

STRUCTURE & DECISION IN MOBILE WIRELESS SENSOR NETWORKS

by

PRASANNA MOHAN BALLAL

Presented to the Faculty of the Graduate School of  
The University of Texas at Arlington in Partial Fulfillment  
of the Requirements  
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2005

Copyright © Prasanna Ballal 2005

All Rights Reserved

## ACKNOWLEDGEMENTS

I would like to thank many people who have helped me make this thesis possible. First of all, I sincerely thank my supervising professor, Dr. Frank Lewis for his invaluable guidance and constant motivation. It is because of his deep involvement in my thesis work, I could make this thesis possible.

I would like to thank my supervising committee consisting of Dr. Dan Popa and Dr. Qilian Liang for their interest in my thesis work and for taking their time to be on the thesis committee. I am grateful to all the ARRI-ACS members and especially Ms. Sarah Densmore and Dr. Sajal Das for their support and help.

Above all, I thank my mother Kalanidhi Ballal and my father Mohan Ballal for believing in my abilities, constantly providing the support I needed and being the greatest teachers of my life.

Last but not the least; I also would like to thank Mr. Vincenzo Giordano for his much needed suggestions during my work tenure at Mobile Wireless Sensor Networks Lab at Automation and Robotics Research Institute (ARRI- UTA).

This work was sponsored by ARO grant DAAD 19-02-1-0366, NSF grants IIS-0326505 and CNS-0421282, Singapore SERC TSRP grant 0421120028 and JC Penney, Inc.

July 18, 2005

## ABSTRACT

### STRUCTURE & DECISION IN MOBILE WIRELESS SENSOR NETWORKS

Publication No. \_\_\_\_\_

Prasanna Mohan Ballal, M.S

The University of Texas at Arlington, 2005

Supervising Professor: Dr. Frank Lewis

This work concerns the development of a novel structure of Discrete Event Controller (DEC) for mobile wireless sensor networks consisting of mobile robots and stationary wireless sensing nodes known as Unattended Ground Sensors (UGS). These are the three contributions:

1. Decision-making systems such as Fuzzy Logic and Dempster Shafer Theory for sensor fusion are implemented.
2. Decision-making is used for a special case of routing of resources for multiple tasks performed by different resources.
3. The DEC was actually implemented in the Mobile Wireless Sensor Networks lab at Automation & Robotics Research Institute.

This work was supported by ARO grant DAAD 19-02-1-0366, NSF grants IIS-0326505 and CNS-0421282, Singapore SERC TSRP grant 0421120028 and JC Penney, Inc. The theory involved in developing such systems is included and aptly supported by relevant simulation and experimental results.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iii
ABSTRACT .....	iv
LIST OF ILLUSTRATIONS.....	viii
LIST OF TABLES.....	xi
Chapter	
1. INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 Objective.....	2
2. DECISION SYSTEMS.....	5
2.1 Introduction.....	5
2.2 Bayesian Theory.....	5
2.3 Dempster Shafer Theory.....	9
2.3.1 Definition of Terms.....	11
2.3.2 Rule of Combination.....	13
2.4 Fuzzy Logic Systems.....	16
2.4.1 Fuzzy Logic Architecture.....	16
2.5 Petri Nets.....	21

3. DISCRETE EVENT CONTROLLER (DEC).....	25
3.1 Introduction.....	25
4. IMPLEMENTATION OF DEC ON WSN.....	30
4.1 Introduction. ....	30
4.2 Implementation on Sentries & UGS.....	38
4.3 Simulation and experimental results.....	45
4.4 Decision for routing of resources.....	48
5. SENSOR FUSION.....	50
5.1 Introduction.....	50
5.2 Fuzzy Logic.....	52
5.3 Dempster Shafer.....	55
5.4 Analysis of Dempster Shafer and Fuzzy Logic.....	58
6. CONCLUSION.....	63
6.1 Conclusion.. ....	63
6.2 Future Scope .....	64
Appendix	
A. IMPORTANT LABVIEW BLOCKS.....	65
REFERENCES .....	69
BIOGRAPHICAL INFORMATION.....	74

## LIST OF ILLUSTRATIONS

Figure	Page
2.1 A Simple Bayesian Network.....	8
2.2 Probability, Belief and Plausibility of A and B.....	13
2.3 Combination of evidence .....	14
2.4 Fuzzy Logic Architecture.....	17
2.5 Membership functions.....	19
2.6 FL system with triangular and Gaussian membership functions .....	21
2.7 A simple Petri net.....	22
2.8 Tokens in Petri net.....	22
2.9 Transitions in Petri net .....	23
2.10 Two transitions in a Petri net.....	23
2.11 Conflict in a Petri net.....	24
2.12 Timing in a Petri net.....	24
4.1 Complete System Architecture.....	31
4.2 Sequencing of missions.....	32
4.3 Matrix formulation .....	33
4.4 Reallocation of resources through matrix operations.....	34
4.5 Initial priorities.....	35
4.6 Changed priorities .....	35



4.7	Initial resource matrix .....	37
4.8	Changed resource matrix.....	37
4.9	Mission1 job sequencing matrix $F_v^1$ (a), resource requirement matrix $F_r^1$ (b).....	43
4.10	Mission1 Task start matrix $s_v^1$ (a) and resource release matrix $s_r^1$ (b) .....	43
4.11	Mission2 Task sequencing matrix $F_v^2$ (a), resource requirement matrix $F_r^2$ (b) and conflict resolution matrix $F_{ud}^2(R1)$ (c).....	44
4.12	Mission2 Task start matrix $s_v^2$ (a) and resource release matrix $s_r^2$ (b).....	45
4.13	Overall monitoring operation- Matrix formulation matrices $F_v$ , $F_r$ , $S_v$ , $S_r$ ....	45
4.14	Simulation results Mission 1 (a) Mission 2 (b).....	46
4.15	Utilization time trace of the WSN- Experimental results .....	47
5.1	Sensor Fusion Architecture .....	51
5.2	FL Membership function editor in LabView .....	53
5.3	Rulebase editor in LabView .....	53
5.4	Testing FL system in LabView .....	54

5.5	FL system block diagram in LabView .....	55
5.6	FL controller system block diagram in LabView.....	58
5.7	When Light values go low .....	59
5.8	BPA of Intrusion increases.....	60
5.9	BPA of Earthquake increases .....	60

## LIST OF TABLES

Table	Page
2.1 Sample Table for Bayesian Network .....	8
2.2 Sample Table for Bayesian Network for different probability values .....	9
4.1 Mission 1 Task sequence.....	41
4.2 Mission 1 Rule-base.....	42
4.3 Mission 2 Task sequence.....	42
4.4 Mission 2 Rule-base.....	42
5.1 Table for two sources of evidence.....	57

## CHAPTER 1

### INTRODUCTION

#### 1.1 Introduction

There has been increased research interest in systems composed of multiple autonomous mobile robots and stationary wireless sensors exhibiting cooperative behavior. Emphasis has been put on developing wide area distributed wireless sensor networks with self-organization capabilities to cope with sensor failures, changing environmental conditions, and different environmental sensing applications [19]. In particular, mobile sensor networks hold out the hope to support self-configuration mechanisms, guaranteeing adaptability, scalability and optimal performance, since the best network configuration is usually time-varying and context dependent.

Decision-making plays an important role in mobile wireless sensor networks. There are three levels of decision-making in sensor fusion namely, low level, intermediate level and high-level sensor data fusion. This thesis is mainly concerned with the decision making which involves combining decisions coming from several experts. By extension, one speaks of decision fusion even if the experts return a confidence (score) and not a decision. To distinguish both cases, one speaks of hard and soft fusion. Decision fusion can be achieved using voting methods, statistical methods, Dempster Shafer, Fuzzy logic based methods, etc.

Decision-making systems are very important in sensor fusion since it involves lot of uncertainty. Mobile wireless sensor networks present a range of challenges as they are closely coupled to the physical world with all its unpredictable variation, noise, and asynchrony; they involve many energy-constrained, resource-limited devices operating in concert; they must be largely self-organizing, self-maintaining and robust despite significant noise, loss, and failure.

## 1.2 Objective

The objective of this thesis is to study and implement decision-making systems on Mobile Wireless Sensor Network test bed at Automation and Robotics Research Institute which uses an efficient matrix-based Discrete Event Controller (DEC) [20]. In this thesis a presentation of a DEC for the coordination of cooperating heterogeneous wireless sensors, namely UGS and mobile robots is made. Contributions are also made in sensor fusion and decisions for routing of resources.

Advancements in wireless communications, Microelectromechanical Systems (MEMS), and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate undeterred in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, can be used along with mobile robots to attain a common goal. These sensor nodes are called Unattended Ground Sensors (UGS) and they along with the mobile robots form a part of the Smart Environment [19]. Smart environments represent the next evolutionary development step in building, utilities, industrial, home,

shipboard, and transportation systems automation. Like any sentient organism, the smart environment relies first and foremost on sensory data from the real world. Sensory data comes from multiple sensors of different modalities in distributed locations. The smart environment needs information about its surroundings as well as about its internal working which is similar to biological systems. An example of Smart Environment would be a warehouse guarded by mobile robots constantly interacting with strategically placed ground sensors to keep the risk of fire, trespass, robbery, etc to a minimum.

Different techniques are available to coordinate the task assignment and resource dispatching in mobile wireless sensor networks such as decentralized and centralized techniques. In decentralized coordination, the robots and sensors only have information about their local neighbors and do not have complete information of the entire network. Whereas, in centralized technique, a supervisor controls the coordination of the robots and sensors, in the decentralized approaches, robots possess similar functionalities, perform similar tasks and just one mission at a time is usually implemented. To overcome the inherent limitations of decentralized approaches, supervisory (centralized) control techniques are preferred. In [10], a supervisory control is proposed which reschedules the mission planning in response to uncontrollable events (node failures) using computationally efficient algorithms. Also the use of a coordinator can ensure that the group possesses certain desired properties and remains within the bounds of pre-specified behavioral constraints. Some significant results in supervisory control have also been obtained using Petri nets [16]. Nevertheless the

implementation of high-level mission specifications is not straightforward, the dynamical description of the system is incomplete and a new design stage, almost from scratch, is required if objectives or resources change. Thus, there is a lack of supervisory control and decision-making techniques, which can sequence different missions according to the scenario (adaptability) and reformulate the mission if some of the robots fail (fault tolerance) in a predictable way and using a high-level interface.

DEC was first used in manufacturing systems [22] in order to sequence the most suitable tasks for each agent according to the current perception of the environment. A novel matrix formulation makes the assignment of the mission planning straightforward and easily adaptable if agents or applications change. It represents a complete dynamical description of the system that allows computer simulation analysis. The matrices are direct to write down given the sequence rules for a given task. Priority rules for efficiently dispatching shared resources and handling simultaneous missions can also be easily taken into account.

This thesis includes the study of various decision-making techniques such as Bayesian Theory, Dempster Shafer Theory, Fuzzy logic and Petri Nets. It also introduces the theory of matrix Discrete Event Controller (DEC) and its simulation on wireless sensor networks. Implementation of decision-making in Sensor Fusion is also studied. Also, simulation and implementation of decision-making for routing of resources in the DEC is studied.

## CHAPTER 2

### DECISION SYSTEMS

#### 2.1 Introduction

There are different decision-making techniques, which can be used for sensor data fusion. All these techniques can be organized into two categories, i.e. statistical based and rule-based. The former class contains problem in which there is genuine randomness in the world. Playing card games such as bridge and blackjack is a good example of this class. Although in these problems it is not possible to predict the world with certainty, some knowledge about latter second class contains problems in which the relevant world is not random; it behaves “normally” unless there is some kind of exception. An important goal for many decision-making systems is to collect evidence as the system goes along and to modify its behavior on the basis of this evidence. To model this behavior one needs a statistical theory of evidence.

#### 2.2 Bayesian Theory

One needs a system that is adaptive as the system goes along collecting more evidence. Bayesian statistics is one such theory. The fundamental notion of Bayesian statistics is that of conditional probability. Suppose there is an expression about the probability of a hypothesis  $H$  given that one has observed evidence  $E$ . To compute this there is a need for prior probability of  $H$  and the extent to which  $E$  provides the



evidence of  $H$ . For this, one needs to define a universe that contains an exhaustive, mutually exclusive set of  $H_i$ 's among which one can discriminate.

$P(H_i/E)$ = probability that  $H_i$  is true given  $E$ .

$P(E/H_i)$ = probability that we observe  $E$  given  $H_i$  is true.

$P(H_i)$ = the a priori probability that hypothesis  $i$  is true in the absence of any specific evidence. These probabilities are called prior probabilities or priors.

$k$ = number of possible hypothesis.

Then the Bayes' theorem states that,

$$P(H_i / E) = \frac{P(E / H_i).P(H_i)}{\sum_{n=1}^k P(E / H_n).P(H_n)} \quad (2.1)$$

The key to using Bayes' theorem as a basis for uncertain reasoning is to recognize exactly what it says. Specifically,  $P(A/B)$  describes the conditional probability of  $A$  given that the only evidence available is  $B$ . If there is also other relevant evidence, then it too must be considered. Suppose in sensor network event detection, there are events such as fire, thermostat malfunction and explosion. If there are assertions such as light sensor values, temperature sensor values and sound sensor values, all these values have an impact on the probabilities of the given events. Thus, given a priori body of evidence ' $e$ ' and some new observation ' $E$ ', one needs to compute:

$$P(H_i / E, e) = P(H / E) \cdot \frac{P(e / E, H)}{P(e / E)} \quad (2.2)$$

But, in an arbitrarily complex world, the size of the set of joint probabilities required in order to compute this function grows as  $2^n$  if there are  $n$  different propositions being considered. This makes using Bayes' theorem intractable for several reasons:

1. The knowledge acquisition problem is insurmountable, too many probabilities to be provided.
2. The space that would be required to store all the probabilities is too large.
3. The time required to compute the probabilities is too large.

