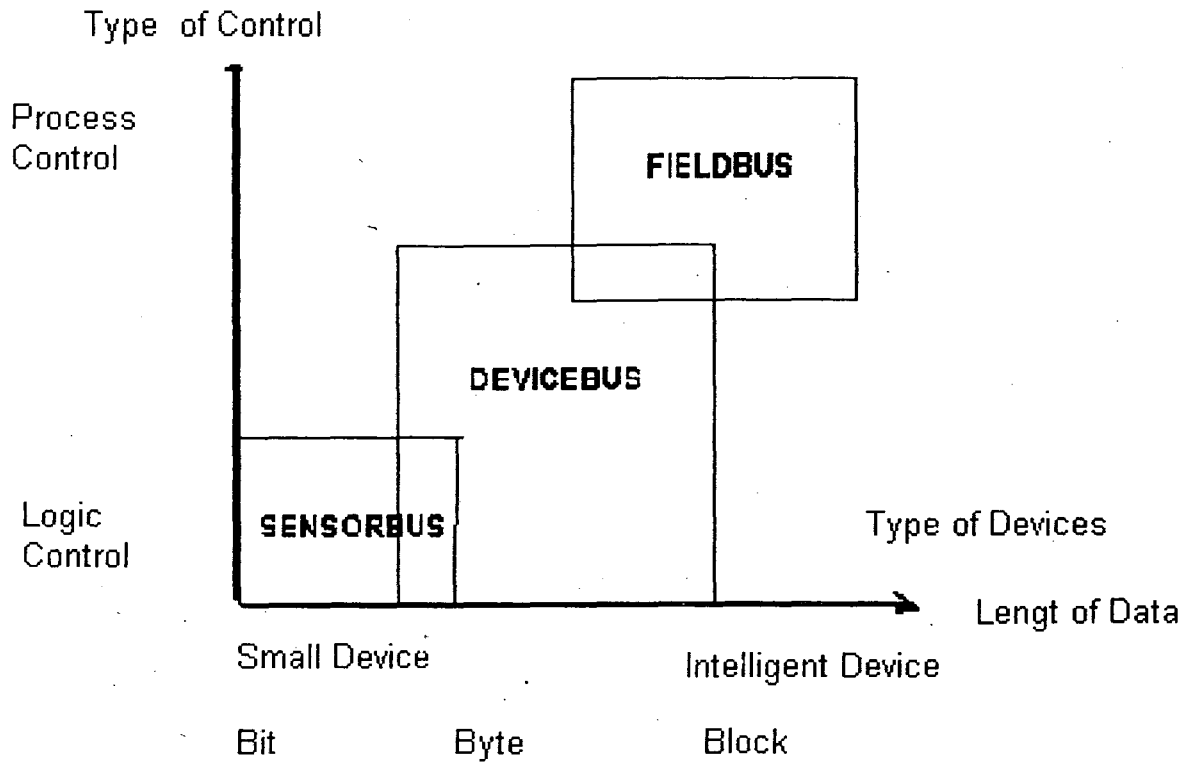


Figure 2.1 A Classification of different fieldbuses

**Devicebus** is usually used in factory automation. It is a fast bus that allows "**high efficiency communication for small data package**," as it is defined in Euro Norm 50254. Typically the communication is taking place in byte-level.

**Sensorbus** means a bus that is used for binary devices with a bit-level communication. It is useful when an application contains a lot of binary sensors and actuators.

Now we will discuss about the architecture of Fieldbus communication system.

## **2.2 Fieldbus Architecture**

The specification of a Fieldbus ideally covers all of the seven layers of the ISO/OSI Reference Model but only layers 1, 2 and 7 are used by Fieldbus. Application in the Fieldbus is the functionality provided by the function blocks.

### **2.2.1 OIS/OSI Reference Model**

OSI model is a seven-layer architecture; this architecture looks like in figure 2.2 which is given below. ISO/OSI Model. The function of this Model is to separate device to device interaction for devices exchanging information on a network. An outline of how these 7 layers of OSI model should be covered is as follows.

**Layer 7: Application Layer** → It is the entry point for an application to send data across a network. Functions of this layer include flow control, network access, and some functions of error recovery.

**Layer 6: Presentation Layer** → This layer is responsible for translation of data into a generic format, compression, data translation, encryption/decryption.

**Layer 5: Session Layer** → This layer is responsible for establishing communication between two computers, taking care of packet size, and timing for transmissions.

**Layer 4: Transport Layer** → This layer provides additional error checking and assures error transmissions between client and server.



## OSI Reference Model

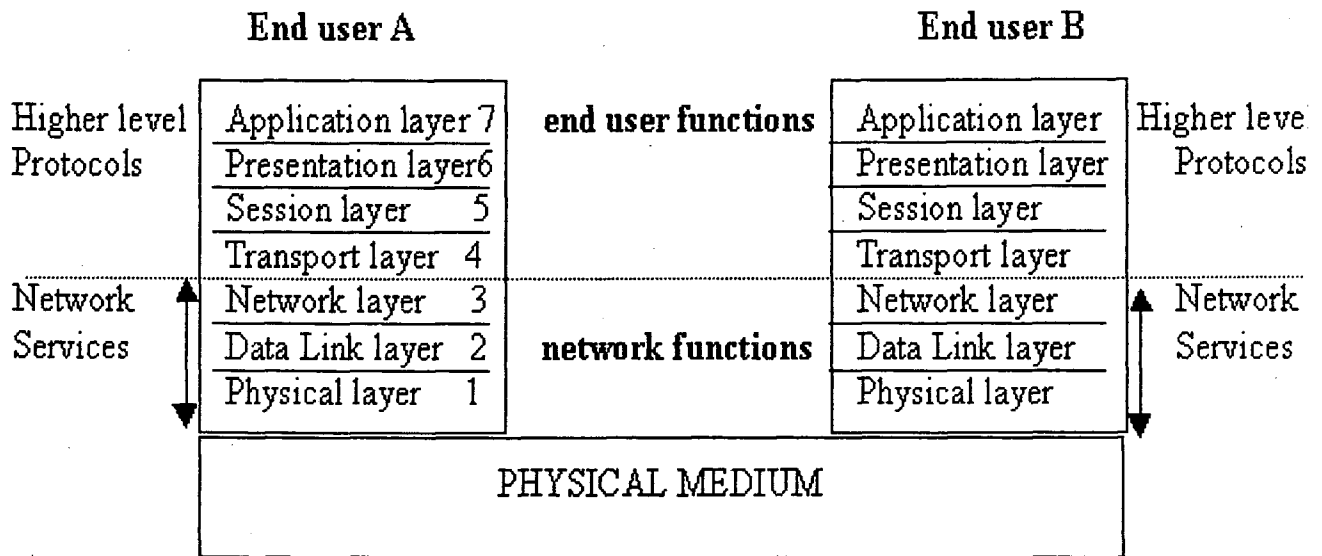


Figure 2.2

**Layer 3: Network Layer** → This layer determines paths to the destination computer, addresses messages, and translates logical address.

**Layer 2: Data Link Layer** → It Creates a data frame for the data to be sent across the network and packages this in a usable format for the physical layer (bitstream).

**Layer 1: Physical Layer** → It allows data to be transmitted across the wiring medium. Interface between the computer and the cable function at this level. (ISDN Smart Cards, PADS, and ATM cards).

### 2.2.2 OSI Layers for Fieldbuses

Here we will discuss about OSI layer in Fieldbus communication system shown in figure 2.3 which is given below. . An outline of how these 7 layers of Fieldbus model be covered is as follows.

**1:Physical Layer** → It deals with the types of signals present, levels, representation of 1's and 0's, the type of media, connects etc.

**2:Data Link Layer** → It deals with the techniques for establishing links between two communicating parties(end users).

**3:Network Layer** → It deals with the method of selecting the node of interest and method of routing data.

**4:Transport Layer** → It ensuring what was sent arrives at the receiver and corrects any correctable problems.

### Fieldbus Model

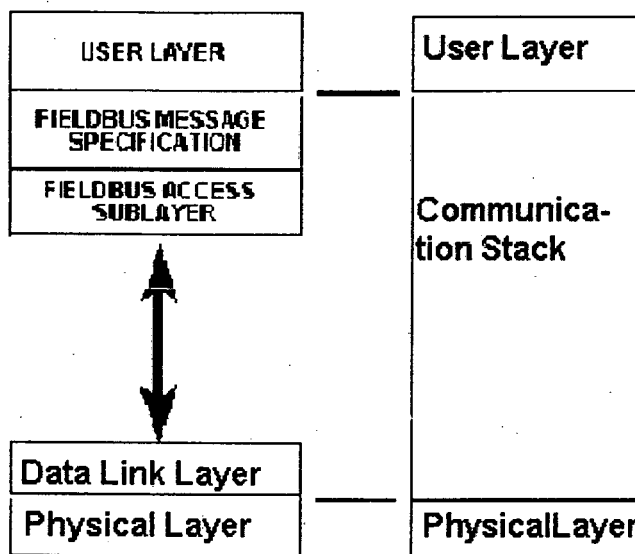


Figure 2.3

**5:Session Layer** → It is not applicable to Fieldbuses.

**6: Presentation Layer** → It is not applicable to Fieldbuses.

**7:Application Layer** → Meaning of data.

### 2.2.3 Comparison Between ISO/OSI and Fieldbus Models

On the basis of previous discussion we can compare the ISO/OSI model with Fieldbus model. In figure 2.4 Fieldbus protocol model is compared to the 7-layer OSI-communications protocol model. The observation, which is to be made, is that OSI does not define a user application layer, while Fieldbus does.

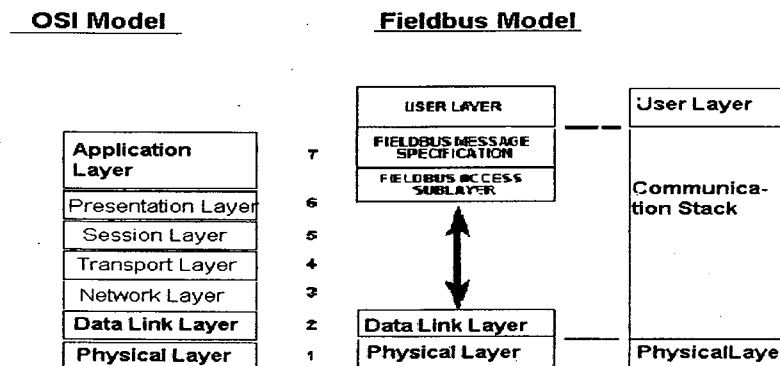


Figure 2.4 OSI and Fieldbus Models

ISO/OSI Application Layer is comparable to combination of Fieldbus Access Sublayer and Fieldbus Message Specification.

Functionality of Fieldbus layers are discussed below.

#### Fieldbus Access Sublayer

Fieldbus Access Sublayer (FAS) provides interfaces to its service users and the Data Link Layer. Application Processes in a distributed communication system use a set of services and logical communication channels of the Application Layer to communicate with each other. In the Fieldbus environment, the FAS provide such communication

channels, which are called Application Relationships between these Application Processes.

### **Fieldbus Message Specification**

Fieldbus Message Specification (FMS) is based on a series of object oriented models. FMS provide distributed applications with the capabilities and services necessary for their implementation in an open, interoperable environment. FMS models and their associated services include:

- Virtual Field Device (VFD) Support
- Object Dictionary (OD) Management
- Context Management
- Domain Management
- Program Invocation Management
- Variable Access
- Event Management

### **User Layer**

Functionality of **user layer** is based on function blocks applications. Function blocks are building blocks of function block applications, each handling one clearly distinguishable entity, like PID control. Function block applications are defined as plant or factory applications that perform one or more automatic monitoring and control functions. User Layer also provides the means for time critical scheduling of function blocks.

Function blocks whose scheduling and usage is completely user configurable and is called "device application" blocks. These function blocks provide various functions that are used in a control system, including input, output, signal selection and various control actions.

## 2.3 Advantages of Fieldbus

The fieldbus system offers two-way communication up to field devices. It means that we cannot only get but also can send data between a control room and field devices. Two-way communication actually extends *visibility* to the field level. An opportunity to make inquiries to field devices helps us in supervision and diagnostics. Some advantages of bi-directional digital communications, over 4-20 mA, are

- Higher accuracy and data reliability
- Multi-variable access
- Remote configuration and diagnostics
- Reduction of wiring
- Use existing 'analog wiring'

## CHAPTER 3

### NEURAL NETWORK TECHNOLOGY

---

#### 3.1 What are Artificial Neural Networks?

Artificial Neural Networks are being looked as the wave of the future in computing. They are collections of mathematical models that emulate some of the observed properties of biological nervous systems. It can also be seen as an electronic models based on the neural structure of the brain.

The brain basically learns from experience. By research it is known that brains store information as patterns. . Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages which involves the creation of massively parallel networks and the training of those networks to solve specific problems.

##### 3.1.1 Analogy to the Brain

The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell, which, unlike the rest of the body, does not appear to regenerate. Because this type of cell is the only part of the body which is not slowly

replaced, it is assumed that these cells provides us the abilities to remember, think, and apply previous experiences to our every action. These cells, all 100 billion of them, are known as **neurons**. Each of these neurons can connect with up to 200,000 other neurons.

The power of the human mind comes from the sheer numbers of these basic components and the multiple connections between them. It also comes from genetic programming and learning. The individual neurons are complicated. Together these neurons and their connections form a process, which is not binary, not stable, and not synchronous. These artificial neural networks try to replicate only the most basic elements of this complicated, versatile, and powerful organism.

### 3.1.2 Artificial Neurons and their Working Process

The fundamental processing element of a neural network is a neuron. Neuron looks like the figure 3.1 which is given below This building block of human awareness encompasses a few general capabilities. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then outputs the final result. Figure 3.1 shows the relationship of these four parts.

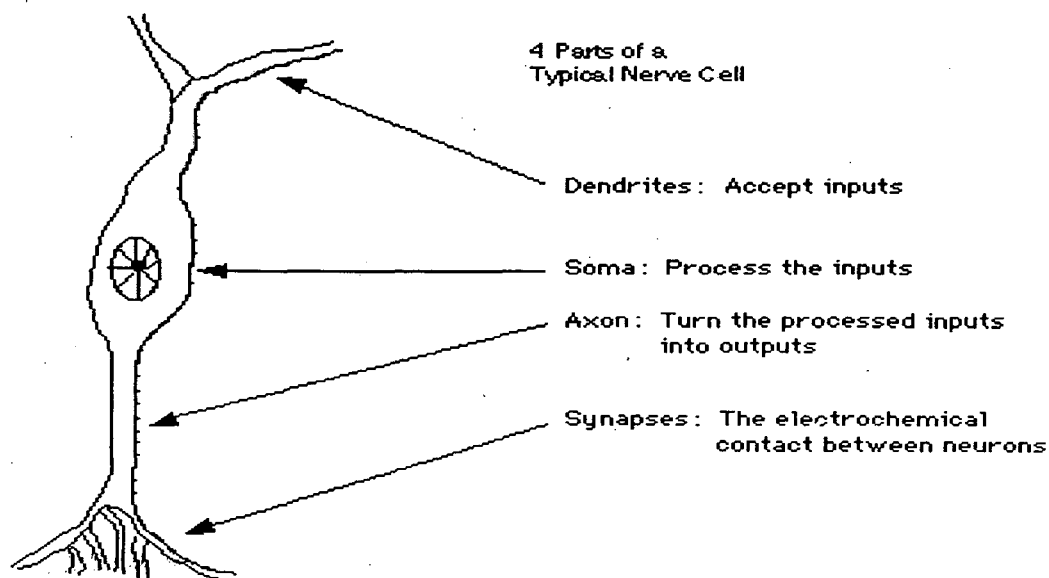


Figure 3.1 A Simple Neuron.

Within humans there are many variations in this basic type of neuron but all natural neurons have the same four basic components. These components are known by their biological names - dendrites, soma, axon, and synapses. Dendrites are hair-like extensions of the soma, which act like input channels. These input channels receive their input through the synapses of other neurons. The soma then processes these incoming signals over time. The soma then turns that processed value into an output, which is sent out to other neurons through the axon and the synapses. Currently, the goal of artificial neural networks is not the grandiose recreation of the brain. On the contrary, neural network researchers are seeking an understanding of nature's capabilities for which people can engineer solutions to problems that have not been solved by traditional computing. To do this, the basic units of neural networks, the artificial neurons, simulate the four basic functions of natural neurons. Figure 3.2 shows a fundamental representation of an artificial neuron.

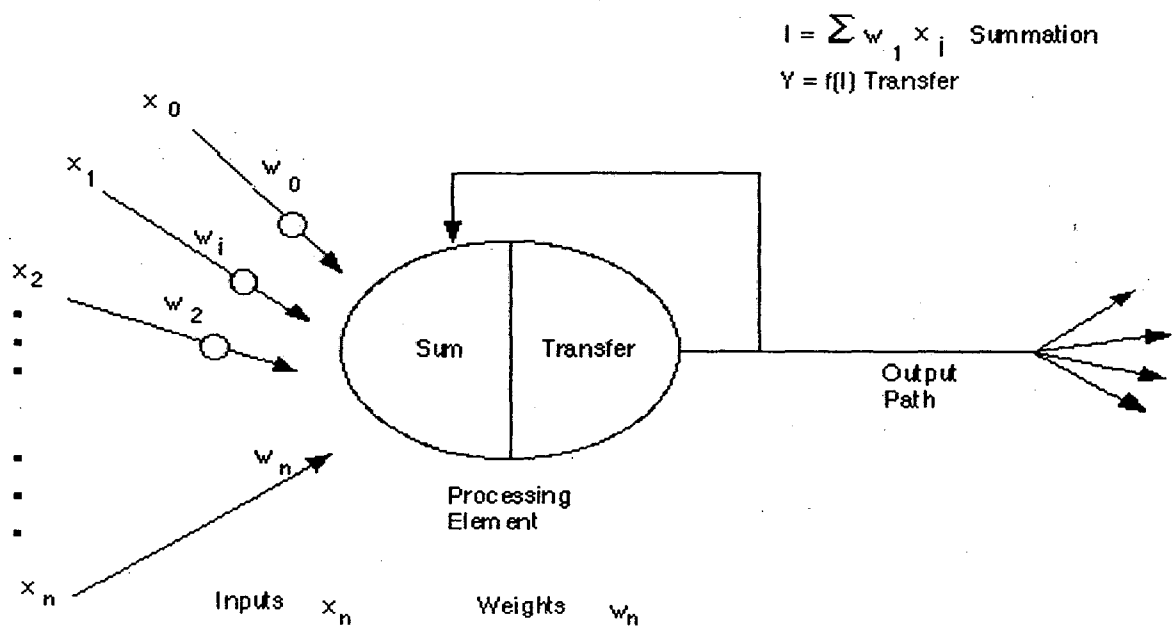


Figure 3.2 A Basic Artificial Neuron.

In Figure 3.2 various inputs to the network are represented by the mathematical symbol,  $x(n)$ . Each of these inputs is multiplied by a connection weight. These weights



are represented by  $w(n)$ . In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output.

### 3.1.3 Artificial Network Operations

Biologically, neural networks are constructed in a three-dimensional world as shown in figure 3.3. These neurons seem capable of nearly unrestricted interconnections. That is not true of any proposed, or existing, man-made network. Integrated circuits, using current technology, are two-dimensional devices with a limited number of layers for interconnection.

This physical reality restrains the types, and scope, of artificial neural networks that can be implemented in silicon. Currently, neural networks are the simple clustering of the primitive artificial neurons. This clustering occurs by creating layers, which are then connected to one another. Connection of these layers depends on the world problems. Although there are useful networks, which contain only one layer, or even one element but most applications require networks that contain at least the three normal types of layers - input, hidden, and output.