Nevertheless, Bayesian statistics provide an attractive basis for an uncertain reasoning system. As a result, several mechanisms for exploiting its power while at the same time making it tractable have been developed. Some of them are attaching certainty factors to rules, Bayesian networks and Dempster Shafer Theory.

In case of Bayesian networks, if one can state some number of propositions and clearly define the causal relationships between them (" $A$  causes  $B$ "), one can arrange those propositions and their interrelationships in a graph structure. Once the graph is created, probabilistic reasoning can be used to perform inference on the graph.

For example if an event such as a burglar's alarm has been set off. It could be that a particular sensor might have picked up an intruder, but the alarm might have also been set off by an earthquake (vibration sensor might have picked it up). Thus a belief network based on those three propositions, demonstrating that "intruder ( $I$ )" and "earthquake ( $E$ )" can both cause "Burglar Alarm ( $A$ )".

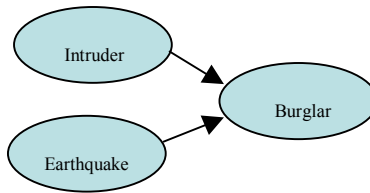


Figure 2.1 A Simple Bayesian Network

Firstly, begin by calculating the independent probabilities of intruder and earthquake, other things being equal. Then determine the conditional probability of the alarm going off in the event of all possible permutations of earthquakes and intruder alerts i.e. Intruder alert only, earthquake only, both or neither. Assume that by testing the alarm system in various ways or reviewing historical data. Now take the numbers that one has arrived at for the prior probabilities for earthquake and intruder alert and plug them into the table. For each row in the table, multiply the prior probability of each of  $I$  and  $E$  by the resulting conditional probability of an alarm to determine the actual probability of the alarm being set off in each case i.e. Calculate  $P(I)*P(E)*P(A,I,E)$ .

Construct a sample table as given below:

Table 2.1 Sample Table for Bayesian Network

$P(I)$	$P(E)$	$P(A/I,E)$	$P(A)$	$\alpha P(A)$
T=0.001	T=0.002	0.95	0.000002	0.000795
T=0.001	F=0.998	0.94	0.000938	0.372814
F=0.999	T=0.002	0.29	0.000579	0.230127
F=0.999	F=0.998	0.001	0.000997	0.396264
			0.002516	1

The symbol ' $\alpha$ ' is used for normalization i.e.  $P(A)$  is adjusted so that they add to 1. This tells that if the alarm was set off, 37% chance, it was caused by intruder, 23% was caused by earthquake, 39% for no reason at all and 0.0795% by some combination of intruder and earthquake. If it is known that there was an earthquake, one can substitute  $P(E)=1$  for true and  $P(E)=0$  for false. The table would dramatically change.

Table 2.2 Sample Table for Bayesian Network for different probability values

$P(I)$	$P(E)$	$P(A/I,E)$	$P(A)$	$\alpha P(A)$
T=0.001	T=1	0.95	0.00095	0.003269
T=0.001	F=0	0.94	0	0
F=0.999	T=1	0.29	0.28971	0.996731
F=0.999	F=0	0.001	0	0
			0.29066	1

Now the table says that 99% chance that the alarm was set off by the earthquake and only 0.33% by an intruder. This is the calculation principle employed in Bayesian networks.

### 2.3 Dempster Shafer Theory

Recently, the scientific and engineering community has begun to recognize the importance of defining multiple types of uncertainty. The dual nature of uncertainty is described by the following concepts:

**Aleatory Uncertainty:** The type of uncertainty that results from the fact that a system can behave in random ways (ex. Noise).

**Epistemic Uncertainty:** The type of uncertainty that results from the lack of knowledge about a system and is a property of the analysts performing the analysis (Subjective uncertainty).

Probability theory has been traditionally used to characterize both types of uncertainty. E.g. aleatory uncertainty can be best dealt with using the frequentist approach associated with traditional probability theory. However, probability theory is not capable of completely capturing epistemic uncertainty.

The traditional method was called Bayesian Probability. In this method, it is necessary to have information on the probability of all the events. When this is not available, uniform distribution function is often used, which means that all simple events for which a probability distribution is not known in a given sample space are equally likely. An additional assumption in the classical probability is the axiom of additivity where all probabilities that satisfy specific properties must add to 1 i.e.  $P[A] + P[\bar{A}] = 1$ .

The 3 axioms of Bayesian Theory are:

$$P[\Phi] = 0;$$

$$P[\theta] = 1;$$

$$\text{if } A \cap B = \emptyset \text{ then } P[A \cup B] = P[A] + P[B]$$

Dempster Shafer Theory offers an alternative to traditional probabilistic theory for the mathematical representation of uncertainty. The significant innovation of this framework is that it allows for the allocation of probability mass to sets or intervals. It does not require an assumption regarding the probability of the individual element of

the interval or set. An important aspect of this theory is the combination of evidence obtained by multiple sources and the modeling of conflict between them. It provides good results for evaluation of risk and reliability in engineering applications when it is not possible to obtain a correct and precise measurement from experiments. Dempster Shafer Theory ignores the 3<sup>rd</sup> Bayesian Axiom and states that,

$$P[A \cup B] \geq \text{or} \leq P[A] + P[B] - P[A \cap B]. \quad (2.3)$$

### 2.3.1 Definition of Terms

A set is represented by  $\theta$ , which contains all the possible elements of interest in each particular context, and its elements are exhaustive and mutually exclusive events. This  $\theta$  is called universal set or the frame of discernment. E.g. in tossing a fair die, the frame of discernment is  $\{1, 2, 3, 4, 5, 6\}$ . This looks similar to the sample space in probability theory, but the difference is that in DS theory, the number of possible hypothesis is  $2^{|\theta|}$  while in probability theory it is  $|\theta|$  where  $|X|$  is the cardinality of the set  $X$ . For simplicity we assume the number of elements of  $\theta$  to be finite.

**Definition 1:** If  $\theta$  is a frame of discernment then a function  $m: 2^\theta \rightarrow [0, 1]$  is called a basic probability assignment if

$$m(\Phi) = 0$$

and

$$\sum_{A \subset \theta} m(A) = 1$$

The term  $m(A)$  is called A's basic probability number and  $m(A)$  is the measure of the *belief* that is committed exactly to A. It is not required that  $m(\theta)=1$ . It is not required that  $m(A)\leq m(B)$  when  $A\subset B$ . No relationship between  $m(A)$  and  $m(\bar{A})^2$  is required. Also  $m(A)+m(\bar{A})$  does not always have to be 1.

**Definition 2:** Belief function Bel:  $2^\theta \rightarrow [0, 1]$

$$Bel(A) = \sum_{B\subset A} m(B)$$

For any  $A\subset\theta$ .  $Bel(A)$  measures the total belief of all possible subsets of A.

Properties of Belief Functions:

1.  $Bel(\Phi)=0$ .
2.  $Bel(\theta)=1$ .
3.  $Bel(A_1 \cup \dots \cup A_n) \geq \sum_i Bel(A_i) - \sum_{i<j} Bel(A_i \cap A_j) + \dots + (-1)^{n+1} Bel(A_1 \cap \dots \cap A_n)$ .

**Definition 3:** Plausibility function Pl:  $2^\theta \rightarrow [0, 1]$

$$Pl(A) = \sum_{B\cap A\neq\emptyset} m(B)$$

Also,

$$Pl(A) = 1 - Bel(A).$$

and,

$$Bel(A) \leq Pl(A).$$

Properties of Plausibility Function:

$$Pl(\Phi)=0.$$

$$Pl(\theta)=1.$$

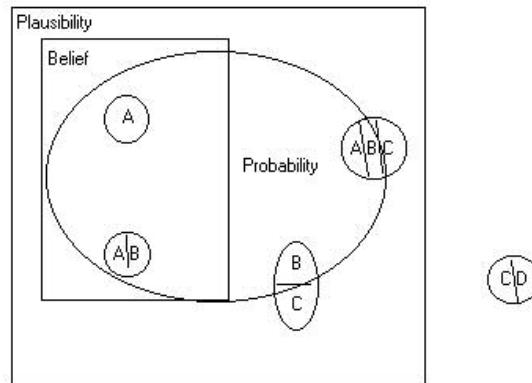
$$Pl(A_1 \cap \dots \cap A_n) \leq \sum_i Pl(A_i) - \sum_{i < j} Pl(A_i \cup A_j) + \dots + (-1)^{n+1} Pl(A_1 \cup \dots \cup A_n).$$

**Definition 4:** Doubt is defined as:

$$Doubt(A) = Bel(\bar{A}).$$

$Pl(A)$  measures the total belief mass that can move into  $A$ , whereas  $Bel(A)$  measures the total belief mass that is confined to  $A$ . Belief is also called lower probability and Plausibility is called upper probability. One can show a graphical representation the definition as follows:

If  $A$ ,  $AB$ ,  $ABC$ ,  $CD$  and  $BC$  are events and their probabilities are known, then one can represent probability, belief and plausibility of  $AB$  as:



Horizontal Lines mean that individual probabilities are known.

Figure 2.2 Probability, Belief and Plausibility of A and B

### 2.3.2 Rule of Combination

Suppose there are two distinct bodies of evidence and they are expressed by two basic assignments  $m_1$  and  $m_2$  on some focal points from power sets of the frame of



discernment. The combined two belief functions can be represented by means of a joint basic assignment  $m_{1,2}$ .

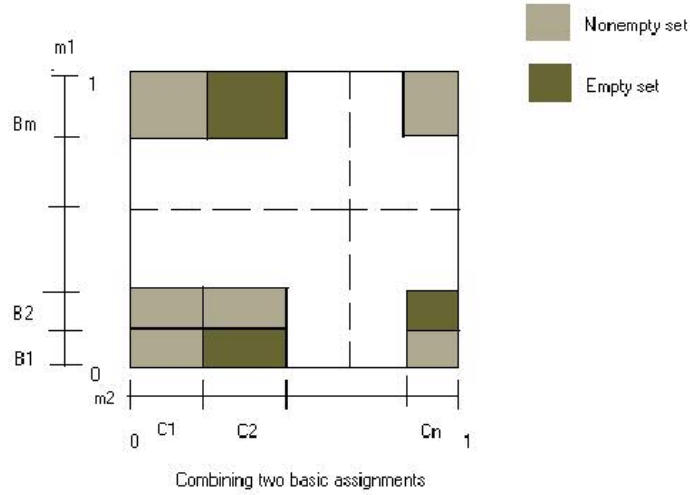


Figure 2.3 Combination of evidence

From the figure 2.3,  $m_1$  is a basic assignment over  $\theta$  and its focal points are  $B1...Bn$ . The other basic assignment  $m_2$  has focal points on  $C1...Cn$ . Then the joint basic assignment for  $A$  is given by:

$$m_{1,2}(A) = \sum_{B_i \cap C_j = A} m_1(B_i) m_2(C_j). \quad (2.4)$$

But, there is a problem in this formula when  $B_i \cap C_j = \emptyset$  for particular  $i$  and  $j$  (such as  $(i, j) = (1,2), (2,n), \dots, (m,2)$  as shown in the figure). In this case, the total area of rectangles which are now the focal elements for  $m_{1,2}$  is less than one, which violates second condition of basic probability assignment. To make the sum of  $m$ 's to be equal to 1 one has to multiply them by the factor:

$$(1 - \sum_{A_i \cap B_j = \emptyset} m_1(B_i) m_2(C_j))^{-1}. \quad (2.5)$$

Thus the rule of combination is expressed by:

$$m_{1,2}(A) = \frac{\sum m_1(B_i)m_2(C_j)}{1 - \kappa}$$

$$\text{where } \kappa = \sum_{B \cap C = \phi} m_1(B_i)m_2(C_j). \quad (2.6)$$

$\kappa$  is also called conflict. It indicates the degree of conflict between two distinct bodies of evidence. Larger  $\kappa$  indicates that there is more conflict in evidence. This rule satisfies both associative and commutative rules; i.e. if  $m_3(A)$  comes in, then use  $m_{12}$  and  $m_3$ . ( $m_{13}$ ,  $m_2$ ,  $m_{23}$ ,  $m_1$ : Order does not matter.). The main function of sensor fusion techniques is to extract more reliable and accurate information from an object using two or more sensors. Since there are not only operating limits of a sensor but also noise-corrupted environment, no single sensor can guarantee to deliver acceptably accurate information all the time. One can use Dempster Shafer Theory in sensor fusion. According to this scheme, sensor classifier produces two parameters for each hypothesis, supportability variable and plausibility variable. The difference is the ignorance of the hypothesis. Dempster Shafer's rule of combination provides a means of computing composite supportability/plausibility intervals for each hypothesis. As a result, a supportability/plausibility interval vector is obtained with a normalization vector. If one hypothesis is strongly supported by a group of sensor classifiers, then the credibility interval is reduced for that hypothesis. The normalization vector indicates the consistency of each sensor.

Potential advantage of using Dempster Shafer theory in Sensor Fusion:

1. Different level of abstraction.

2. Representation of Ignorance.

3. Conflict Resolution.

### 2.4 Fuzzy Logic Systems

Fuzzy logic (FL) is a branch of mathematics that deals with vague or gray concepts that can be said to be true to matter of degree rather than being limited to having a binary value of true or false. Fuzzy logic has been growing in use in the fields of science and engineering over the past few decades since it was first introduced to limited fanfare in 1965 by Lotfi Zadeh.

FL is different from conventional control methods as it incorporates a simple, rule based *IF X AND Y THEN Z* approach to solving a control problem rather than attempting to model a system mathematically. For example, rather than dealing with temperature control in terms such as "Temp =500F", "Temp <1000F", or "210C <Temp <220C", terms like "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)" or "IF (process is too hot) AND (process is heating rapidly) THEN (cool the process quickly)" are used.