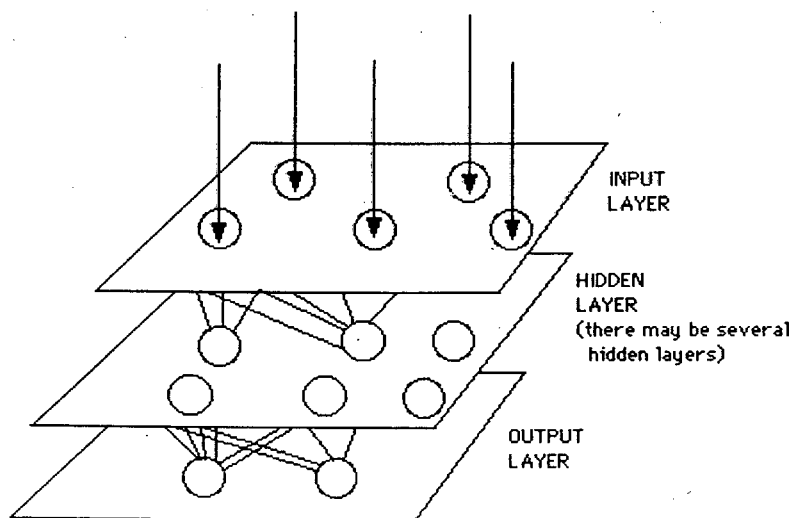


Figure 3.3 A Simple Neural Network Diagram.

Almost all artificial neural networks have a similar structure as shown in Figure. In that structure some of the neurons interfaces to the real world to receive its inputs.

Other neurons provide the real world with the network's outputs. All the rest of the neurons are hidden from view. But a neural network is more than a bunch of neurons. It needs logical presentation of neurons. One of the easiest ways to design a structure is to create layers of elements. It is the grouping of these neurons into layers, the connections between these layers, and the summation and transfer functions that comprises a functioning neural network. The general terms used to describe these characteristics are common to all networks.

In most networks, each neuron in a hidden layer receives the signals from all of the neurons in a layer above it, typically an input layer. After a neuron performs its function it passes its output to all of the neurons in the layer below it, providing a feed forward path to the output.

These lines of communication from one neuron to another are important aspects of neural networks. They are the connections, which provide a variable strength to an input. There are two types of these connections. One causes the summing mechanism of the next neuron to add while the other causes it to subtract.

Some networks want a neuron to inhibit the other neurons in the same layer. This is called lateral inhibition. The most common use of this is in the output layer. For example, in text recognition if the probability of a character being a "P" is .85 and the probability of the character being an "F" is .65, the network wants to choose the highest probability ( here "P") and inhibit all the others. It can do that with lateral inhibition. This concept is also called competition.

Another type of connection is feedback. This is where the output of one layer routes back to a previous layer. An example of this is shown in Figure 3.4

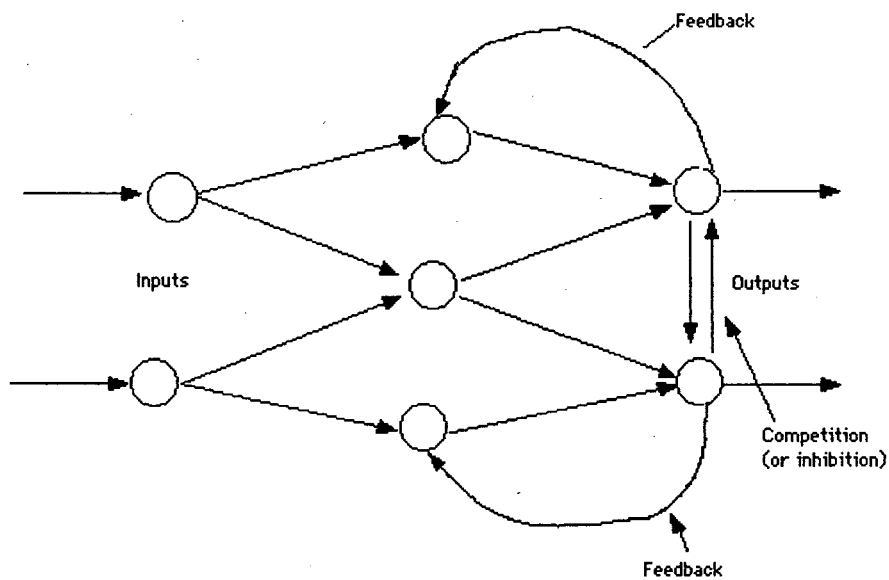


Figure 3.4 Simple Network with Feedback and Competition.

## 3.2 Training an Artificial Neural Network

Once a network has been structured for a particular application, the network is ready to be trained. To start this process the initial weights are chosen randomly. Then the training or learning begins.

There are two approaches to training

Supervised Training.

Unsupervised Training.

### 3.2.1 Supervised Training

Supervised training requires the pairing of each input vector with a target vector representing the desired output; together these are called a training pair. Usually a network is trained over a number of such training pairs.

The network processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights, which control the network.. This process occurs over and over as the weights are continually tweaked. The set of data, which enables the training, is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refired. When finally the system has been correctly trained means no further learning is needed, the weights can freeze.

### 3.2.2 Unsupervised Training.

In this training mechanism there is no requirement of any target vector for the outputs, and hence no comparisons to predetermined ideal responses. The training set consists of input vectors. The training algorithm modifies the network weights to produce output vectors that are consistent; i.e. both application of one of the training vectors or application of a vector that is sufficiently similar to it will produce the same pattern of outputs.

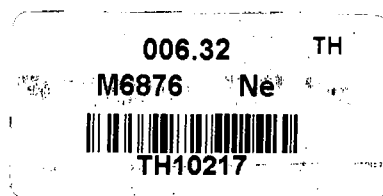
The training process extracts the statistical properties of the training set and groups similar vectors into classes. Applying a vector from a given class to the input will produce a specific output vector. There is no way to determine prior to training which specific output pattern will be produced by a given input vector class.

### 3.2.3 Learning Rates

The rate at which ANNs learn depends upon several controllable factors. In selecting the approach there are many exchanges and compromises. A slower rate means a lot more time is spent in accomplishing the off-line learning to produce an adequately trained system. With the faster learning rates, however, the network may not be able to make the fine discriminations possible with a system that learns more slowly.

Generally, several factors besides time have to be considered when discussing the off-line training task, which is often described as "tiresome." Network complexity, size,

TH-10217



paradigm selection, architecture, type of learning rule and desired accuracy must all be considered. These factors play a significant role in determining how long it will take to train a network. Changing any one of these factors may either extend the training time to an unreasonable length or even result in an unacceptable accuracy.

Most learning functions have some provision for a learning rate, or learning constant. Usually this term is positive which is between zero and one. If the learning rate is greater than one, it is easy for the learning algorithm to overshoot in correcting the weights, and the network will oscillate. Small values of the learning rate will not correct the current error as quickly, but if small steps are taken in correcting errors, there is a good chance of arriving at the best minimum convergence.

### **3.2.4 Learning Laws**

Many learning laws are in common use. Most of these laws are some sorts of variation of the best known and oldest learning law, Hebb's Rule. Learning is certainly more complex than the simplifications represented by the learning laws currently developed. Some of the major laws are briefed here.

**Hebb's Law:** The first, and undoubtedly the best known, learning law was introduced by Donald Hebb. His basic law is that If a neuron receives an input from another neuron, and if both are highly active (mathematically have the same sign), the weight between the neurons should be strengthened.

**Hopfield Law:** It is similar to Hebb's law with the exception that it specifies the magnitude of the strengthening or weakening. It states, "if the desired output and the input are both active or both inactive, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate."

**Kohonen's Learning Law:** This procedure, developed by Teuvo Kohonen, was inspired by learning in biological systems. In this procedure, the processing elements compete for the opportunity to learn, or update their weights. The processing element with the largest

the opportunity to learn, or update their weights. The processing element with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbors. Only the winner is permitted an output, and only the winner plus its neighbors are allowed to adjust their connection weights.

The usual paradigm is to start with a larger definition of the neighborhood, and narrow in as the training process proceeds. Because the winning element is defined as the one that has the closest match to the input pattern, Kohonen's networks model the distribution of the inputs. This is good for statistical or topological modeling of the data and is sometimes referred to as self-organizing maps or self-organizing topologies.

### **3.3 Major Components of an Artificial Neuron**

Here we describe the seven major components, which make up an artificial neuron. These components are valid whether the neuron is used for input, output, or is in one of the hidden layers.

**Component 1. Weighting Factors:** A neuron usually receives many simultaneous inputs. Each input has its own relative weight, which gives the input the impact that it needs on the processing element's summation function. These weights perform the same type of function, as do the varying synaptic strengths of biological neurons.

Weights are adaptive coefficients within the network that determine the intensity of the input signal as registered by the artificial neuron. They are a measure of an input's connection strength. These strengths can be modified in response to various training sets and according to a network's specific topology or through its learning rules.

**Component 2. Summation Function:** The first step in a processing element's operation is to compute the weighted sum of all of the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as  $(i_1, i_2 \dots i_n)$  and  $(w_1, w_2 \dots w_n)$ . The total input signal is the dot product of these two vectors. This simplistic summation function is found by multiplying each component of the  $i$  vector by the

corresponding component of the  $w$  vector and then adding up all the products.  $input1 = i1 * w1$ ,  $input2 = i2 * w2$ , etc., are added as  $input1 + input2 + \dots + input n$ . The result is a single number, not a multi-element vector.

Geometrically, the inner product of two vectors can be considered a measure of their similarity. If the vectors point in the same direction, the inner product is maximum, if the vectors point in opposite direction (180 degrees out of phase), their inner product is minimum.

Some summation functions have an additional process applied to the result before it is passed on to the transfer function. This process is sometimes called the activation function. The purpose of utilizing an activation function is to allow the summation output to vary with respect to time.

**Component 3. Transfer Function:** The result of the summation function, almost always the weighted sum, is transformed to a working output through an algorithmic process known as the transfer function. In the transfer function the summation total can be compared with some threshold to determine the neural output. If the sum is greater than the threshold value, the processing element generates a signal. If the sum of the input and weight products is less than the threshold, no signal (or some inhibitory signal) is generated. Both types of response are significant. The threshold or transfer function is generally non-linear. The transfer function could be something as simple as depending upon whether the result of the summation function is positive or negative. The network could output zero and one, one and minus one, or other numeric combinations.

**Component 4. Scaling and Limiting:** After the processing element's transfer function, the result can pass through additional processes, which scale and limit. This scaling simply multiplies a scale factor times the transfer value, and then adds an offset. Limiting is the mechanism, which insures that the scaled result does not exceed an upper, or lower bound. This limiting is in addition to the hard limits that the original transfer function may have performed.

**Component 5. Output Function (Competition):** Each processing element is allowed one output signal, which it may output to hundreds of other neurons. This is just like the biological neuron, where there are many inputs and only one output action. Normally, the output is directly equivalent to the transfer function's result. Some network topologies, however, modify the transfer result to incorporate competition among neighboring processing elements. Neurons are allowed to compete with each other, inhibiting processing elements unless they have great strength. Competition can occur at one or both of two levels. First, competition determines which artificial neuron will be active, or provides an output. Second, competitive inputs help determine which processing element will participate in the learning or adaptation process.

**Component 6. Error Function and Back-Propagated Value:** In most learning networks the difference between the current output and the desired output is calculated. This raw error is then transformed by the error function to match particular network architecture. The artificial neuron's error is then typically propagated into the learning function of another processing element. This error term is sometimes called the current error.

The current error is typically propagated backwards to a previous layer. Yet, this back-propagated value can be either the current error, the current error scaled in some manner (often by the derivative of the transfer function), or some other desired output depending on the network type. Normally, this back-propagated value, after being scaled by the learning function, is multiplied against each of the incoming connection weights to modify them before the next learning cycle.

**Component 7. Learning Function:** The purpose of the learning function is to modify the variable connection weights on the inputs of each processing element according to some neural based algorithm. This process of changing the weights of the input connections to achieve some desired result which can also be called the adaptation function, as well as the learning mode. There are two types of learning: supervised and unsupervised which we have already discussed.



### 3.4 Neural Network Selection

Because all artificial neural networks are based on the concept of neurons, connections and transfer functions, there is a similarity between the different structures or architectures of neural networks. Some problem areas are prediction, classification, data association, data conceptualization data filtering .A brief description of the neural architectures are given in the table below used in solving the problem in given problem areas.

Network Type	Networks	Use of Network
<b>Prediction</b>	Back-propagation Delta Bar Deita Extended Delta Bar Delta Directed Random Search Higher Order Neural Networks Self-organizing map into Back-propagation	Use input values to predict some output (e.g. pick the best stocks in the market, predict weather, identify people with cancer risks etc.)
<b>Classification</b>	Learning Vector Quantization Counter-propagation Probabilistic Neural Networks	Use input values to determine the classification (e.g. is the input the letter A, is the blob of video data a plane and what kind of plane is )
<b>Data Association</b>	Hopfield Boltzmann Machine Hamming Network	Analyze the inputs so that grouping relationships can be inferred (e.g. extract from a database the names of those most

	Bi-directional associative Memory Spation-temporal Pattern Recognition	likely to buy a particular product)
<b>Data Conceptualization</b>	Adaptive Resonance Network Self Organizing Map	Analyze the inputs so that grouping relationships can be inferred (e.g. extract from a database the names of those most likely to buy a particular product)
<b>Data Filtering</b>	Recirculation	Smooth an input signal (e.g. take the noise out of a telephone signal)

Network Selection Table

### 3.5 Uses of Neural Network

Many of the networks now being designed are statistically quite accurate. But the problems with these systems are that these networks might be 85% to 90% accurate. Unfortunately, few applications tolerate that level of error. Still the problem is with those systems, which demand 100% accuracy. Currently, neural networks are not the user interface, which translates spoken words into instructions for a machine. Neural networks are simply used in the area where statistical accuracy is valuable. Some of such areas are discussed here.

#### 3.5.1 Language Processing

Language processing includes a wide variety of applications. These applications include text-to-speech conversion, auditory input for machines, automatic language translation, secure voice keyed locks, automatic transcription, aids for the physically disabled which respond to voice commands, and natural language processing.

This application needs capability to be able to understand the 50,000 most commonly spoken words. Currently, according to the academic journals, most of the hearing-capable neural networks are trained to only one talker. These one-talker, isolated-word recognizers can recognize a few hundred words. Within the context of speech, with pauses between each word, they can recognize up to 20,000 words.

### **3.5.2 Signal Processing**

Neural networks' promise for signal processing has resulted in a number of experiments in various university labs. Neural networks have proven capable of filtering out noise. Widrow's MADALINE was the first network applied to a real-world problem. It eliminates noise from phone lines.

Another application is a system that can detect engine misfire simply from the noise. This system, developed by Odin Corp., works on engines up to 10,000 RPMS. The Odin system satisfies the California Air Resources Board's mandate that by 1994 new automobiles will have to detect misfire in real time. Misfires are suspected of being a leading cause of pollution. The Odin solution requires 3 Kbytes of software running on a Motorola 68030 microprocessor.

### **3.5.3 Character Recognition**

Character recognition is another area in which neural networks are providing solutions. The largest amount of research in the field of character recognition is going on for scanning oriental characters into a computer.

### **3.5.4 Image (data) Compression**

A number of studies have been done proving that neural networks can do real-time compression and decompression of data. These networks are auto associative. They can reduce eight bits of data to three and then reverse that process upon restructuring to eight bits again. The problem with these implementations is that they are not lossless.

### **3.5.5 Financial**

Neural networks are being used in the field of financial worlds. Banking, credit card companies, and lending institutions deal with decisions that are not clear-cut. They involve learning and statistical trends.

The loan approval process involves filling out forms, which hopefully can enable a loan officer to make a decision. The data from these forms is now being used by neural networks, which have been trained on the data from past decisions.

Credit card companies are also using similar back-propagation networks to aid in establishing credit risks and credit limits.

Neural networks are being used in all of the financial markets - stock, bonds, international currency, and commodities. Indeed, neural networks are reported to be highly successful in the Japanese financial markets. Daiichi Kangyo Bank has reported that for government bond transactions, neural networks have boosted their hit rate from 60% to 75%. Daiwa research Institute has reported a neural net system, which has scored 20% better than the Nikkei average.

### **3.6 What Currently Exists**

Neural network products exist which are simply add-ons to the popular databases and spreadsheets. Some other products are also exists in the area of operating systems on particular machines. Some of these packages are developed for particular applications such as image processing. Others are general but lack good data routing capabilities. Some of the systems are briefed here.

### **3.6.1 Development Systems**

Good development systems allow a user to prototype a network, to train it and to use it. These systems run on the standard range of computers. These packages usually don't run on specialized hardware, although some vendors have packaged fast RISC processors into special neural processing boards. Usually, these packages are simply tools, which create networks that prove concepts but may be way too slow to run.

### **3.6.2 Hardware Accelerators**

The key to the continued evolution of neural networking lies in the hardware. Traditional hardware does not enable the massive parallelism that is required by neural networks. There are several approaches that are being worked on. One is to develop a processor, which is specifically tailored to performing the tasks of individual artificial neurons. Another approach is to package fast processors, primarily RISCs, onto a hardware accelerator.

### **3.6.3 Dedicated Neural Processors**

Dedicated neural processors are processors with specific capabilities that enable their use in neural networks. Several of the large chip manufacturers have developed neural processors. Some of these processors were created specifically for the development system vendors. Some of these chips package a number of simplistic neurons onto one chip. Others incorporate proprietary concepts, such as creating a specific type of fuzzy neuron. These chips come in many broad technologies - analog, digital, hybrid, and optical.

### **3.6.4 What the next developments will be ?**

The vendors within the industry predict that migration from tools to applications will continue. In particular, the trend is to move toward hybrid systems. These systems will encompass other types of processes, such as fuzzy logic, expert systems, and kinetic algorithms. Indeed, several manufactures are working on "fuzzy neurons" .

The fuzzy logic is merge with neural networks. Fuzzy logic incorporates the inexactness of life into mathematics. In life most pieces of data do not exactly fit into certain categories. For instance, a person is not just short or tall. He can be pretty tall, a little above average, or very tall. Fuzzy logic takes these real-world variations into account. In potential application of neural networks, in systems, which solve real problems, this fuzziness is a large part of the problem. In automating a car, to stop is not to slam on the brakes; to speed up is not to "burn rubber." To help neural networks accommodate this fuzziness of life, some researchers are developing fuzzy neurons. These neurons do not simply give yes/no answers. They provide a more fuzzy answer.

Systems built with fuzzy neurons may be initialized to what an expert thinks are the rules and the weights for a given application. This merging of expert systems and fuzzy logic with neural networks utilizes the strength of all three disciplines to provide a better system than either can provide themselves.

Expert systems have the problem that most experts do not exactly understand all of the nuances of an application and, therefore, are unable to clearly state rules, which define the entire problem to someone else. But the neural network does not care that the rules are not exact, for neural networks and can then learn and correct, the expert's rules.

It can add nodes for concepts that the expert might not understand. It can tailor the fuzzy logic, which defines states like tall, short, fast, or slow. It can tweak itself until it can meet the user-identified state of being a workable tool. In short, hybrid systems are the future.

## CHAPTER 4

# HOPFIELD NETWORK IN PROCESS SCHEDULING IN FIELDBUS COMMUNICATION SYSTEMS

---

### 4.1 Process Scheduling

The problems of scheduling the information flow in communication systems for process control known as FieldBuses. A FieldBus is a serial communication system interconnecting several field devices such as sensors, and PLCs.

The need arise to serialize the information traffic produced by the field devices connected to Fieldbus network. This traffic is mostly made up of system variable values, which are updated by **producer processes** and used by **consumer processes**. They can be updated periodically or asynchronously. A temperature sensor, for example, samples an analog signal and periodically produces temperature values. An alarm, on the other hand is a variable, updating of which is unpredictable, i.e. asynchronous.

Management of a periodic information flow has to meet two types of constraints. From the viewpoint of the producer process, each variable value has to be transmitted before the new value is produced and written over the previous one. As far as the consumer is concerned, it is necessary to respect the time required to reconstruct the analog signal from the values it receives.

If the information flow in a FieldBus is to be managed correctly, it is therefore not sufficient for the producer process to respect its time constraints. Although transmitted



within the production period, a variable may, in fact, be received by the consumer process in the stipulated time. Each periodically updated variable is therefore associated with a deadline which takes the time constraints into account. The scheduling of a periodic information flow has to meet all the variables' deadlines.