#### *2.4.1 Fuzzy Logic Architecture*

The fuzzy controller is composed of the following four elements:

- 1) **Rule-Base:** It is a set of "If-Then" rules, which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.
- 2) **Inference Engine:** It emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant.

3) **Fuzzification**: This converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.

4) **Defuzzification**: This converts the conclusions of the inference mechanism into actual inputs for the process.

The block diagram is shown figure 2.4.

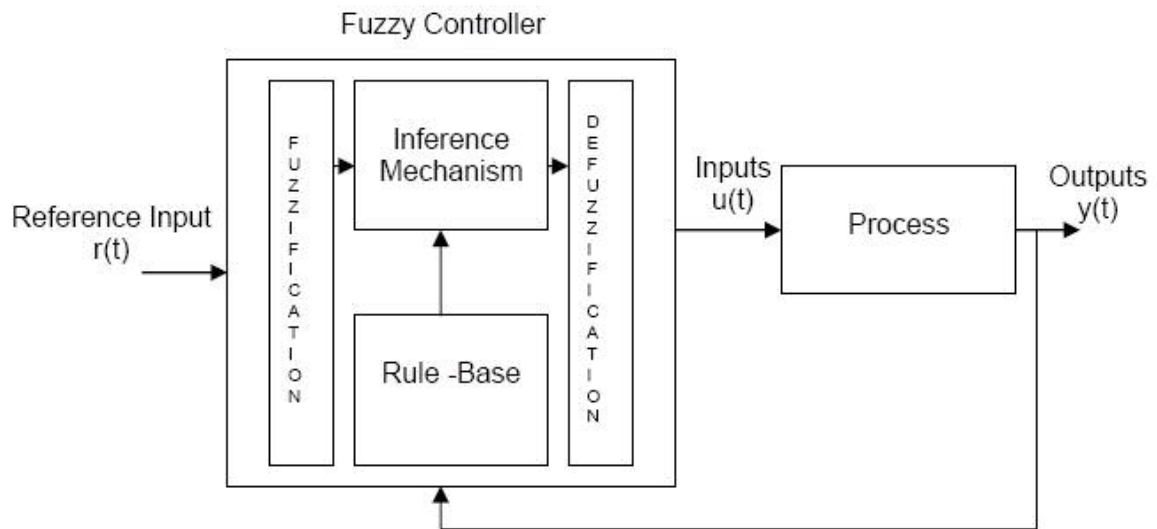


Figure 2.4 Fuzzy Logic Architecture

In 1973, Lotfi Zadeh proposed the concept of linguistic or "fuzzy" variables. Think of them as linguistic objects or words, rather than numbers. The sensor input is a noun, e.g. "temperature", "displacement", "velocity", "flow", "pressure", etc. Since error is just the difference, it can be thought of the same way. The fuzzy variables themselves are adjectives that modify the variable (e.g. "large positive" error, "small positive" error, "zero" error, "small negative" error, and "large negative" error). As a minimum, one could simply have "positive", "zero", and "negative" variables for each of the

parameters. Additional ranges such as "very large" and "very small" could also be added to extend the responsiveness to exceptional or very nonlinear conditions, but aren't necessary in a basic system. This linguistic quantification is used to specify a set of rules (a rule-base) that captures the expert's knowledge about how to control the plant. The general form of the linguistic rules is:

**If** premise **Then** consequent

Premises are also called "antecedents" and are associated with the fuzzy controller inputs and are on the left-hand side of the rules. The consequents are associated with the controller outputs and are on the right-hand-side of the rules. The number of fuzzy controller inputs and outputs places an upper limit on the number of elements in the premises and consequents. It may be noted that there does not need to be a premise (consequent) term for each input (output) in each rule, although often there is.

### **Membership Functions:**

As shown in the figure below, the set of values that is described by  $\mu_A$  and  $\mu_B$  is called a "fuzzy set". The membership function  $\mu_A(x)$  describes the membership of the elements 'x' of the base set X in the fuzzy set A, whereby for  $\mu_A(x)$  a large class of functions can be taken.

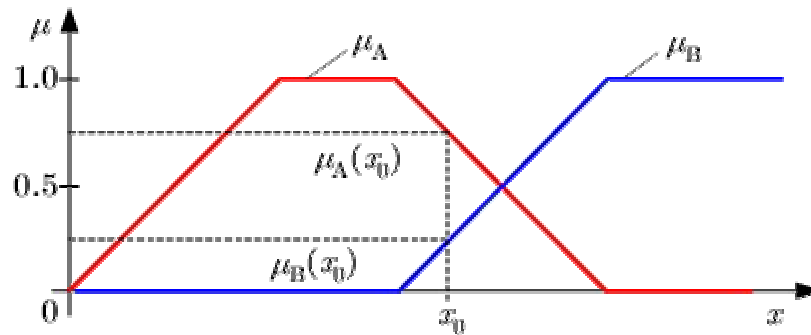


Figure 2.5 Membership functions

Reasonable functions are often piecewise linear functions, such as triangular or trapezoidal functions. The grade of membership  $\mu_A(x_0)$  of a membership function  $\mu_A(x)$  describes for the special element  $x=x_0$ , to which grade it belongs to the fuzzy set  $A$ . This value is in the unit interval  $[0, 1]$ . Of course,  $x_0$  can simultaneously belong to another fuzzy set  $B$ , such that  $\mu_B(x_0)$  characterizes the grade of membership of  $x_0$  to  $B$ . The membership function is not a probability density function, and there is no underlying probability space. The membership function does not quantify random behavior; it simply makes more accurate (less fuzzy) the meaning of linguistic descriptions. In the figure 2.5, while the vertical axis represents certainty, the horizontal axis is called the “**Universe of Discourse**” for the input  $X$  since it provides the range of values of  $X$  that can be quantified with linguistics and fuzzy sets. In summary, depending on the application and the designer (expert), many different choices of membership functions are possible.

**Fuzzification:**

Fuzzification is the process of obtaining a value of an input variable and finding the numeric values of the membership function(s) that are defined for that variable. For example if  $x = x_0$ , the fuzzification process amounts to finding the values of the input membership functions for this. In this case  $\mu(x) = 0.75$ .

### **Fuzzy Set Operations**

Consider two fuzzy sets A and B with the membership functions  $\mu_A$  and  $\mu_B$ .

- *Union* of the two fuzzy sets is defined as the maximum of the two individual membership functions. This is called the *maximum* criterion.

$$\mu(A \cup B) = \max(\mu_A, \mu_B) \quad (2.7)$$

- *Intersection* of the two fuzzy is defined as the minimum of the two individual membership functions. This is called the *minimum* criterion.

$$\mu(A \cap B) = \min(\mu_A, \mu_B) \quad (2.8)$$

Intersection of two fuzzy can also be defined as the product of the two Individual membership functions. This is called the *product* criterion.

$$\mu(A \cap B) = \mu_A \times \mu_B \quad (2.9)$$

- *Complement* of a fuzzy set is defined as the negation of the specified membership function. This is called the *negation* criterion.

$$\mu_{\hat{A}} = 1 - \mu_A \quad (2.10)$$

### **Inference:**

The inference process is used for determining the extent to which each rule is relevant to the current situation and drawing conclusions using the current inputs and the information in the rule-base. The inference process includes the fuzzy set

operations. Graphical representation of the product implication rule with triangular and Gaussian membership functions is shown in figure 2.6.

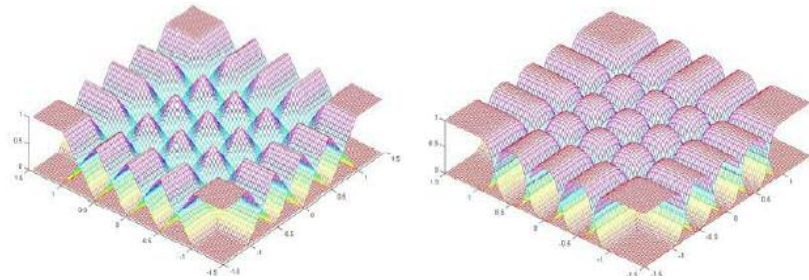


Figure 2.6 FL system with triangular and Gaussian membership functions

### Defuzzification

A number of defuzzification methods exist where each method provides a means to choose a single output based on the inference strategy employed. The most commonly used defuzzification strategy used is the “Centroid Defuzzification” which is given by the equation:

$$f(x) = \frac{\sum_{i=1}^N z^i \prod_{j=1}^n \mu_{ij}(x_j)}{\sum_{i=1}^N \prod_{j=1}^n \mu_{ij}(x_j)} \quad (2.11)$$

where the control representative values are  $z_i$  and the 1-D membership functions are  $\mu_{ij}$ .  $x_j$  are the components of the  $n$ -vector  $x$ .

### 2.5 Petri Nets

A Petri net consists of places (represented as circles), transitions (represented as bars) and place input/outputs (represented as directed arcs) as shown in the figure 2.7.



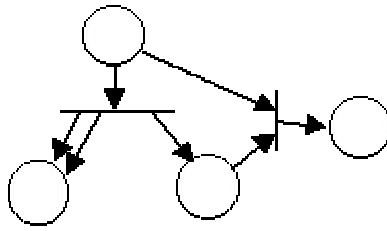


Figure 2.7 A simple Petri net

It can be used to model asynchronous systems with concurrency. The transitions represent events and the places represent states, or conditions. Inputs and outputs are allowed only between places and transitions: you cannot go directly from one place to another without a transition.

Not only can you model a system's architecture with a Petri net, you can also simulate execution of the system. This is done by "marking" the Petri net with "tokens" represented as dots in the places, as shown in the figure 2.9 .

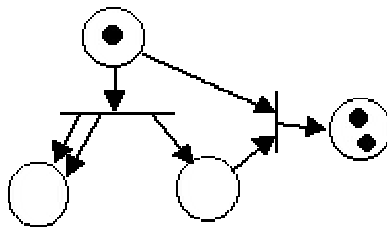


Figure 2.8 Tokens in Petri net

The rules for executing the system are:

- a transition is "active" when each of its input places contains a token

- each active transition in the diagram is "fired" by removing one token from each input place and generating one in each output place

The figure below shows the next stage in the execution of the net in the previous figure.

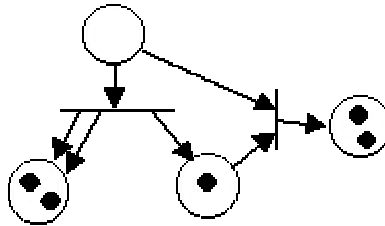


Figure 2.9 Transitions in Petri net

Systems modeled with Petri nets are inherently asynchronous, as a transition will fire as soon as it is active. Realistic simulations can build delays into the passing of tokens, but this still does not avoid conflicts. When two transitions are active, as in the figure below, they can both fire together.

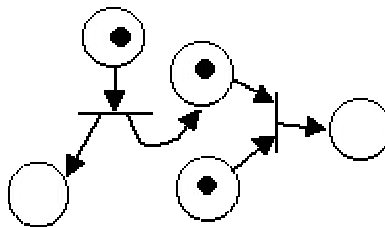


Figure 2.10 Two transitions in a Petri net

However, it may happen that the two transitions require the same token, as in figure 2.11. This is called a "conflict"

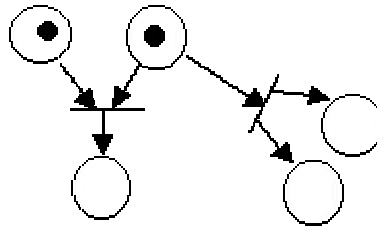


Figure 2.11 Conflict in a Petri net

This could be resolved by making the choice of which transition fires arbitrary, or by designating this "decision" point, controlled by an external source. Timing can be added to a Petri net by naming all the places and transitions, and drawing up a table with minimum and maximum times for each transition to occur based on the time of arrival of the tokens at its input places.

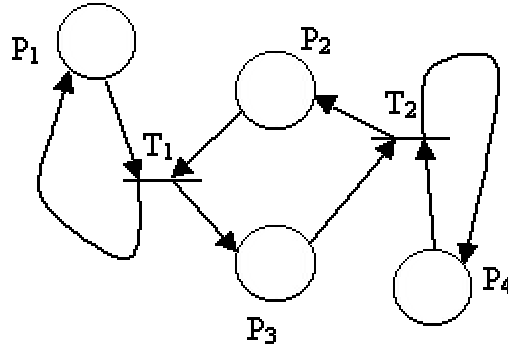


Figure 2.12 Timing in a Petri net

Thus, timing would help in decision-making in a Petri net. There is also another concept of probabilistic Petri nets, which is also useful for decision-making. Petri nets are further explained in the following chapters.

CHAPTER 3  
DISCRETE EVENT CONTROLLER (DEC)

3.1 Introduction

An efficient Discrete event controller (DEC) based on matrices was first introduced in [20] and it has been in constant development [12, 35]. The DEC is completely based on matrices and it has important advantages in design, flexibility, computer simulation and online supervisory control of DE systems. The DEC has also been implemented on a practical robotic cell in [22]. This section presents a novel matrix-based discrete event controller for modeling and analysis of complex interconnected DE systems with shared resources, routing decisions, and dynamic resource management in a mobile wireless sensor network. This approach provides a rigorous, yet intuitive mathematical framework to represent the dynamic evaluation of DE systems according to linguistic if-then rules such as “If <conditions hold> then <consequences>”.

Multi-agent systems such as one composed of mobile robots and wireless sensor network face problems of coordination. One can write down a set of if-then rules to define the mission planning of the sensor agents, such as:

Rule i: If <sensor 1 has completed task1 (data acquisition), robot 1 is available and a fire hazard is detected > then <robot 1 starts task2 and sensor 1 is released>

These linguistic rules can be easily represented in mathematical form using matrices. Following the same notation used in [20], let  $r$  be the vector of resources used in the system (i.e. mobile robots and UGSs),  $v$  the vector of tasks that the resources can perform (i.e. go to a given target, perform data acquisition, and deploy UGS),  $u$  the vector of input events (i.e. occurrence of sensor detection events) and  $y$  the vector of completed missions (outputs). Finally, let  $x$  be the state logical vector of the rules of the DE controller, whose entry of '1' in position  $i$  denotes that rule  $i$  of the supervisory control policy is currently activated. Then we can define two different sets of logical equations, one for checking the conditions for the activation of rule  $i$  (matrix controller state equation), and one for defining the consequences of the activation of rule  $i$  (matrix controller output equation). In the following, all matrix operations are defined to be in the or/and algebra, where  $+$  denotes logical or and 'times' denotes logical and.

The controller state equation is

$$\bar{x} = F_v \bar{v} + F_r \bar{r} + F_u \bar{u} + F_{ud} \bar{u}_d \quad (3.1)$$

where  $x$  is the task or state logical vector,  $F_v$  is the task sequencing matrix,  $F_r$  is the resource requirements matrix,  $F_u$  is the input matrix.  $F_{ud}$  is the conflict resolution matrix and  $u_d$  is the conflict resolution vector. They are used to avoid simultaneous activation of conflicting rules, as will be shown later. The current status of the DE system includes task vector  $v$ , whose entries of '1' represent 'completed tasks', resource vector  $r$ , whose entries of '1' represent 'resources currently available', and the input vector  $u$ , whose entry of 1 represent occurrence of a certain predefined event (fire

alarm, intrusion etc.). The over bar in equation (1) denotes logical negation so that tasks complete or resources released are represented by '0' entries.

$F_v$  is the task sequencing matrix [33], and has element (i,j) set to '1' if the completion of task  $v_j$  is an immediate prerequisite for the activation of logic state  $x_i$ .

$F_r$  is the resource requirements matrix [17] and has element (i,j) set to '1' if the availability of resource  $j$  (robot or UGS) is an immediate prerequisite for the activation of logic state  $x_i$ .

On the ground of the current status of the DE system, equation (3.1) calculates the logical vector  $x$ , i.e. which rules are currently activated. The activated rules determine the commands that the DEC has to sequence in the next iteration, according to the following equations

$$v_s = S_v x \quad (3.2)$$

$$r_s = S_r x \quad (3.3)$$

$$y = S_y x \quad (3.4)$$

$S_v$  is the task start matrix and has element (i,j) set to '1' if logic state  $x_j$  determines the activation of task  $i$ .

$S_r$  is the resource release matrix and has element (i,j) set to '1' if the activation of logic state  $x_j$  determines the release of resource  $i$ .

$S_y$  is the output matrix and has element (i,j) set to '1' if the activation of logic state  $x_j$  determines the completion of mission  $i$ .

The task start equation (3.2) computes which tasks are activated and may be started, the resource release equation (3.3) computes which resources should be released (due to completed tasks) and the mission completion equation (3.4) computes which missions have been successfully completed.

Vector  $v_s$ , whose '1' entries denote which tasks are to be started, and vector  $r_s$ , whose '1' entries denote which resources are to be released, represent the commands sent to the DE system by the controller. '1' entries in vector  $y$  denote which missions have been successfully completed.

Equations 3.1-3.4 represent the rule-base of the supervisory control of the DE system. All the coefficient matrices are composed of Boolean elements and are sparse, so that real time computations are easy even for large interconnected DE systems.

The task sequencing matrices ( $F_v$  and  $S_v$ ) are direct to write down from the required operational task sequencing. On the other hand, the resource requirements matrices ( $F_r$ ,  $S_r$ ) are written down based on the resources needed to perform the tasks and are assigned independently of the task sequencing matrices. Matrix  $F_{ud}$  in equation (3.1) is used to resolve conflicts of shared resources, i.e. conflicts deriving by the simultaneous activation of rules, which start different tasks requiring the same resource. Matrix  $F_{ud}$  has as many columns as the number of tasks performed by shared resources. Element (i,j) is set to '1' if completion of shared task  $j$  is an immediate prerequisite for the activation of logic state  $x_i$ . Then an entry of '1' in position  $j$  in the conflict resolution vector  $u_d$ , determines the inhibition of logic state  $x_i$  (rule  $i$  cannot be fired). It results that, depending on the way one selects the conflict-resolution strategy to generate vector

$u_d$ , different dispatching strategies can be selected, in order to avoid resource conflicts or deadlocks. In the conversion of linguistic rules into matrix form using DEC, the following assumptions must be considered:

1. A resource cannot be removed from a task until it is complete.
2. A single resource can be used for only one task at a time.
3. A process holds the resource allocated to it until it has all the resources required to perform a task.
4. Resource is released immediately after it has executed its task.



## CHAPTER 4

### IMPLEMENTATION OF DEC ON WSN

#### 4.1 Introduction

To use the DEC as a Supervisory Controller for task assignment and resource dispatching in mobile wireless sensor networks, one needs to have an architecture that is modular, flexible and adaptable. One such architecture consists of three layers, namely agent control layer, network control layer and organization control layer. The important aspect of this architecture is that improvements and updates on one layer results in minor changes in other layers, making the system intelligent and adaptable.

The first layer (agent control level) deals with the control of each agent (being either a UGS or a mobile robot), keeping into account its peculiar functionalities. At this level one defines the processing capabilities of the UGSs (e.g. signal processing) and the control algorithms for the behavior of each robot (e.g. reach the target, follow another robot etc.). The second layer (network control level) deals with the implementation of communication protocols for energy efficient data transmission between the UGS, robots and the supervisor. The third layer (organization control level) consists of matrix-based DE supervisory controller whose matrix formulation allows one to employ a high-level human interface to define the mission planning, the resource allocation and the dispatching rules. The supervisor is in charge of sequencing the tasks each agent has to perform according to the perception of the environment; assuming that the agent

level controllers correctly perform the assigned tasks and that the communication protocol for each agent perfectly works.

The complete architecture is shown in the figure 4.1.

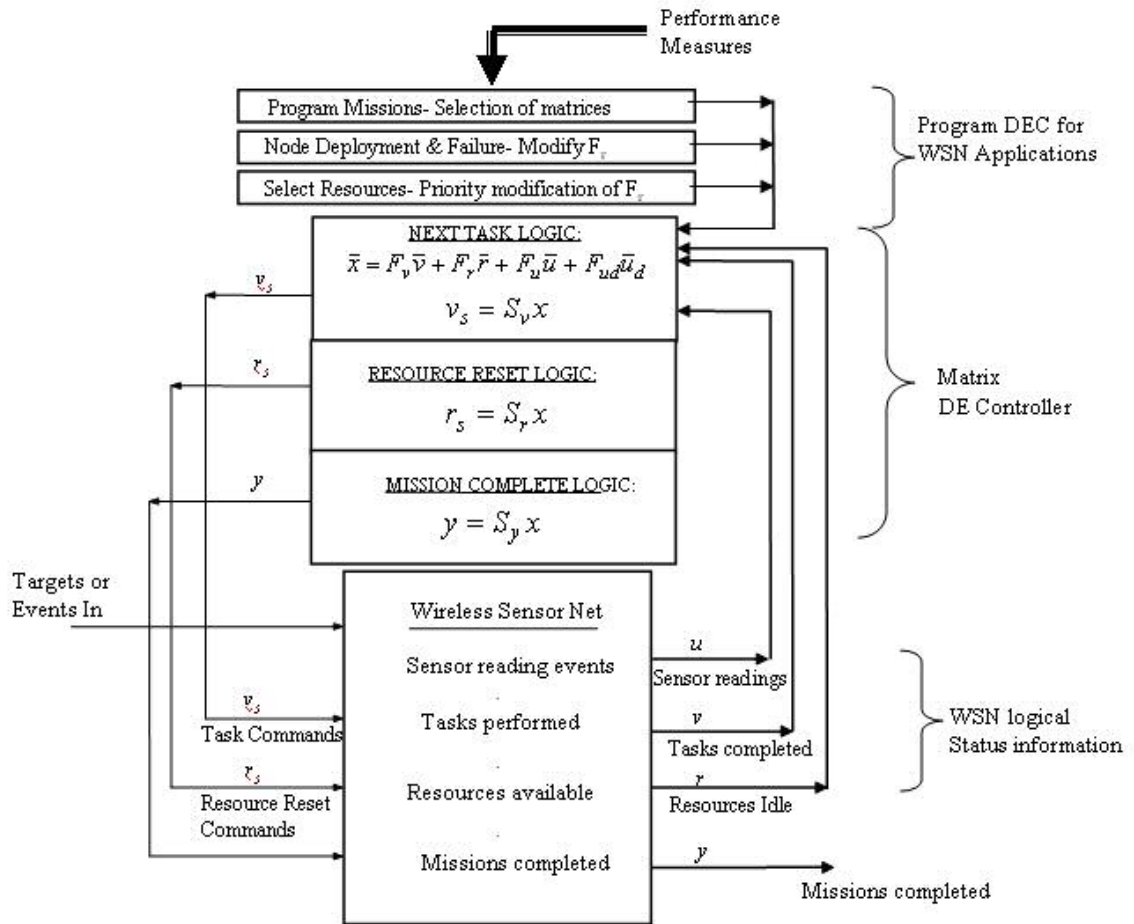


Figure 4.1 Complete System Architecture

Thus, using this architecture, a complex system can be decomposed into missions, tasks and rules for task sequencing, resource dispatching and conflict resolution (figure 4.2).

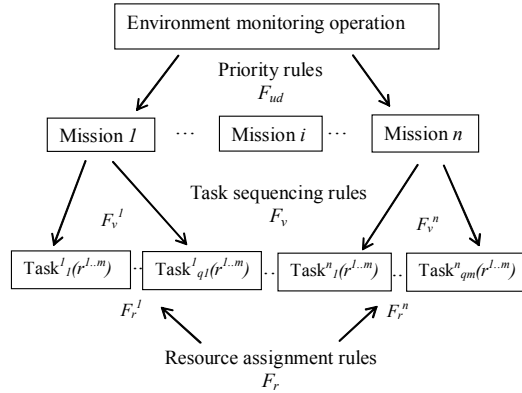


Figure 4.2 Sequencing of missions

This block diagram can be represented using matrices using the technique of DEC. Suppose that there are  $m$  resources  $r^j$   $j=1\dots m$  (mobile robots and stationary sensors) each one capable of performing  $p_j$  tasks, and define  $n$  different missions, each one composed of  $q_i$  tasks. For each mission, there are corresponding set of matrices  $F_v^i, F_r^i, S_v^i, S_r^i$  which represent the coordination rules of the agents in the execution of the tasks. In order to take into account the priority among missions, there is a global conflict resolution matrix  $F_{ud}$ . After assigning a priority order  $k$  to each mission, calculate for every resource  $j$  and every mission  $i$ , a matrix  $(F_{ud}^i(r_j))$ , creating a new column for every '1' appearing in the  $j$ th column of  $F_r^i$ . Then one constructs the global conflict resolution matrix of resource  $r_j$   $(F_{ud}(r_j))$  inserting each  $F_{ud}^i(r_j)$  matrix in position  $(i,k)$ .

As shown in figure 4.3 the matrix formulation of the overall environment monitoring operation is then obtained by stacking the set of matrices together. The correspondence between figure 4.2 and the matrices is very obvious.

$$F_{ud} = [F_{ud}(r_1) \dots F_{ud}(r_j) \dots F_{ud}(r_m)]$$

$$q_1 \quad \dots \quad q_i \quad \dots \quad q_n \quad m$$

$$F_v = \begin{bmatrix} F_v^1 & & & & \\ & \dots & & & \\ & & F_v^i & & \\ & & & \dots & \\ & & & & F_v^n \end{bmatrix} \quad F_r = \begin{bmatrix} F_r^1 \\ \dots \\ F_r^i \\ \dots \\ F_r^n \end{bmatrix}$$

Figure 4.3 Matrix formulation

In WSN, two issues have to be tackled i.e. adaptability and scalability. Adaptability and scalability are crucial requirements to guarantee optimal performances for agent team operations. These issues can be tackled by using the principle of DEC.

**Adaptability:** Adaptability is the ability of an agent team to change its behavior according to the dynamical evolution of the environment. Following methods make the system adaptable using DEC.

1. Implementation of distributed algorithms:

Adaptability can be resolved both at the agent control level and at the supervisor control level. In the agent control level, individual agents are autonomous and perform tasks using their perception of the environment. In the framework of the DEC, these operations can be considered as a generic (fully decentralized) mission  $i$  (or part of it) composed of simultaneous tasks. Therefore there is enhanced adaptability decision, at the supervisor level (on the grounds of the present situation), which decentralized mission has the priority (changing  $F_{ud}^i$ ) and which resources should be used (changing  $F_r^i$  and  $S_r^i$ ).

2. Dynamic reallocation of resources:

A dynamic reallocation of the agents to missions can be performed by rearranging the '1' relative to similar resources in the matrices  $F_r$  and  $S_r$ , when new missions (or new agents) are added. Due to the matrix representation of the mission plans, these objectives can be pursued using computationally efficient algorithms.