As far as an asynchronous information flow is concerned, it is important for an asynchronously updated variable to be transmitted with a minimum delay.

If FieldBuses belonging to different communication protocols, and the presence of different modes of transmission (coaxial cable, fibre optics, radio) are connected with each other by bridges. This kind of communication system will be referred to as a Multi- FieldBus environment to distinguish it from a Single FieldBus scenario featuring a single communication system.

Periodically updated variables generally have very strict deadlines. Typical values are in the range of a few ms. The delay times in asynchronous traffic is allowed, it may be as long as a few tens of ms. It is therefore clear that scheduling of a periodic flow is much more critical than that of an asynchronous flow.

## **4.2 Scheduling in a Single FieldBus Environment**

We have taken the mode with which all the devices gain access to the FieldBus is of the Producer ( $P_{px}$ )/Consumer ( $P_{cx}$ ). One device takes on the role of  $P_{px}$ , controlling the information flow to be transmitted through the communication system. This occurs by polling the  $P_{cx}$  stations and authorizing transmission on request.

A producer  $P_{px}$  is characterized by a period,  $T_{px}$  of production and by an instant  $r_{px}$ , at which the first production starts. On the basis of above hypothesis, a consumer process,  $P_{cx}$ , is characterized by the tuple  $(T_{px}, r_{px}, r_{cx})$  where  $T_{px}$ ,  $r_{px}$  are referred to producer process characteristics and  $r_{cx}$  is the time between production and consumption.

Periodic and asynchronous information flows are managed in two separate time windows, which we will call synchronous and asynchronous windows. These windows are shown in figure 4.1. During the communication, periodic traffic is served, while the latter is dedicated to asynchronous traffic. The two windows follow each other cyclically, as shown in figure 4.1.

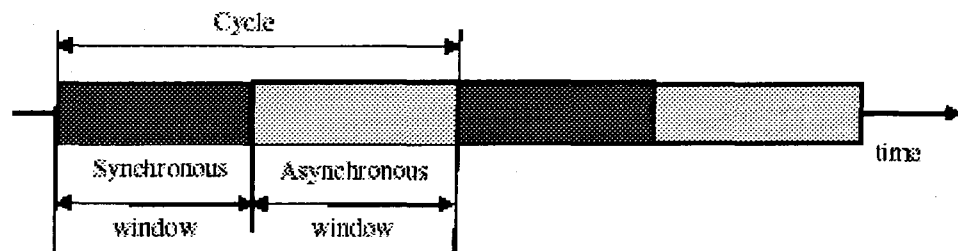


Figure 4.1 Synchronous and Asynchronous time windows.

The figure 4.1 also gives a definition of a cycle, which is equal to a single sequence of one synchronous and one asynchronous window. The reason for choosing two separate windows is to minimize the cost of context switching between the transfer of periodic traffic and that of asynchronous traffic.

As we discussed earlier to guarantee correct transmission of a periodic flow, each periodically updated variable value has to be transmitted at least once before its deadline. The deadline takes into account the time constraints of both the producer and consumer processes. Figure 4.2 shows the deadline and production period of producer consumer processes. It is defined as the length of each time interval between a generic production instant and the subsequent consumption instant. If these two coincide, the deadline will be taken as equal to the production period of the variable.

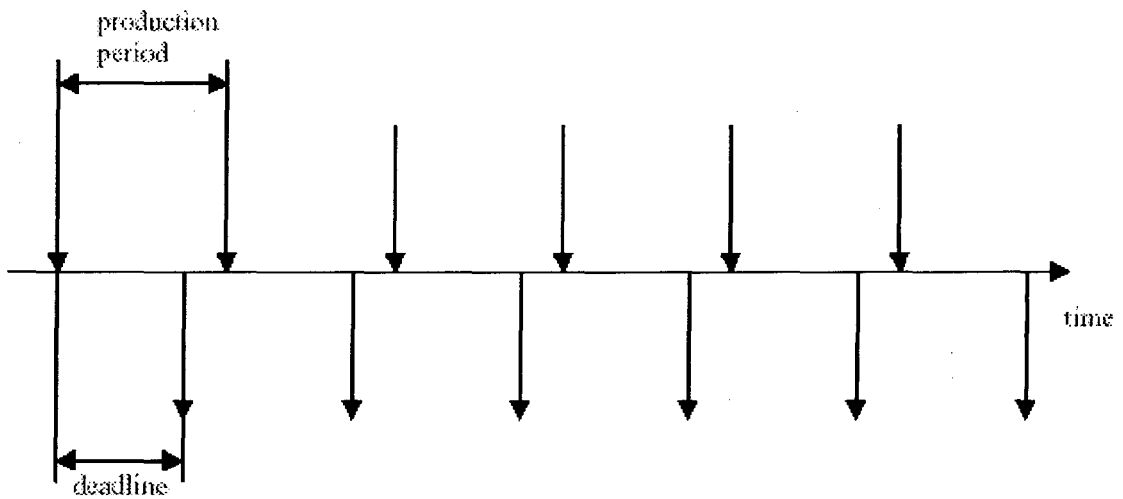


Figure 4.2 Production Period and Deadline

The problem of scheduling in a single FieldBus environment is to insert the polling for transmission of the periodically updated variables into the synchronous window in such a way that their deadlines are met.

Several approaches are used to the scheduling of periodic information in a single FieldBus communication system. Some of which we will discuss in brief in next section. To present the scheduling problem solution we have used the following notations

**i-th group**: Set of variables periodically updated with the same period (i.e. deadline).

**Ng** : Number of groups.

**ni** : Number of variables in the i-th group.

**Ti** : Period of the i-th group, which coincides with the deadline for group i.

**trr** : Data Transition Time for a variable, given by the sum of the time required to poll one variable and that needed to accomplish one data transfer.

### 4.3 Monocycle Polling Scheduling

Monocycle polling is the most common scheduling strategy used. For example, it is the strategy, which is considered in the FIP standard (French Association for Standardization 1990).

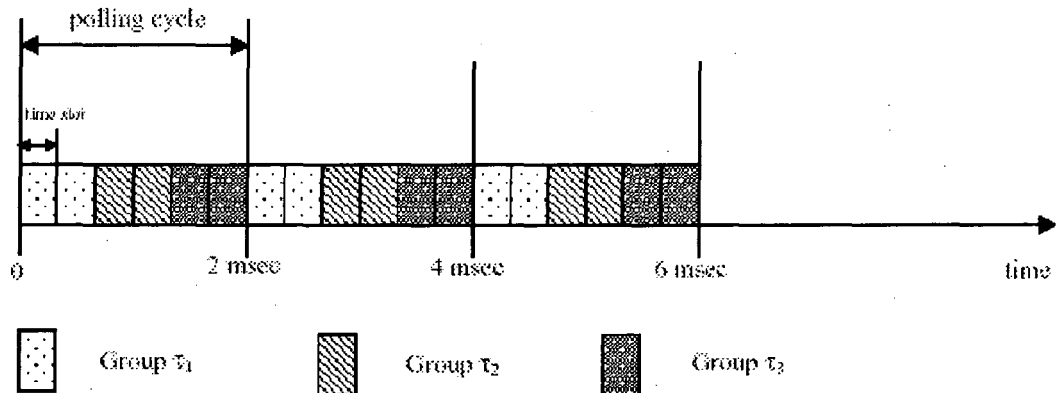


Figure 4.3 Example of Monocycle Polling Scheduling.

In monocycle polling scheduling which is shown in figure 4.3, the Producer uses a single cycle for the polling required for all the variables belonging to the  $g$  groups in the task set to be transmitted. The main disadvantage of this approach is represented by the poor exploitation of the available bandwidth, as the values of all the variables in the task set are transmitted in each polling cycle.

A solution that overcomes this limitation is known as multicycle polling scheduling.

### 4.4 Multicycle Polling Scheduling

This scheduling strategy is based on the idea of differentiating between the cycles used by the Producer to poll the periodic variables, on the basis of their period.

### 4.5 Extended Rate Multicycle Polling Scheduling

In this scheduling strategy the case has been considered when we have one or more variables which have nonharmonic periods, the solution can be extended

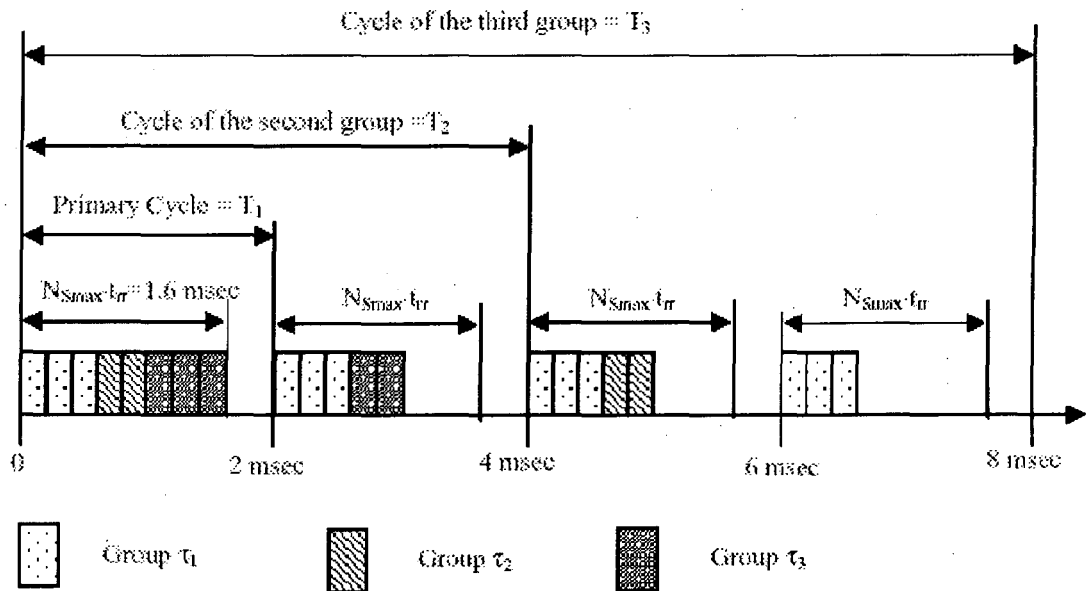


Figure 4.4 Example of Multicycle Polling Scheduling.

by rounding down the periods of the task set so as to meet the hypothesis that all the periods of the periodic variables are harmonic, i.e. the following relation always holds:

$$\forall i, j: 1 \leq i, j \leq g \text{ and } T_i \geq T_j \Rightarrow T_i = k \cdot T_j \quad k \in \mathbb{N}$$

where  $g$  is total number of group and  $\mathbb{N}$  is 1, 2, ...

In this implementation values of all the variables are assumed to be of the same size. The  $tr$  value is therefore the same for all the groups. The groups are ordered in ascending order by their deadline, so the group with the index 1 has the most urgent deadline (higher priority).

As we discussed, all the deadlines are rounded up/down to multiples of the deadline for group 1. For example, if we have two variables with periods of 25 msec and 315 msec, the latter is rounded down to 300 msec, so as to satisfy the above hypothesis. If this is 20ms., and a generic deadline is 310 ms, in this case also it is rounded down to 300ms. The length of each cycle thus coincides with the deadline for group 1 and is called a **primary cycle**; the deadline for a generic group with the index  $i$ , called a **secondary cycle**. The value of secondary cycle is  $T_i = k_i * T_1$ , where  $K_i \in \mathbb{N}$ .

On the basis of this new hypothesis the length of each cycle, which is still called a **primary cycle**, is equal to the Least Common Multiple (LCM) of all the deadlines.

Scheduling Algorithms are classified as static and dynamic algorithms according to whether the length of the synchronous time window is considered to be constant or not. Extended Rate Multicycle Polling Scheduling algorithm is static and priority based algorithm.

In Extended Rate Multicycle scheduling, each time window making up the primary cycle is divided into a certain number of slots of time, each lasting  $trr$ . If  $NS_{max}$  is the number of variables to schedule in the synchronous time window, the length of this window will be  $TS_{max} = NS_{max} * trr$ .

The scheduling of the periodic information flow consists of determining the complete polling sequence in each synchronous time window, such that all the deadlines are met. It is also necessary for  $NS_{max}$  to be minimized so that the number of slot times for the asynchronous flow is maximized.

These two objectives can be achieved by means of an algorithm that determines the polling sequence in each synchronous time window on-line, during the evolution of the system. As we discussed ERMM is an priority-based algorithm. It meets the above discussed requirements and comprises of three steps:

#### **A. Calculation of $NS_{max}$ .**

As said previously, it is assumed that the periods of the tasks have quite arbitrary values. On the basis of this assumption, the value of  $NS_{max}$  can be

calculated starting from the general condition governing the schedulability of a task set given by (Lehoczky 1989)(Lehoczky 1992):

$$N_{smax} = \max_{1 \leq i \leq g} \left[ T_{pe} \cdot \sum_{j=1}^i \frac{n_j}{T_i} \left\lceil \frac{T_i}{T_j} \right\rceil \right]$$

### B. Calculation and assignment of priorities:

Each group of tasks is assigned a priority based on its period, the highest priority corresponding to the shortest period. Before polling is carried out all the tasks are ordered by decreasing priority;

### C. Execution of the polling algorithm:

The bellow given algorithm is the ERMM polling algorithm.

#### Algorithm

- 1.do {
- 2.if (the Primary Cycle starts) {
- 3.poll all the  $n_1$  variables of group  $g_1$ ;
- 4.set to 0 the number of variables to be polled in the group  $g_1$ ;
- 5.set to  $n_1$  the number of variables polled in the primary cycle;
- 6.do {
- 7.get group with the highest priority, in which variables remain to be polled;
- 8.if (remaining variables of group  $g_i$  ( $N_{Smax}$ -number of variables polled in the primary cycle))
- {
- 9.poll all remaining variables of the group;
- 10.set to 0 the number of variables to be polled in the group;
- }
- 11.else {
- 12.poll ( $N_{Smax}$ -number of variables polled in the primary cycle) variables of the group;
- 13.update the number of variables to be polled in the group;

```

}
14.update the number of variables polled in the primary cycle;
}
15.while (number of variables polled in the primary cycle <NSmax);
16.for (i=0; i<g; i++)
17.if (the cycle of the group gi starts)
18.set to ni the number of variables to be polled in the group i;
}
19.while (true);

```

#### 4.6 Determination of Scheduling Sequence( S<sub>tx</sub>)

Assume primary cycle Ts be a multiple of T<sub>px</sub> then the number of number production /consumption interval( Ng) will be in one primary cycle which is represented by n<sub>px</sub>.

$$n_{px} = Ts / T_{px}$$

If every production is taking a transmission in that case the total number of transmissions n<sub>tx</sub> will be

$$n_{tx} = n_{px}$$

For determining the scheduling sequence S<sub>tx</sub> we are taking two-dimensional matrix of size n<sub>px</sub> \* Ts with binary values, we will call it Tbl. The i<sup>th</sup> row ( i = 1.....n<sub>px</sub>) in Tbl represent the interval between a production and the subsequent consumption in group i. By using Tbl, we can determine S<sub>tx</sub> which cover all the n<sub>px</sub> intervals represented in a row in the Tbl.

##### Example:

Here we have shown one example of scheduling sequence determination. We have taken Ts = 12, n<sub>tx</sub> = 3, n<sub>px</sub> = 3, T<sub>px</sub> = 4, r<sub>px</sub> = 0 and r<sub>cx</sub> = 3.



	0	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	1	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	1	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	1

Tbl for Determining Scheduling Sequence

#### 4.7 Hopfield Neural Network and Scheduling Problem in Single Fieldbus Systems

The Hopfield neural network is very suitable for solving optimization problems. This network was first used to solve the well-known Traveling Selsman Problem, then it has been used in a large number of optimization problem like "Neural Network for Shortest Path Computation and Routing in Computer Networks" and "Neural Networks for Process Scheduling Communication System". The Hopfield-type model is based on a single-layer architecture of neurons, the outputs of which are fed back toward the inputs. In our problem i.e. to schedule  $p$  Producer/Consumer process  $P_{px}/P_{cx}$  we are using Hopfield strategy.

The neural solution to this scheduling problem has been achieved by going through the following steps.

- a. Definition of the neural architecture.
- b. Specification of the surrounding conditions of the scheduling problem to be solved.
- c. Calculation of the bias currents and weights for the neural architecture defined.

#### 4.8 Definition of the neural architecture:

The definition of the neural architecture ( i.e. the numer of nodes) strictly depends on the coding chosen for the solution of the problem . In this problem i.e. scheduling problem we assumed that the output of each neuron corresponds to a transmission instant in a primary cycle. We can say that each pair of  $P_{px}/P_{cx}$  is associated with  $T_s$  neurons representing their instants in the primary cycle.

For each pair of processes, the generic neuron takes a value 1 if a transmission relating to the pair occurs in the instant it represents. A value 0 means no transmission was scheduled for the pair  $P_{px}/P_{cx}$  in the instant represented by the neuron.

The total number of neurons in the neural model is equal to the product of number of pairs of producer consumer processes and the length of primary cycle, i.e.  $T_s \cdot p$ . The neurons of the neural model are logically divided into  $p$  groups of  $T_s$  neurons each.

According to our assumption , each group refers to a pair of producer/Consumer process  $P_{px}/P_{cx}$  with  $x \in [1...p]$  and contains  $T_s$  neurons, each representing to an instance in the scheduling interval .

Here we have an example of a possible neural solution to the scheduling of three pairs of Producer/Consuler processes characterized by  $( T_{p1}= 3, r_{p1} 0, r_{c1}= 3)$ ,  $( T_{p2} = 4, r_{p2} = 0, r_{c2} = 4)$ ,  $( T_{p3}= 6, r_{p3} = =0, r_{c3}= 6)$ and  $T_s = 12$ .

	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	1	0	0	1	0	0	1	0
2	0	0	1	0	0	1	0	0	1	0	0	0
3	0	0	0	1	0	0	0	0	0	1	0	0

**Example of Neural Solution**

The rows of the above matrix refer to the pairs of process, while the columns refer to the instant of the primary cycle. We can see that the neural model solution of transmission instants are :

$$St1 = \{1, 4, 7, 3\} , St2 = \{2, 5, 8\} \text{ and } St3 = \{3, 9\}$$

## 4.9 Neurons Specification

Each neuron is specified by a double index  $x_i$  or  $y_j$  where  $x, y = 1 \dots p$  identify the group to which neuron belongs, while  $I, j = 0 \dots T_s-1$  identify an instant in the primary cycle.

On the basis of our neuron specification we are representing the symbols used in Hopfield model.

1.  $O_{x_i} \rightarrow$  Output of the neuron with the index  $x_i$ , which assumes the value 0 or 1 according to the following rule

$O_{x_i} = 1$  if the transmission instant  $I$  belongs to the correct scheduling solution for  $P_{px}/P_{cx}$ .

$O_{x_i} = 0$  if no transmission is scheduled at instant  $I$  for  $P_{px}/P_{cx}$ .

2.  $U_{x_i}$ : Input of the neuron with index  $x_i$ .

3.  $W_{x_i, y_j}$ : Weight associated to the feedback between the output of the neuron with the index  $y_j$  and the input of the neuron with the index  $x_i$ .

4.  $I_x$ : External bias current supplied to the neuron with index  $x_i$ .

5. Activation monotonic increasing function of the neuron with the index  $x_i$ , where  $u_0$  control the steepness (control learning).

$$O_{x_i} = g_{x_i}(U_{x_i}) = \frac{1 + \tanh\left(\frac{U_{x_i}}{u_0}\right)}{2} \quad \text{----- (A)}$$

6.: The dynamics of the Hopfield network, where  $\tau$  is a user-selected decay constant.

$$\frac{dU_{x_i}}{dt} = -\frac{U_{x_i}}{\tau} + \sum_y \sum_j w_{x_i, y_j} \cdot O_{y_j} + I_{x_i}$$

## 4.10 Specification of the surrounding conditions of the scheduling problem to be solved

The neural network has to find a solution for the scheduling of  $p$  pairs of producer/consumer process in such a way that all the surrounding conditions are met. These conditions are

Number of Transmissions:  $n_{tx}$  i.e. the number of transmissions must be fixed in the primary cycle for the pair  $P_{px}/P_{cx}$ .