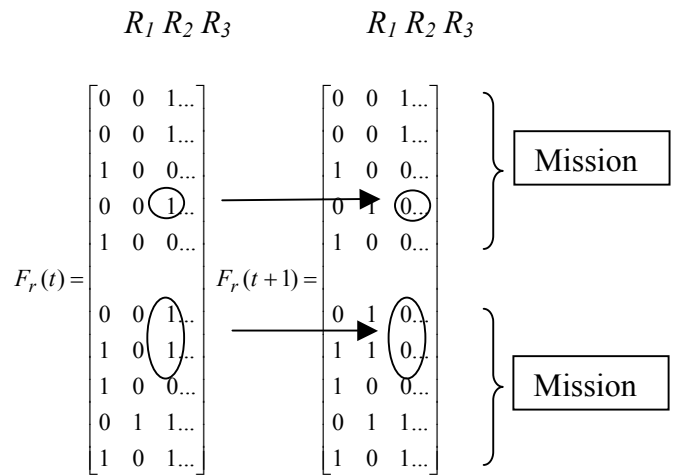


Figure 4.4 Reallocation of resources through matrix operations

Figure 4.4 shows an example of reallocation of tokens among three similar resources ( $R_1$ ,  $R_2$  and  $R_3$ ) in the case of two missions. After the reallocation, the workload of the resources is more balanced since each resource performs a similar number of tasks (equal to the number of '1' in the corresponding column).

3. Combining multiple plans for the same mission:

In certain circumstances, different sequences of tasks can be used to implement the same mission. A computationally efficient algorithm can be used to combine the plans together and derive one single compact matrix representation for the DEC. In this

way, the DEC automatically sequences the most suitable succession of tasks depending on the current available resources.

#### 4. Priority among missions:

Another way to adapt to the WSN control scenario is to make the set of mission priority rules adaptable. For example, suppose that resource  $r_l$  is shared among three different missions whose priority rank is 3, 1, 2. After defining the conflict resolution matrix of  $r_l$  for each mission ( $F_{ud}^1(r_1), F_{ud}^2(r_1), F_{ud}^3(r_1)$ ), the overall conflict resolution matrix of  $r_l$  ( $F_{ud}(r_1)$ ) is built as:

$$F_{ud}(r_1) = \begin{matrix} & \begin{matrix} \textit{priority}^1 & \textit{priority}2 & \textit{priority}3 \end{matrix} \\ \begin{matrix} \textit{mission}^1 \\ \textit{mission}^2 \\ \textit{mission}^3 \end{matrix} & \begin{bmatrix} 0 & F_{ud}^1(r_1) & 0 \\ 0 & 0 & F_{ud}^2(r_1) \\ F_{ud}^3(r_1) & 0 & 0 \end{bmatrix} \end{matrix}$$

Figure 4.5 Initial priorities

If the priority of the missions changes in 2, 3, 1 then one can have the following configuration as shown in figure 4.6.

$$F_{ud}(r_1) = \begin{matrix} & \begin{matrix} \textit{priority}^1 & \textit{priority}2 & \textit{priority}3 \end{matrix} \\ \begin{matrix} \textit{mission}^1 \\ \textit{mission}^2 \\ \textit{mission}^3 \end{matrix} & \begin{bmatrix} 0 & 0 & F_{ud}^1(r_1) \\ F_{ud}^2(r_1) & 0 & 0 \\ 0 & F_{ud}^3(r_1) & 0 \end{bmatrix} \end{matrix}$$

Figure 4.6 Changed priorities

Thus, a change of priority results in a simple permutation of the block matrices  $F_{ud}^i$  for each resource.

**Scalability:** Scalability defines the possibility to add and remove agents. One can use the DEC to tackle scalability at the supervisor level, updating the matrix based representation of the missions to take into account the failure of agents as well as the adding of new ones.

If a new agent is added to the system, a new column is added in the matrices  $F_r$  and  $S_r'$  ( $S_r$  transpose). Then, dispatching algorithms (based on matrix operations) can be applied to rearrange the tasks among resources. In a similar fashion, an agent failure can be tackled rearranging the tasks among the resources so that the column vectors relative to the failed resources in  $F_r$  and  $S_r'$  are null. In the following example, a simple algorithm is used for reallocating (off-line, i.e. when no missions are in progress) resources after agent failure. The mission planning is revised in such a way that predefined back-up agents execute the tasks of the failed agents. In the matrix formulation this is equivalent to move the elements equal to one in the matrices  $F_r^i$  and  $S_r^i$  from the column of the failed resource to the column of the back-up resource. This can be achieved through a simple linear combination of the columns of  $F_r^i$  and  $S_r^i$  respectively. Thus,

$$F_r^{i,new} = F_r^{i,old} \cdot B^i \quad (4.1)$$

$$S_r^{i,new} = S_r^{i,old} \cdot B^i \quad (4.2)$$

where  $B^i$  is a square matrix of dimension equal to the number of the resources of the system. The diagonal elements of  $B^i$  ( $a_j$ ) are parameters which are equal to 1 if resource  $r_j$  is working properly and 0 otherwise. On row  $j$  the element  $(j,j)$  is equal to

$a_j$  and the element  $(j,k)$  is equal to  $1 - a_j$ , where  $k$  is the column of matrix  $F_r^i$  corresponding to the back-up resource of agent  $j$  for mission  $i$ . If  $a_j=0$ , the  $j$ th column of  $F_r^{i,new}$  will be null (meaning that agent  $j$  is not supposed to perform any task) and the  $k$ th column will have '1's in correspondence of the tasks for which resource  $j$  was required. If no failure occurs,  $B^i$  is the identity matrix and mission plans are not changed. Clearly, in the definition of the matrix  $B^i$  one has to make sure that each back-up agent does not perform any simultaneous task with the resource they are supposed to substitute. For example, suppose there are three agents, and that, for a certain mission  $i$ , the resource requirement matrix and the back-up matrix  $B^i$  are

$$F_r^{i,old} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad B^i = \begin{pmatrix} a_1 & 1-a_1 & 0 \\ 0 & a_2 & 0 \\ 1-a_3 & 0 & a_3 \end{pmatrix}$$

Figure 4.7 Initial resource matrix

The  $B^i$  matrix corresponds to the case where, in mission  $i$ , agent 2 is the backup of agent 1, agent 2 has no back-up and agent 1 is the back up of agent 3. If agent 1 fails ( $a_1=0$ ) whereas agent 2 and 3 work properly ( $a_2=a_3=1$ ),

$$B^i = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad F_r^{i,new} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 4.8 Changed resource matrix

i.e., in mission  $i$ , agent 1 has been replaced by agent 2.



Another method to cope up with agent failure is by routing resources, which means that one has to define a set of multiple resource choices initially for certain critical tasks. The routing resources automatically assign to the task the first available resource of the corresponding set providing redundancy and robustness against agent failures. This novel matrix formulation supports task routing efficiently, since there is no need to distinguish between physical and logical resources. Routing of resources will be explained later in the chapter.

#### 4.2 Implementation on Sentries & UGS

It is well known that a matrix approach can be used to describe the marking transitions of a Petri Net using the PN transition equation

$$m(t+1) = m(t) + (S' - F) \cdot x(t) \quad (4.3)$$

where  $S$  and  $F$  are the output and input incidence matrix respectively. This equation gives a useful insight on the dynamics of discrete event systems but does not provide a complete dynamical description of DE systems.

Observe that the vector  $x$  in equation (4.3) is the same as in equation (3.1), then one may identify  $x$  as the vector associated with the PN transitions and  $u, v, r, u_d$  as associated with the places. Then it follows that,

$$m(t) = [u(t)', v(t)', r(t)', u_d(t)'] \quad (4.4)$$

$$S = [S_u', S_v', S_r', S_{u_d}', S_y'] \quad (4.5)$$

$$F = [F_u', F_v', F_r', F_{u_d}', F_y'] \quad (4.6)$$

Therefore, we can use equation (3.1) to generate the allowable firing vector to trigger transitions in equation (4.3). The combination of the DEC and the PN marking transition equation, therefore, provides a complete dynamical description of the system.

In order to take into account the time durations of the tasks and the time required for resource releases, one can split  $m(t)$  into two vectors, one representing available resources and current finished tasks ( $m_a(t)$ ) and the other representing the tasks in progress and idle resources ( $m_p(t)$ )

$$m(t) = m_a(t) + m_p(t) \quad (4.7)$$

This is equivalent to introducing timed places in a Petri net and to dividing each place into two parts, one relative to the pending states (task in progress, resource idle) and the other relative to the steady states (task completed and resource available). As a consequence, we can also split equation (4.3) into two equations

$$m_a(t+1) = m_a(t) - F \cdot x(t) \quad (4.8)$$

$$m_p(t+1) = m_p(t) + S' \cdot x(t) \quad (4.9)$$

When a transition fires a token is moved from  $m_p(t)$  to  $m_a(t)$  where it may be used to fire subsequent transitions. Therefore equations (4.3), (4.8) and (4.9) represent a complete description of the dynamical behavior of the discrete event system and can be implemented for the purposes of computer simulations using any programming language (e.g. Matlab® or C). In the case of a mobile wireless sensor network, where experiments on wide and hostile areas can be really complex and challenging, it

allows one to perform extensive simulations of the control strategies and then test experimentally only those that guarantee the most promising results.

Consider an experimental scenario; A network consisting of two mobile robots and two wireless sensors. Two different missions have been implemented to show the potentialities of the proposed DEC. In the first mission, after one of the sensors launches an intruder alert, the network automatically reconfigures its topology to further investigate the phenomenon. In the second mission, one of the sensors detects a huge vibration indicating that there is an earthquake. The procedure for implementing the supervisory control policy consists of three different steps. First of all one defines the vector of resources  $r$  present in the system and the tasks they can perform. In ARRI WSN test-bed there are two robots ( $R_1$  and  $R_2$ ), each one able of performing certain number of tasks (say 4 or 5), and two stationary sensors ( $UGS_1$ ,  $UGS_2$ ), each one able of performing one task (i.e. taking measurement). The resource vector is  $r = [R_1, R_2, UGS_1, UGS_2]$ .

Then for each mission  $i$ , define the vector of inputs  $u^i$ , of outputs  $y^i$  and of tasks  $v^i$ , and the task sequence of each mission (refer table 4.1 and 4.2 for mission 1 and mission 2), and write down the if-then rules representing the supervisory coordination strategy to sequence the programmed missions (table 4.3 and table 4.4). In the definition of the rule bases particular attention has to be devoted to the definition of consecutive tasks performed by the same resources. If the consecutive tasks are interdependent (e.g. *go to sensor 2* and *retrieve sensor 2*), the corresponding resource should be released just at the end of the last of the consecutive tasks. Instead, if the

tasks are not interdependent, before starting the new consecutive task, the DEC releases the corresponding resource and makes sure that no other missions are waiting for it. If after a predetermined period of time no other missions request that resource, the previous mission can continue. Finally translate the linguistic description of the coordination rules into a more convenient matrix representation, suitable for mathematical analysis and computer implementation. As an example, the following shows a derivation of the matrix formulation from the rule-base of mission 1 (table 4.3).

For example, considering the rule-base of mission1 (table 4.3), one can easily write down the  $F_v^1$  and  $F_r^1$  matrices considering that  $F_v^1(i, j)$  is '1' if task  $j$  is required as an immediate precursor to rule  $i$  and  $F_r^1(i, j)$  is '1' if resource  $j$  is required as an immediate precursor to rule  $i$ .

Table 4.1 Mission 1 Task sequence

<b>mission1</b>	<b>Notation</b>	<b>Description</b>
<i>Input 1</i>	$U^1$	<i>UGS1</i> launches earthquake alert
<i>Task 1</i>	$T1a$	<i>UGS2</i> takes measurement
<i>Task 2</i>	$T2a$	<i>R1</i> goes to <i>UGS1</i> and takes measurement
<i>Task 3</i>	$T3a$	<i>UGS1</i> takes measurements again
<i>Task 4</i>	$T4a$	<i>R2</i> goes to <i>UGS2</i> and takes a measurement.
<i>output</i>	$Y^1$	False Alarm, Mission 1 completed

Table 4.2 Mission 1 Rule-base

<b>Mission1-operation sequence</b>	
<i>Rule1</i> $x_1^1$	If input then T1a
<i>Rule2</i> $x_2^1$	If T1a then T2a
<i>Rule3</i> $x_3^1$	If T2a then T3a
<i>Rule4</i> $x_4^1$	If T3a then T4a
<i>Rule5</i> $x_5^1$	If T4a then y

Table 4.3 Mission 2 Task sequence

<b>Mission2</b>	<b>Notation</b>	<b>Description</b>
<i>input</i>	$U^2$	<i>UGS2</i> detects intruder
<i>Task 1</i>	<i>T1b</i>	<i>UGS2</i> takes measurement again
<i>Task 2</i>	<i>T2b</i>	<i>UGS1</i> takes measurement
<i>Task 3</i>	<i>T3b</i>	<i>R1</i> goes to <i>UGS2</i>
<i>Task 4</i>	<i>T4b</i>	<i>R2</i> goes to <i>UGS1</i>
<i>Task 5</i>	<i>T5b</i>	<i>R1</i> goes to the door
<i>output</i>	$Y^2$	Intruder detected, Mission 2 completed

Table 4.4 Mission 2 Rule-base

<b>Mission2- operation sequence</b>	
<i>Rule1</i> $x_1^2$	If <i>input</i> then <i>T1b</i>
<i>Rule2</i> $x_2^2$	If <i>T1b</i> then <i>T2b</i>
<i>Rule3</i> $x_3^2$	If <i>T2b</i> then <i>T3b</i>
<i>Rule4</i> $x_4^2$	If <i>T3b</i> then <i>T4b</i>
<i>Rule5</i> $x_5^2$	If <i>T4b</i> then <i>T5b</i>
<i>Rule6</i> $x_6^2$	If <i>T5b</i> then $y^2$

If one sees Mission 2, Robot 1 is shared resource. Hence one needs to construct the  $F_{ud}$  matrix.

$$\begin{array}{c}
 \begin{array}{cccc}
 T1a & T2a & T3a & T4a
 \end{array} \\
 F_v^1 = \begin{array}{c} x_1^1 \\ x_2^1 \\ x_3^1 \\ x_4^1 \\ x_5^1 \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
 \end{array}
 \qquad
 \begin{array}{c}
 \begin{array}{cccc}
 R1 & R2 & UGS1 & UGS2
 \end{array} \\
 F_r^1 = \begin{array}{c} x_1^1 \\ x_2^1 \\ x_3^1 \\ x_4^1 \\ x_5^1 \end{array} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}
 \end{array}
 \end{array}$$