### 4.10.1 Correctness of the Scheduling of Processes

The neural network has to choose the transmission instants contained in each tuple with  $n_{tx}$  element in such a way that correct scheduling of pair  $P_{px}/P_{cx}$  is guaranteed. Let us consider a tuple of  $n_{tx} > 1$  transmissions and any two transmission instant  $t_{xi}$  and  $t_{xj}$  belonging to this tuple ( $i \neq j$ ). Let us assume that the transmission instants  $t_{xi}$  and  $t_{xj}$  are such as not to cover  $n_0$  rows in the matrix Tbl.  $n$  number of rows can be covered by a single transmission is 1, it follows that if

$n_0 > n_{tx} - 2$  holds.

By taking above condition into consideration and  $d_{xi}$  the  $i$ -th column in the matrix Tbl. A vector of  $n_{px}$  rows and 1 column ( $n_{px} \times 1$ ), where 1s identify the interval  $s$  for the pair  $P_{px}/P_{cx}$  covered by the transmission placed at the  $i$ -th instant. If we take the product  $d_{xi} \cdot d_{xj}^T$  (where  $i, j = 1.. n_{tx}, i \neq j$ ).

The integer value provided by this product is zero if the interval identified by  $d_{xi}$  are different from those identified by  $d_{xj}$ .

This consideration suggests that the introduction of the term  $d_{xi} \cdot d_{xj}^T$  (where  $i, j = 1.. n_{tx}, i \neq j$ ). in the expression of the energy function of the network will help determination of tuples with  $n_{tx}$  elements such that the set of open production/consumption intervals by each transmission will have a **null** intersection.

For the neural network to be able to verify the validity of a scheduling solution, we consider the product given by  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$ , where the vector with 1 row and  $n_{px}$  columns (1, x,  $n_{px}$ ),  $v_{xij}$ , is defined as follow

#### 4.10.2 Formalization of the Surrounding Conditions

$$v_{xij}(k) = \begin{cases} D & \text{if } d_{xi}(k)=1 \text{ and } d_{xj}(k)=1 \\ D & \text{if } (d_{xi}(k)=D \text{ or } d_{xj}(k)=D) \text{ and } (n_0 > n_{tx} - 2) \\ D & \text{if } (d_{xi}(k)=D \text{ and } d_{xj}(k)=D) \text{ and } (n_0 > n_{tx} - 2) \\ 1 & \text{therwise} \end{cases}$$

Hopfield showed that if the weights matrix  $W = [w_{xi,yj}]$  is symmetrical and if the function  $g_{xi}$  is a steep like curve (i.e.  $u \rightarrow 0$ ), the dynamics of the neurons described by activation monotonic increasing function follows the function known as Lyapunov function that is given bellow

$$E = \sum_x \sum_i \sum_y \sum_j w_{xi,yj} O_{xi} \cdot O_{yj} + \sum_x \sum_i I_{xi} \cdot O_{xi} \quad \text{---- (B)}$$

Here the surrounding conditions of the scheduling problem are expressed in the form of energy function given by above that is Lyapunov function. Then each term of the energy function referring to a specific surrounding condition is multiplied by a real coefficient weighting the influence of the condition itself on the neural solution. So each surrounding condition term be linked to a real coefficient(  $\alpha$ ,  $\beta$  and  $\gamma$  ).

### 4.10.3 Number of Transmissions

The first condition concerns the presence in each group of neurons  $x$  of exactly  $n_{tx}$  activated neurons. This constrains the neural network to choose tuples  $n_{tx}$  elements for each pair  $P_{px}/P_{cx}$ . The expression of this surrounding condition is

$$\frac{\alpha}{2} \cdot \sum_x \left( \sum_i O_{xi} - n_{tx} \right)^2 \quad \text{-----(C)}$$

### 4.10.4 Correctness of the Scheduling of a Single Process

This surrounding condition refers to the strategy for choosing the  $n_{tx}$  transmissions for each pair  $P_{px}/P_{cx}$ . The aim is to choose transmissions which will minimize overlapping between the production /consumption intervals identified by each transmission. And the condition for this surrounding condition is

$$\frac{\beta}{2} \cdot \sum_x \sum_i \sum_{j \neq i} v_{xij} \cdot d_{xi} \cdot d_{xj}^T \cdot O_{xi} \cdot O_{xj} \quad \text{----- (D)}$$

in which the term  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$  is same as describe as mentioned above.

### Correctness of the Scheduling of Several Processes

It is necessary to establish that activation of the  $i$ -th neuron belonging to group  $x$  will inhibit activation of the  $i$ -th neuron belonging to any other group  $y$  ( $\forall y \in [1,p], y \neq x$ ). if this were not so ,the transmission instants referring to different pairs of processes might coincide. This condition is

$$\frac{\gamma}{2} \cdot \sum_x \sum_i \sum_{y \neq x} O_{xi} \cdot O_{yi} \quad \text{----- (E)}$$

#### 4.11 Determination of Weight and Biases

If we consider equation (A) which represent the dynamics of the neural model . Taking in to account previously discussed Lyapunov function, equation (A) can be written as

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{\delta E}{\delta O_{xi}} \quad \text{----- (F)}$$

Replacing this expression of the energy function which takes into account the surrounding conditions, obtained by summing the terms in equations ( C ),( D ) and ( E)

$$E = \frac{\alpha}{2} \cdot \sum_x \left( \sum_i O_{xi} - n_{ix} \right)^2 + \frac{\beta}{2} \cdot \sum_x \sum_i \sum_{j \neq i} v_{xij} \cdot d_{xi} \cdot d_{xj}^T \cdot O_{xi} \cdot O_{xj} + \frac{\gamma}{2} \cdot \sum_x \sum_i \sum_{y \neq x} O_{xi} \cdot O_{yi} \quad \text{----- (G)}$$

And equation ( F ) becomes

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} + \alpha \cdot n_{ix} - \alpha \cdot \sum_j O_{xj} - \beta \cdot v_{xij} \cdot d_{xi} \cdot d_{xj}^T \cdot \sum_{j \neq i} O_{xj} - \gamma \cdot \sum_{y \neq x} O_{yi} \quad \text{----- (H)}$$

By comparing equation ( A ) and ( h ) the following weights and bias

$$w_{xi,yj} = -\alpha \cdot \delta_{xy} - \beta \cdot v_{xij} \cdot d_{xi} \cdot d_{xj}^T \cdot \delta_{xy} (1 - \delta_{ij}) - \gamma \cdot \delta_{ij} \cdot (1 - \delta_{xy}) \quad \text{----- (I)}$$

currents are obtained

$$I_{ix} = \alpha \cdot n_{ix} \quad \text{----- (J)}$$

We see that the value of weights  $w_{xi,yj}$  depends of he term  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$ . This implies that the effect of term  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$  may cause asymmetry in the distribution of the values of weights among the various processes. In such cases, the values of this term

would be normalized between two values MIN and MAX. More specifically prior to weights calculation, the minimum and maximum value of the term  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$ ,  $\forall i, j$  can be assumed.

The relation then calculates the actual value of the term  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$  for each pair  $P_{px}/P_{cx}$ :

$$v_{xij} \cdot d_{xi} \cdot d_{xj}^T = \text{MIN} + (\text{MAX} - \text{MIN}) \cdot (v_{xij} \cdot d_{xi} \cdot d_{xj}^T / \text{min}_x) / (\text{max}_x - \text{min}_x) \quad \text{-----} (K)$$

On the basis of the value given by equation (K), the weights  $w_{xi,yj}$  are calculated in accordance with (J).

#### 4.12 Tuning of the Neural Model

One limit to the use of a Hopfield network in our proposed scheduling solution strictly depends on the value of the coefficients relating to the surrounding conditions ( $\alpha$ ,  $\beta$ ,  $\gamma$ ). The solution may be far from optimal or even incorrect if the surrounding condition coefficients are not determined correctly, hence to solve an optimization problem with Hopfield networks are the tuning of these coefficients. This can be done through analysis to estimate the effect of each coefficient on the energy function. Here we will give some guidelines for determining the parameters  $\alpha$ ,  $\beta$ ,  $\gamma$  which characterize the energy function of the neural model proposed.

The role of the parameter  $\alpha$  is a dual one. It affects the shape of the energy surface, which is characterized by the presence of valleys since the quadratic energy function in equation (B) is continuous and differentiable. Each valley can have certain number of low points, each corresponding to a local minimal state. In order to ensure that every valley has only one low point corresponding to a minimal state. This will provide a graceful descent along the energy function. For this we require:



$$\frac{\delta^2 E}{\delta O_{xi}^2} > 0 \quad \text{----- (K)}$$

According to equation ( G ), the second derivative of Energy Function is

$$\frac{\delta^2 E}{\delta O_{xi}^2} = \alpha$$

On the basis of previous considerations, a value of  $\alpha > 0$  provides a graceful descent along the energy surface .Let us consider this initial state with the inputs of all the neurons being set to zero. For each neuron with index xi, the input increases at a rate of

$$\frac{dU_{xi}}{dt} |_{initialstate} = \alpha \cdot n_{ix} > 0 \quad \text{----- (L)}$$

This means that a high  $\alpha$  value force the network to set  $n_{ix}$  neurons for each group x to 1 from the start of the start of the neural iteration.

By equation ( L ) we can conclude that a very high  $\alpha$  value does not allow the network to modify the positions of neurons activated from the start of the neural iteration.

As for as choice of parameters  $\beta$  and  $\gamma$  is concerned by experiments it is shown that an increase in  $\beta$  has a limit linked to the presence of the term  $\gamma$  . For the p pairs of processes  $P_{px}/P_{cx}$  not to share the same transmission instants the relation  $\gamma > \beta \cdot \max_{x,i,j} (v_{xij} \cdot d_{xi} \cdot d_{xj}^T)$  must always hold. The upper limit for the term  $\beta$  is

therefore  $\beta < \frac{\gamma}{\max_{x,i,j} (v_{xij} \cdot d_{xi} \cdot d_{xj}^T)}$  . On the basis of normalization performed by

equation ( K ), the upper limit for the term  $\beta$  can also be written as  $\beta < \frac{\gamma}{MAX}$  .