(a) (b)

Figure 4.9 Mission1 job sequencing matrix  $F_v^1$  (a), resource requirement matrix  $F_r^1$  (b)

The  $S_v^1$  matrix is built considering which tasks should be executed after a rule fires. The  $S_r^1$  matrix is built considering that  $S_r^1(i,j)$  is 1 if resource  $i$  has to be released after rule  $j$  has been fired. In the same way, the set of matrices relative to mission 2 can be built.

$$\begin{array}{c}
 \begin{array}{ccccc}
 x_1^1 & x_2^1 & x_3^1 & x_4^1 & x_5^1
 \end{array} \\
 S_v^1 = \begin{array}{c} T1a \\ T2a \\ T3a \\ T4a \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}
 \end{array}
 \qquad
 \begin{array}{c}
 \begin{array}{ccccc}
 x_1^1 & x_2^1 & x_3^1 & x_4^1 & x_5^1
 \end{array} \\
 S_r^1 = \begin{array}{c} R1 \\ R2 \\ UGS1 \\ UGS2 \end{array} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}
 \end{array}
 \end{array}$$

(a) (b)

Figure 4.10 Mission1 Task start matrix  $s_v^1$  (a) and resource release matrix  $s_r^1$  (b)

	T1b T2b T3b T4b T5b		R1 R2 UGS1 UGS2
$F_v^2 =$	$\begin{matrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \\ x_5^2 \\ x_6^2 \end{matrix} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$F_r^2 =$	$\begin{matrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \\ x_5^2 \\ x_6^2 \end{matrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
	(a)		(b)

$$F_{ud}^2(R1) = \begin{matrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \\ x_5^2 \\ x_6^2 \end{matrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

(c)

Figure 4.11 Mission2 Task sequencing matrix  $F_v^2$  (a), resource requirement matrix  $F_r^2$  (b) and conflict resolution matrix  $F_{ud}^2(R1)$  (c)

$$\begin{array}{c}
x_1^2 \quad x_2^2 \quad x_3^2 \quad x_4^2 \quad x_5^2 \quad x_6^2 \\
S_v^2 = \begin{matrix} T1b \\ T2b \\ T3b \\ T4b \\ T5b \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \\
\text{(a)}
\end{array}
\qquad
\begin{array}{c}
x_1^2 \quad x_2^2 \quad x_3^2 \quad x_4^2 \quad x_5^2 \quad x_6^2 \\
S_r^2 = \begin{matrix} R1 \\ R2 \\ UGS1 \\ UGS2 \end{matrix} \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\text{(b)}
\end{array}$$

Figure 4.12 Mission2 Task start matrix  $s_v^2$  (a) and resource release matrix  $s_r^2$  (b)

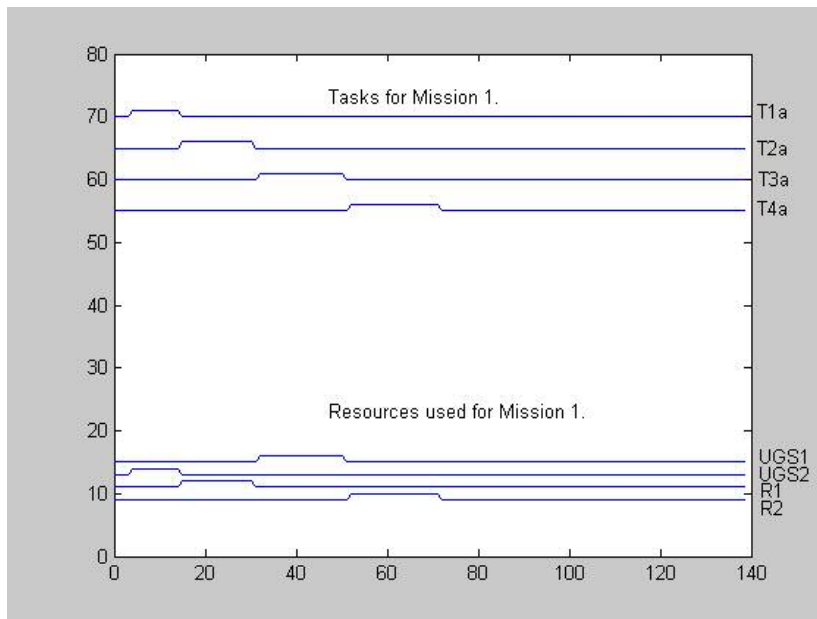
$$F_v = \begin{matrix} x^1 \\ x^2 \end{matrix} \begin{pmatrix} F_v^1 & 0 \\ 0 & F_v^2 \end{pmatrix} \quad F_r = \begin{matrix} x^1 \\ x^2 \end{matrix} \begin{pmatrix} F_r^1 \\ F_r^2 \end{pmatrix} \\
S_v = \begin{matrix} v^1 \\ v^2 \end{matrix} \begin{pmatrix} S_v^1 & 0 \\ 0 & S_v^2 \end{pmatrix} \quad S_r = r \begin{pmatrix} S_r^1 & S_r^2 \end{pmatrix}$$

Figure 4.13 Overall monitoring operation- Matrix formulation matrices  $F_v, F_r, S_v, S_r$

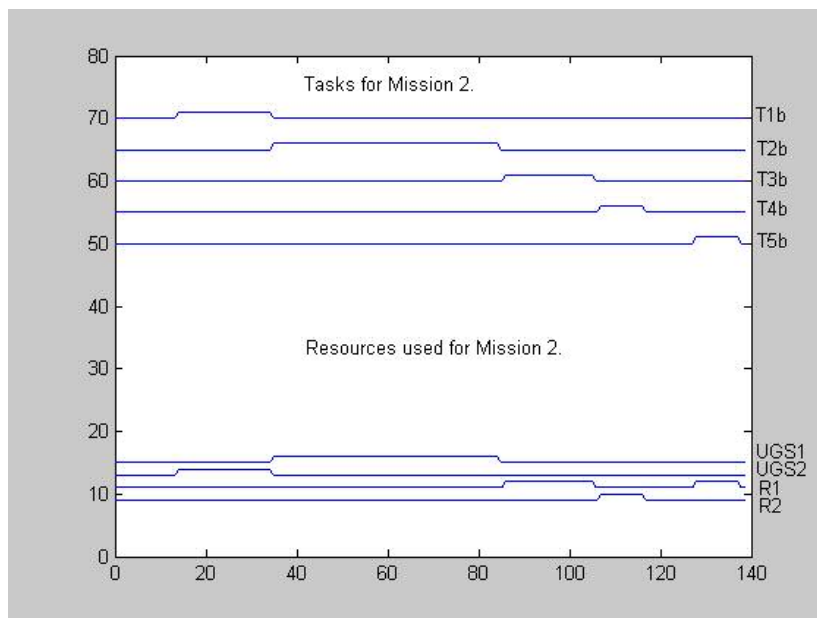
### 4.3 Simulation and experimental results

**Simulation:** Simulation of this system for the given missions using equations 4.3, 4.8 and 4.9 can be done using Matlab. Figure 4.14, shows the utilization time trace of the resources and the execution time trace of the tasks for mission 1 and mission 2.





(a)



(b)

Figure 4.14 Simulation results Mission 1 (a) Mission 2 (b)

**Experiment:** After performing extensive simulations, one can implement the control system directly on the WSN test-bed. Figure 4.15 shows the actual experimental utilization time trace of the agents, assigning higher priority to mission 1. Notice that the time duration of the real WSN runs in terms of discrete-event intervals, whereas the simulation results shown in figure 4.14 is in terms of time. It is interesting to note the similarity and fidelity of the dispatching sequences in both the simulation and experimental cases. This is a key result since it shows that the DEC allows one to perform a “simulate and experiment” approach for a WSN, with noticeable benefits in terms of cost, time and performance.

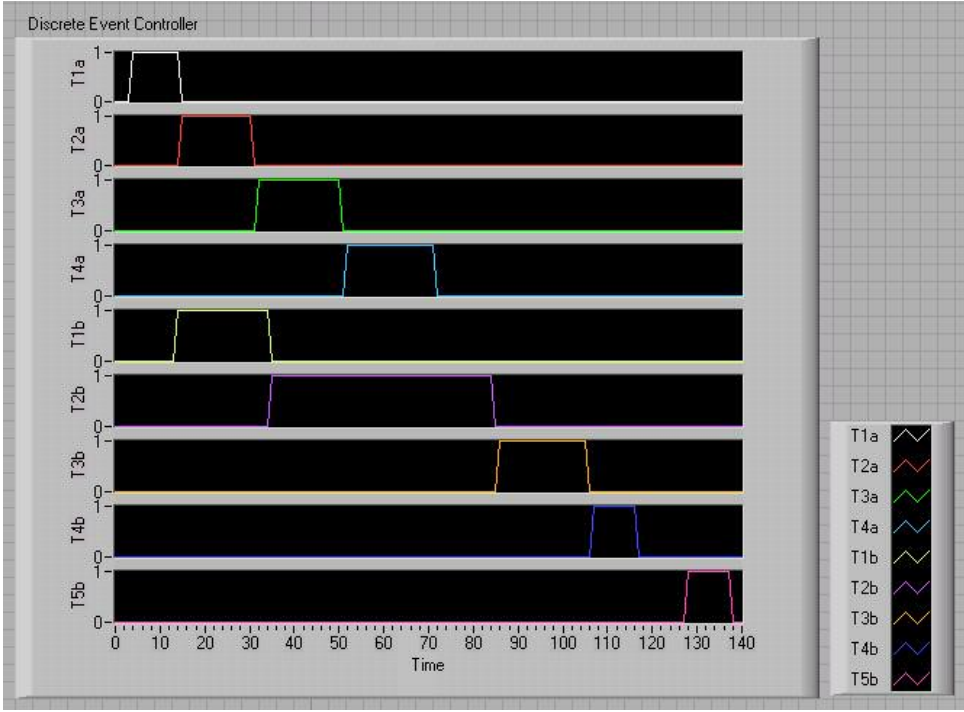


Figure 4.15 Utilization time trace of the WSN- Experimental results

#### 4.4 Decision for routing of resources

“Reentrant flow lines” are of great importance in manufacturing systems [34] where the resources needed for each job are pre-defined. Resource assignment and dispatching for such systems is well understood [35, 36]. But in the case of mobile WSN there are many resources such as distributed sensors and it is not known beforehand which sensor is most useful for resolution of certain events. Dynamic sensor selection is a special sort of *routing problem* [17], or free-choice Petri net, which requires highly complex decision-making. Therefore, one can use a new method for dealing with dynamic sensor selection using a novel Dynamic Priority Assignment Weighting Matrix.

Greedy activity/ resource selector algorithm [8] can be used for dynamic selection of resources most appropriate for a task in the DEC format. This can be done in the following method:

For each task that has a choice of resources to use, define a Dynamic Priority Assignment Matrix (DPAM) according to the example:

$$D_c = \begin{array}{l} \text{task 1} \\ \text{task 2} \\ \text{task 3} \end{array} \begin{array}{ccc} \text{res. 1} & \text{res. 2} & \text{res. 3} \\ \left[ \begin{array}{ccc} 0 & 1 & 0.6 \\ 1 & 0 & 0.4 \\ 0.1 & 0 & 1 \end{array} \right] \end{array}$$

which indicates that task 1 may be efficiently performed by resource 2, or less efficiently by resource 3. The numerical entry in position  $(i,j)$  is between 0 and 1, and indicates the efficiency with which resource  $j$  performs task  $i$ , with 0 indicating that resource  $j$  cannot perform task  $i$ , and 1 indicating that resource  $j$  performs task  $i$  with

maximum efficiency. Note that this matrix indicates that task 1 may be performed with *either* resource 2 *or* resource 3, in contrast to the matrix  $F_r$ , where multiple entries of 1 in a row indicate that *all* those resources are required for that task.

According to greedy dispatching policies [8], one selects the resource to perform a given task according to the immediate 1-step look ahead maximum payoff. Thus, depending on the DPAM, at each step, for the free-choice tasks modify the resource matrix  $F_r$  to have 1's in the entries corresponding to the maximum values of the DPAM in each row. This effectively selects the current most efficient resource to perform each task. Then, compute the DEC equations (3.1)-(3.4) to determine which tasks to start and which resources to reset.

The DPAM dynamically updates based on the evaluation information from the task on how well the assigned resource performed, after the tasks are over. Thus, resources that perform well will be assigned next time to that task. In this way, the DPAM ensures that an optimal resource is always assigned to a particular task to improve the overall efficiency of the system.

## CHAPTER 5

### SENSOR FUSION

#### 5.1 Introduction

Sensor fusion plays an important role in wireless sensor networks. Sensor fusion is an overarching term used to describe the process of collecting, distilling and displaying information from a number of homogeneous or heterogeneous information sources (sensors) to produce an integrated output that is more accurate and complete than that achievable using the same sensors independently of one another. In this way, the fused output is more than the sum of its parts. It is also known as data fusion. The ultimate goal of context-aware computing is to have computers understand the real world. It is an example of ‘Smart Environment’ where human-computer interactions feel natural, as if people are communicating with human personnel or assistants. This seems to be an impossible task as it presents a range of problems such as (1) how to represent this world with all its abstract concepts and unpredictable human feelings and (2) how to design and deploy sensors that can sense all the clues and content and map them into the context representation. Many methods are used for sensor fusion. In this thesis, Dempster Shafer and Fuzzy Logic methods [34] are used for sensor fusion to convert raw sensor data into usable events for the DEC to use. Also, in this thesis, a study of a simplified situation where these methods of sensor fusion are used is made. Here, we assume that context data can be represented by numbers and the mapping

from sensor output data to the context representation data structure is well defined.