### 4.13 Example of Neural Scheduling

This example is a simulation test performed on EasyNN neural network simulator, the values of the  $\alpha$ ,  $\beta$  and  $\gamma$  terms were fixed according to guidelines mentioned previously.  $\alpha = 100$ ,  $\beta = 500$ ,  $\gamma = 4$ , and considering a value of 5 for D in each Tbl. The values of MIN and MAX were fixed to 1 and 50 respectively. The test performed converged to states corresponding to correct scheduling solution within 100 to 300 iterations.

Tbl shows the neural solution to the scheduling problem in which  $p = 5$ ,  $T_s = 12$  and Table given below give summary of the time characteristics of the producer/consumer processes considered.

**Table : Time Charecteristics**

<u>x</u>	<u>T<sub>px</sub></u>	<u>rrpx</u>	<u>rcx</u>	<u>ntx</u>	<u>np<sub>x</sub></u>
1	5	0	5	3	12
2	8	0	8	2	3
3	8	1	7	2	3
4	9	0	9	2	4
5	9	2	8	2	4

**Neural solution to the Scheduling problem in which  $p = 5$  and  $T_s = 12$  :**

	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	1	0	0	0	1	0	0

2	0	0	0	0	1	0	0	0	0	0	0	1
3	1	0	0	0	0	0	0	1	0	0	0	0
4	0	0	0	1	0	0	0	0	0	1	0	0
5	0	1	0	0	0	0	0	0	1	0	0	0

#### 4.14 Neural versus Classical Scheduling

Verification of the neural model proposed to schedule several processes is not enough demonstrate the power of the neural solution. It is necessary to compare its performance with the algorithmic scheduling implementation (solution). Such a comparison cannot be made on the basis of the quality of the scheduling solution provided by the two approaches. Both ensure correct scheduling. Assessing the order of magnitude of the time required to obtain a solution in the two approaches. Evaluation will refer to the programming complexity of the neural model and the computational complexity of the classical algorithm.

##### 4.14.1 Programming Complexity

The computational time needed by the neural model to be obtain a valid scheduling solution is expected to be very short if we refer to a parallel hardware implementation of the model. The use of a hardware implementation of a Hopfield neural network drastically reduce the time required to compute the neural output , given the high calculation power obtainable ( about  $10^{11}$  connection per second) .

The time required for the network to reach a solution to the optimize problem, may be longer. It essentially depends on the number of iterations needed to reach a minimum of the energy function. The iterations is always very limited (few hundred iterations) it is seen during the simulating some scheduling problem with the help of **EasyNN** a neural network simulator, valid solution were always obtained within 250 iterations at the most.

Programming complexity refers to the calculation of the matrix of weights and bias currents linked to optimization problem to be solved, because a neural network solution requires an initial phase in which the weights  $w_{xi,yj}$  and the bias currents  $I_{xi}$  are calculated. In this case the time required to get a neural solution of the problem is practically equal to the time needed to determine the weights matrix  $W = [w_{xi,yj}]$  and the vector of bias currents  $I = [I_{xi}]$ . This complexity calculation involves the following operations.

Calculation of Tbl.

Calculation of the minimum and maximum of  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$ .

Calculation of the bias currents.

Calculation of the weights.

Calculation of Tbl is defined as  $[Tbl_x]$  matrix ( $x = 1 \dots p$ ). It is calculated by determining each matrix  $Tbl_x$  for each process. The complexity of this calculation is proportional to both the number of rows ( $n_{px}$ ) and the number of columns ( $Ts$ ), i.e. to  $Ts \cdot n_{px}$ . This implies that the calculation of the whole table Tbl is also proportional to the number of process ( $p$ ). The programming complexity therefore proportional to  $ts \cdot n_{px} \cdot p$ .

Calculation of the minimum and maximum of  $v_{xij} \cdot d_{xi} \cdot d_{xj}^T$  is done for each table  $Tbl_x$ . This is done with the help of equation (K). This calculation has a complexity proportional to  $Ts^2$  for each process, and for the total process it is proportional to  $(p \cdot Ts^2)$ .

Programming complexity of calculation of the bias current is proportional to the number of neurons, i.e.  $(p \cdot Ts)^2$ .

From the above discussion it can be concluded that the programming complexity is

$$(p \cdot Ts)^2 \text{ ----- } (M)$$

#### 4.14.2 Computational Complexity

The aim of scheduling is to determine the order in which one or more resources are assigned to particular periodic tasks in such a way that they meet their time constraints. Here we are using the timing constraints of a periodic task,  $P_x$  as follows

$T_i$  represents the task period of  $i$ th group.

$R_x$  represents the release time, i.e. the instant starting from which the task can be assigned to a resource.

$D_x$  is the deadline, i.e., the instant by which the task has to be assigned to the resource.

$C_x$  is the runtime, i.e., the time interval during which the task is assigned to the resources.

The aim of scheduling is to assign each task,  $P_x$ , to the resource no earlier than its release time and to complete it no later than its deadline. This condition has to be met in each task repetition period.

In classical scheduling (ERMM, a static scheduling), solutions vary according to the conditions surrounding the problem. The characteristics of the periodic processes involved are known in advance. The transmission instants of all the periodic processes can be determined off-line, before the communication network starts functioning.

In the polling algorithm (ERMM) described above can fix a schedule for the instance of a given set of tasks within a time interval ranging between 0 and the LCM of the task period when applied to a producer/consumer (centralized access control) communication system. This algorithm gives a set,  $S_i$ , of transmission to be scheduled in the interval with a length of  $T_s$ , equal to the LCM of all the periods being considered. Computational complexity of ERMM algorithm is proportional to  $(p^2)$ . If some appropriate data structures are used for searching then this complexity can be reduced proportional to  $(p \cdot \log p)$

$$(p \cdot \log p) \text{ ----- } (N).$$

#### **4.14.3 Comparison Between Computational and Programming Complexity**

The comparison between equation ( M ) and ( N ) shows that the neural approach is comparable with a classical algorithmic solution , if we take length of primary cycle is equal to the LCM of the periods. If number of processes are large in number then neural network solution shows a remarkable advantage in terms of comutational time.

## CONCLUSION

---

It is advantageous to use neural network techniques in process scheduling. The scheduling problem, which is dealt with a time constraint known as primary cycle, whose length is less than the LCM of all the periods. It is shown that it can be solved by classical scheduling strategies also but it need exploration of all the possible solutions until it finds the optimum one. It is difficult to solve the problem by classical methods when the number of processes to be scheduled is very high.

Hopfield model provide the correct scheduling solution in times that are proportional to the size of the problem itself. This is advantageous in real use, as its computation times are always bounded, even when there are a large number of processes.

Hopfield model also allows the scheduling table to be modified in real time when new periodic processes are added to the process control system or when existing processes modify their time requirements. This facility provides dynamic processes scheduling.

It is important to point out that scheduling table updating is not a frequent event in process control systems, still, fast adaptation is required in order to allow reconfiguration after control system fault or after sudden change in some deadline criteria.

## REFERENCES

---

- [1] German Institute of Normalization, "Profibus Standard Part 1 and 2", DIN 19 245, 1991.
- [2] French Association for Standardization, "Bus FIP for Exchange of Information between Transmitters, Actuators and Programmable Controllers", NF C46, documents 601-607, 1990.
- [3] P.Pleinevaux, J.D.Decotignie, "Time Critical Communication Networks: FieldBuses", IEEE Network Magazine, Vol.2, No.3, 1998.
- [4] P Raja, G.Noubir, "Static and Dynamic Polling Mechanism for FieldBus Networks". ACM Operating System Review, 27(1), 1993.
- [5] C.L.Liu, J.W.Layland, "Scheduling Algorithms for Multiprogramming in Hard Real Time Environment", JACM, 20(1):46,61, 1973.
- [6] Minai, A.A., and Williams, R.D., "Acceleration of Back-Propagation through Learning Rate and Momentum Adaptation", International Joint Conference on Neural Networks, Volume 1, January 1998.
- [7] Ahmad, Z., "Improving the Solution of the Inverse Kinematic Problem in Robotics Using Neural Networks", Journal of Neural Network Computing, Vol. 1 Num. 4, Spring 1990.
- [8] Carpenter, Gail, A., Grossberg, Stephen, and Rosen, D.B., "ART-2A: An Adaptive Resonance Algorithm for Rapid Category Learning And Recognition", Neural Networks, Volume 4, 1991.



- [9] Gaudiano, P., and Grossberg, Stephen, "Vector Associative Maps: Unsupervised Real-Time Error-Based Learning and Control of Movement Trajectories", Neural Networks, Volume 4, 1991. .
- [10] Hecht-Nielsen, Robert, Neurocomputing, Addison-Wesley, ISBN 0-201-09255-3, 1990.
- [11] Klimasauskas, Casimir, "Applying Neural Networks: Parts I-VI", PC AI, January-December 1991.
- [12] Miller, W. Thomas, Sutton, Richard S., and Werbos, Paul J., Neural Networks for Control, MIT Press, ISBN 0-262-13261-3, 1991.
- [13][http:// www..gcar.dem.ist.utl.pt/pessoal/sousa/measurment.pdf](http://www.gcar.dem.ist.utl.pt/pessoal/sousa/measurment.pdf).
- [14]<http://www.mel.gov/msidlibray/doc/luis.pdf>.
- [15]<http://tao.doc.wustl.edu/~corsaro/resources/paprs/bias98.pdf>.
- [16] Lakhmi C. Jain, V. Rao Vemuri , Industrial Applications of Neural Networks CRC Press Washington, D.C.
- [17] S. Cavalieri, O. Mirabella ( 1996) Neural Networks for Process Scheduling in Real-Time Communication Systems, IEEE Transection on Neural Networks, vol7, no.5.
- [18] Philip D. Wasserman ANZA reaserch Inc., Van Nostrand Reinhold, New York.
- [19]<http://www.bolton.ac.uk/technology/mind/paderborn/bus-systems/fieldbus/fieldbus.html>.