General sensor fusion architecture is shown below.

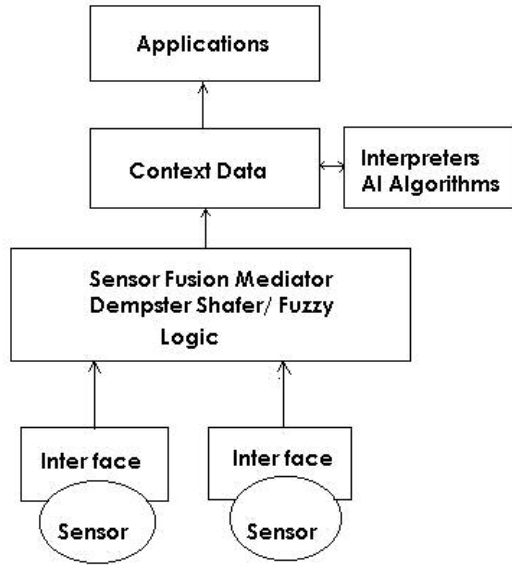


Figure 5.1 Sensor Fusion Architecture

This system generally has one central computer for context data repository for each major entity i.e. user to collect all the relevant context information about that entity. The sensor fusion mediator is responsible for collecting and monitoring the status of its corresponding sensors. The interaction between the sensors and the system is through an interface such as wireless or from RS-232 cable. The sensor fusion mediator converts the raw data into context aware events, which can be used by different applications and algorithms as shown in the figure. Further in this section, comparison of Fuzzy logic method and Dempster Shafer methods for sensor fusion is done.

## 5.2 Fuzzy Logic

Fuzzy Logic techniques have become very popular to address various processes for multi-sensor data fusion. As discussed earlier, important issues in building a fuzzy logic system for sensor fusion depends on the type of the membership functions used for the antecedents and consequents, the appropriate rule-base, and the method of defuzzification used. It also depends on the fuzzy logic operators used. Consider, if we have two sensors, each consisting of one light and one vibration sensor. We can get the raw sensor data and convert them into events such as if the light values are too low, there might be a lighting system malfunction, if light values are low and vibration values are high, there might be an intruder walking past the door where the sensors are fitted, if only vibration values are high, there might be an earthquake. This is done by defining triangular membership functions for input variables i.e. light and vibration. One can define ranges for these inputs such as ‘very high’, ‘medium’ and ‘very low’. These input ranges are mapped together to form fuzzy membership sets. Note that some of them overlap each other. These membership sets serve to ‘fuzzify’ the inputs so that one can deal with concepts that vary by degree. One also needs to define the number of outputs and their membership functions. In this thesis, implementation of the fuzzy decision making is done using both Matlab and LabView fuzzy logic toolkits. Matlab supports multiple input and multiple output systems whereas LabView only supports multiple inputs and single output fuzzy systems. An example of fuzzy membership functions in LabView is shown in figure 5.2.

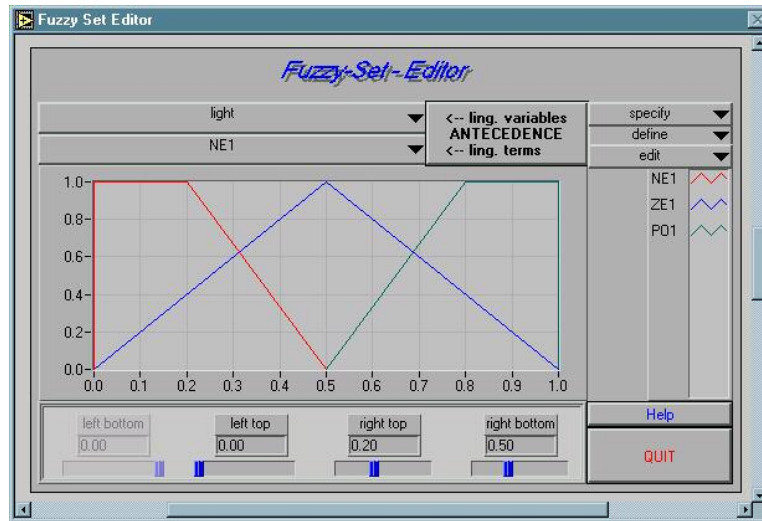


Figure 5.2 FL Membership function editor in LabView

See that the sensor values are normalized to be in the range of 0-1. Then a rule base is made to describe the scenarios of the system. An example of this can be “if light is very low and vibration values are very high then event of intrusion is very high”. An example of fuzzy toolkit rule base in LabView is shown in the figure 5.3.

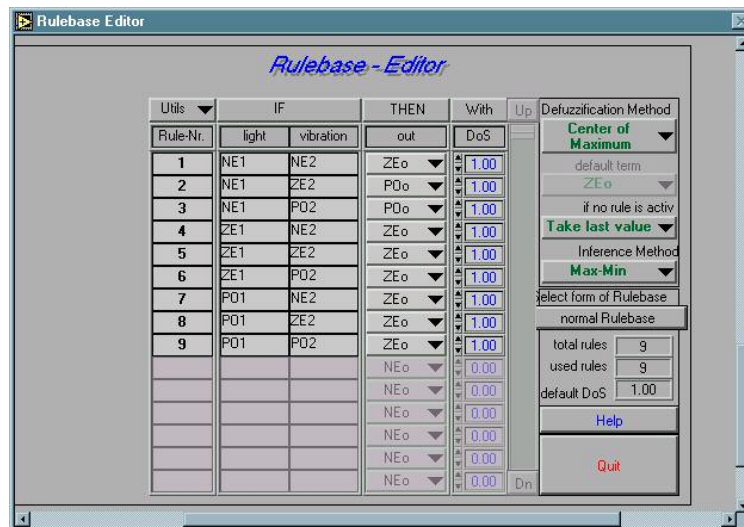


Figure 5.3 Rulebase editor in LabView



The fuzzy toolkits in both softwares provide different methods of defuzzification such as height, centroid, etc. In this thesis centroid defuzzification is chosen because of its inherent usefulness compared to other methods. One can also test this fuzzy system using both toolkits as shown in the figure 5.4.

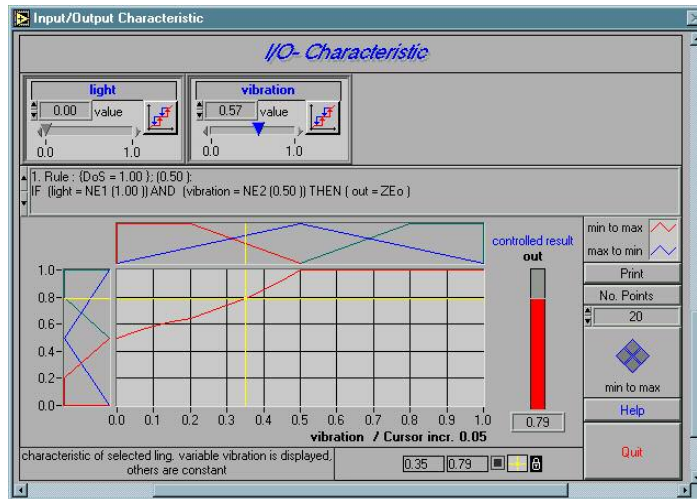


Figure 5.4 Testing FL system in LabView

This system provides good results only if the rule base is properly defined. In case of LabView one can use outputs from two fuzzy engines as inputs to another fuzzy engine for combination of event detection as shown in the figure 5.5.

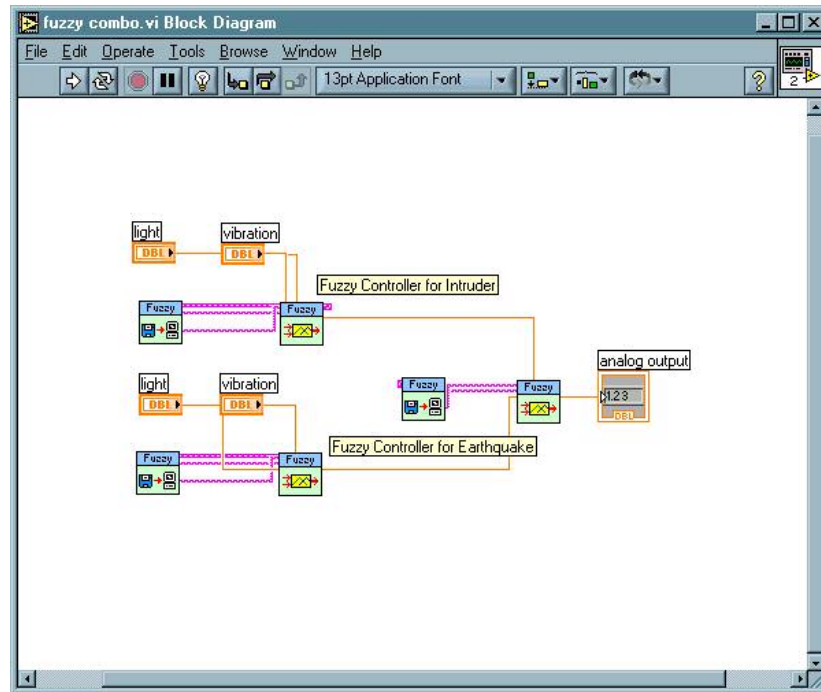


Figure 5.5 FL system block diagram in LabView

One can easily use fuzzy toolkit to test and implement sensor fusion.

### 5.3 Dempster Shafer

The Bayesian theory is the canonical method for statistical inference problems. The Dempster Shafer decision theory is considered a generalized Bayesian theory. It allows distributing support for preposition not only to a preposition itself but also to the union of prepositions that include it. In a Dempster Shafer reasoning system all possible mutually exclusive context facts or events of the same kind are enumerated in the frame of discernment  $\theta$ . Each sensor for example will contribute its observation by assigning its beliefs over  $\theta$ . This assignment is called as the basic probability assignment (BPA). In this thesis, Dempster Shafer rule of combination of evidence is used. Also, a rule-based method is used for updating the basic probability assignment dynamically [34].

The method is to collect evidence from different sensors and accordingly update the probability assignments. Suppose we have two sensors as discussed in the earlier section, with each of them having light and vibration sensors. If events are defined such as “Intrusion”, “Earthquake” or “Lighting malfunction”, we can use a rule base to update the basic probability assignments of each of these events. Thus each source i.e. sensor would use this rule base to update these probability assignments. An example would be:

Sensor 1:

If light values are very low and vibration values are very high then add 0.01 to the BPA of the event ‘Intrusion’.

If light values are very high then add 0.01 to the BPA of the event ‘Lighting malfunction.’

If vibration values are very high then add 0.01 to the BPA of the event ‘Earthquake’.

Similarly one can define the rule base for sensor 2 but in this case if it is assumed that the sensor 2 is not as reliable as sensor 1 then one can express an event such as ‘either intrusion or earthquake’.

Since Dempster Shafer rule is associative and commutative, one can combine the sources in any order. Suppose one has these two sources of evidence to take care of:

Table 5.1 Table for two sources of evidence

Sensor 1	Sensor 2
'Intrusion' with mass 0.5	'Intrusion or Earthquake' with mass 0.6
'Lighting Malfunction' with mass 0.3	'Lighting Malfunction' with mass 0.4
'Earthquake' with mass 0.2	'Earthquake' with mass 0.1

The numerator of Dempster Shafer rule computes a matrix of intersections and the belief masses contributed by each. For example in this case:

'Intrusion' with combined mass of: 0.3 i.e.  $0.5 \cdot 0.6$

'Lighting Malfunction' combined mass of: 0.12

'Earthquake' with combined mass of:  $0.02 + 0.12 = 0.14$

The masses for all the non-null events must be then divided by the denominator of Dempster Shafer's rule, equal to 1 minus the mass of all the null events, to yield a final body of evidence.

Null set mass =  $0.5 \cdot 0.4 + 0.5 \cdot 0.1 + 0.3 \cdot 0.6 + 0.3 \cdot 0.1 + 0.2 \cdot 0.4 = 0.54$

Denominator =  $1 - 0.54 = 0.46$

'Intrusion' with combined mass of:  $0.3 / 0.46 = 0.65$

'Lighting Malfunction' combined mass of:  $0.12 / 0.46 = 0.26$

'Earthquake' with combined mass of:  $0.02 / 0.46 = 0.04$

Here one can see that the possibility of the event 'Intrusion' is very high and hence we can assign a particular mission for that event in the discrete event controller. Credibility of the event X is equal to the sum of the belief masses of all the events that

are subsets of  $X$ . Plausibility of the event set  $X$  is equal to the sum of all the belief masses of all events whose intersection with  $X$  is non-null. In this example, credibility of the event ‘Earthquake’ is equal to the sum of the belief masses for ‘Earthquake’ and ‘Intrusion or Earthquake’.

#### 5.4 Analysis of Dempster Shafer and Fuzzy Logic

In case of the fuzzy logic controller, the block diagram in LabView for the above example is shown in the diagram 5.6.

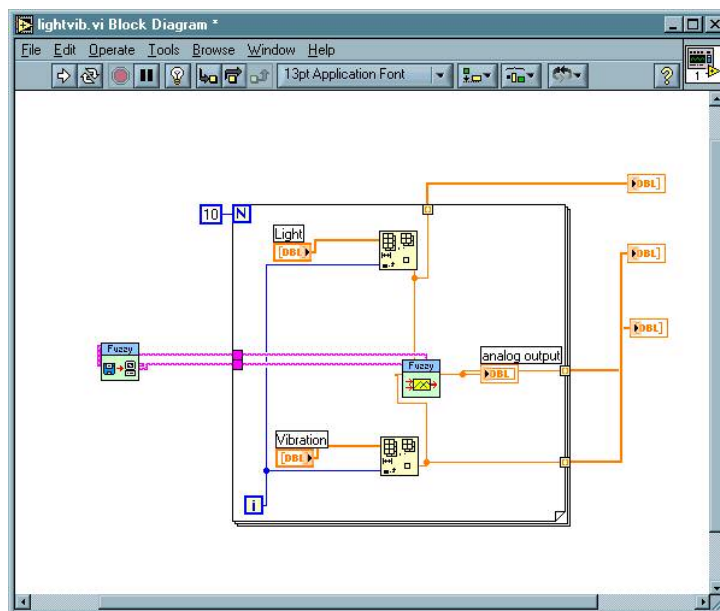


Figure 5.6 FL controller system block diagram in LabView

When simulated, one can see that as soon as the light values go down and vibration values go up, the event of intrusion shows an increased degree, predicting that the event of intrusion has occurred. This output from the fuzzy system triggers a mission in the Discrete Event Controller. The simulation waveform in LabView is shown in figure 5.7.

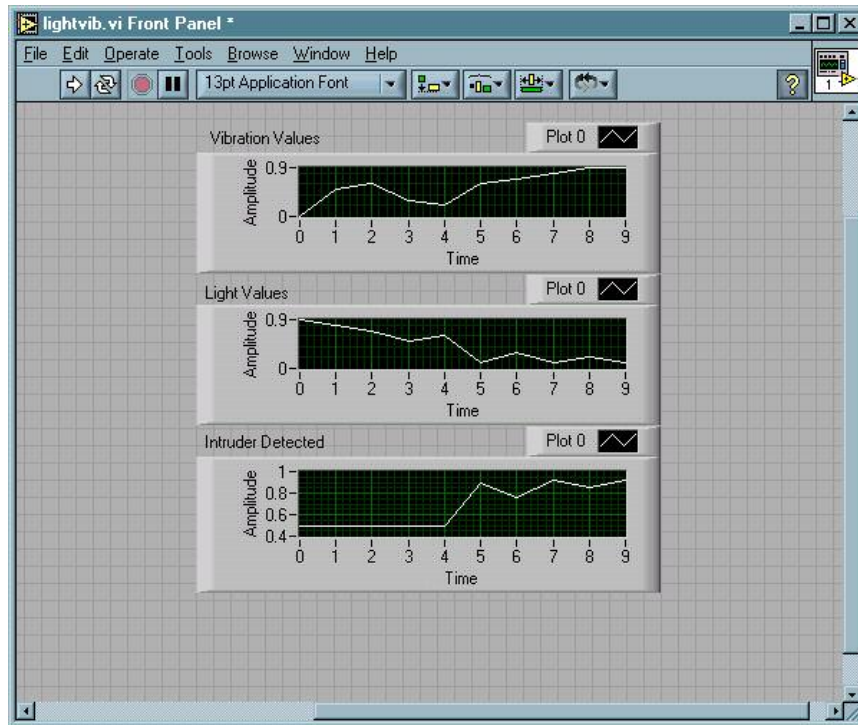


Figure 5.7 When Light values go low

In case of Dempster Shafer Theory, in the example discussed earlier, if the light values go low and vibration values go high, the combined BPA of 'Intruder' goes high as shown in the LabView simulation results in figure 5.8.

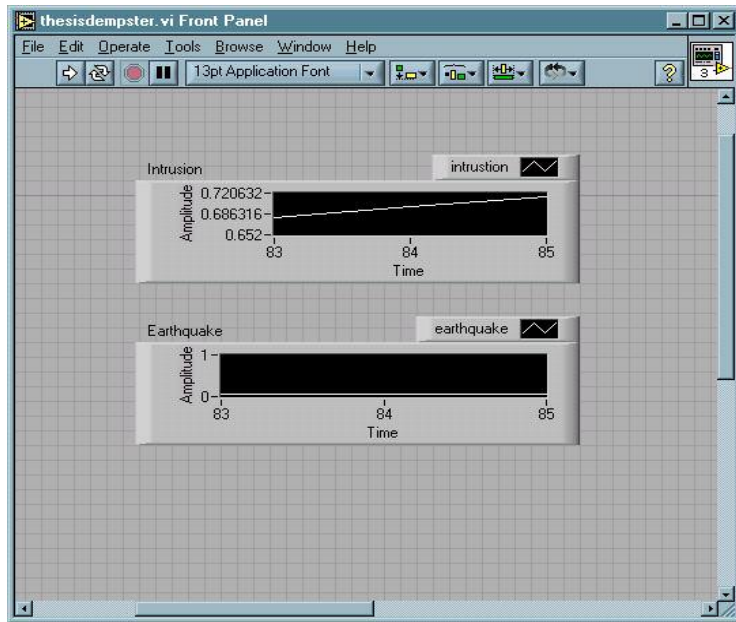


Figure 5.8 BPA of Intrusion increases

If the light values are high and the vibration values are high, the combined BPA of 'Earthquake' goes high as indicated in the LabView simulation results in figure 5.9.

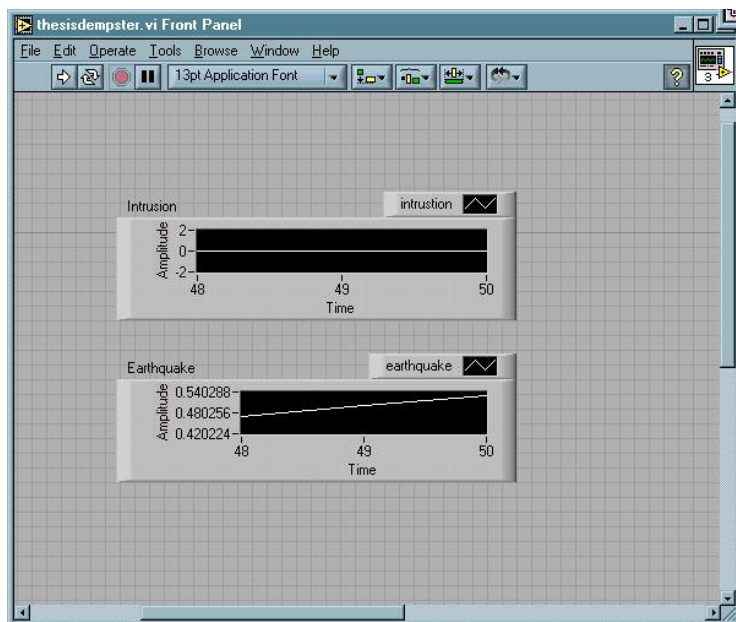


Figure 5.9 BPA of Earthquake increases

One can easily observe that both methods provide similar results. But there are problems in both methods. In case of Dempster Shafer theory, if the conflict is very high, the outputs can be sometimes contradictory. But there are methods to rectify it such as applying Yager's rule, etc. But in general, Dempster Shafer theory for sensor fusion has many advantages. Note that in Dempster Shafer Theory, the probability masses from propositions that contradict each other can also be used to obtain a measure of how much conflict there is in a system. This measure has been used before as a criterion for clustering multiple pieces of seemingly conflicting evidence around competing hypotheses. In addition, one of the advantages of the Dempster-Shafer framework is that priors and conditionals need not be specified, unlike Bayesian methods which often map unknown priors to random variables (i.e. assigning 0.5 to binary values).

In case of Fuzzy Logic method of sensor fusion, implementation seems simple in case of a basic example. But as the number of inputs and outputs increase, the number of rules in a system grows exponentially. For real time systems, this type of brute force is out of the question. But there are methods to improve this system. One can restrict the decision system to a single non fuzzy behavior. If one can cut down on the number of behaviors that is needed to be considered for the current decision cycle, one can avoid all of the rule evaluations associated with each behavior. One can also break down the behaviors into parallel layers that can fire and operate independent of each other to prune down unnecessary rule evaluations. This system can also be made



adaptive so that the behaviors can change the rules or membership functions. But for a basic system, both methods provide excellent results.

## CHAPTER 6

### CONCLUSION

#### 6.1 Conclusion

In this thesis a discrete-event coordination scheme for sensor networks composed of both mobile and stationary nodes was introduced. This architecture supports high-level planning for multiple heterogeneous agents with multiple concurrent goals in dynamic environment. The proposed formulation of the DEC represents a complete dynamical description that allows efficient computer simulation of the WSN prior to implementing a given DE coordination scheme on the actual system. The similarity between simulation and experimental results shows the effectiveness of the DEC for simulation analysis. The obtained results also prove the striking potentialities of the matrix formulation of the DEC, namely: straightforward implementation of missions on the ground of intuitive linguistic descriptions; possibility to tackle adaptability and scalability issues at a centralized level using simple matrix operations; guaranteed performances, since the DEC is a mathematical framework which constraints the behaviour of the single agents in a predictable way. DEC was actually implemented on an actual mobile WSN test-bed to prove the simplicity and effectiveness of the employed method.

A study and implementation of fuzzy logic and Dempster Shafer theory for sensor fusion was also done. The implementation results justify the theoretical premise.

Also, the use of fuzzy logic and Dempster Shafer theory for sensor fusion is very effective.

## 6.2 Future Scope

There are some issues that still need to be answered in DEC architecture such as decision-making for the dynamic change of input matrices and the inherent problem of deadlock. Also there is the problem of decision for routing of tasks and resources when multiple inputs come in and cause a deadlock. Rigorous work has been done on systems having pre-deterministic routing paths. However, very little work has been done for systems having free-choice routing paths such as wireless sensor networks. Serious problems would appear in the performance of these systems if the assignments of resources for specific tasks are not correctly sequenced. Some of the problems are of blocking and system deadlock. In order to analyze the internal deadlock structures for these systems, we need to investigate the routing paths which contain circular wait relationships among resources, while increasing the possibilities for blocking and deadlock. Thus, the future work would involve matrix analysis of mobile WSN systems with routing, deadlock avoidance and adaptive resource failure management.

APPENDIX A  
IMPORTANT LABVIEW BLOCKS

Table A Important LabView Blocks



LabView Block	Description
<div style="text-align: center;">  <p>VISA read. Vi</p> </div>	<p>Reads the specified number of bytes from the device or interface specified by VISA resource name and returns the data in read buffer. Whether the data is read synchronously or asynchronously is platform-dependent. Right-click the node and select Do I/O Synchronously from the shortcut menu to read data synchronously. The operation returns only when the transfer terminates.</p>
<div style="text-align: center;">  <p>VISA Write. vi</p> </div>	<p>Writes the data from write buffer to the device or interface specified by VISA resource name. Whether the data is transferred synchronously or asynchronously is platform-dependent.</p>

Table A - *continued*






 <p>Build Array. vi</p>	<p>Concatenates multiple arrays or appends elements to an n-dimensional array. You also can use the Replace Array Subset function to modify an existing array.</p> <p>The connector pane displays the default data types for this polymorphic function.</p>
 <p>Test Fuzzy Control. Vi</p>	<p>This VI is used to build and test a fuzzy control application. This can be used as a general purpose VI with up to 4 inputs and 1 output.</p>
 <p>Fuzzy Controller. vi</p>	<p>This is a fuzzy controller VI which is run from Fuzzy toolkit. A controller data file is needed for this VI to work.</p>

Table A - *continued*

<div style="text-align: center;">  <p>Load Fuzzy Controller. vi</p> </div>	<p>Load Fuzzy Controller VI is used to load all the controller parameters and components. This VI is used with Fuzzy controller VI to implement fuzzy controller designed with fuzzy logic controller design project manager.</p>
<div style="text-align: center;">  <p>While loop</p> </div>	<p>Repeats the subdiagram inside it until the conditional terminal, an input terminal, receives a particular Boolean value. The Boolean value depends on the continuation behavior of the While Loop. Right-click the conditional terminal and select Stop if True or Continue if True from the shortcut menu. You also can wire an error cluster to the conditional terminal, right-click the terminal, and select Stop on Error or Continue while Error from the shortcut menu. The While Loop always executes at least once. The iteration (i) terminal provides the current loop iteration count, which is zero for the first iteration.</p>

## REFERENCES

- [1] Akyldiz I., Su W., Sankarasubramaniam Y, Cayirci E., “A survey on sensor networks”, IEEE Communications magazine, August 2002
- [2] Balch T., Hybinette M., “Social potentials for scalable multi-robot formations”, Proceedings of the International Conference of Robotics and Automation, April 2000
- [3] Balch T., Arkin R., “Behavior-based formation control for multirobot teams”, IEEE Transactions on Robotics and Automation, vol. 14, no.6, December 1998
- [4] Burgard, W.; Moors, M.; Fox, D.; Simmons, R.; Thrun, S.; “Collaborative multi-robot exploration”, Proceedings of the IEEE International Conference on Robotics and Automation, 2000, vol. 1 , 24-28 April 2000 Pages:476 – 481
- [5] Bronson, R., and Naadimuthu, G., *Operations Research*, 2 ed., Schaum’s Outline, McGraw-Hill, New York, 1997.
- [6] Butler Z., Rus D., “Event-based motion control for mobile-sensor network”, IEEE Transactions. on Pervasive Computing, vol.2 issue 4, October-December 2003
- [7] Christensen T., Noergaard M., Madsen C., Hoover A., “Sensor networked mobile robotics”, Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, June 2000
- [8] Corman, T., Leisenson, C., and Rivest R., *Introduction to Algorithms*, Prentice Hall of India, 2001.



- [9] Cortes J., Martinez S., Karatas T., Bullo F., "Coverage control for mobile sensing network", *IEEE Trans. on Robotics and Automation*, vol.20, no.2, April 2004
- [10] Gordon-Spears A., Kiriakidis K., "Reconfigurable robot teams: modeling and supervisory control", *IEEE Transactions on control system technology*, vol. 12, no. 5, September 2004
- [11] Gerkey B., Mataric M., "A formal analysis and taxonomy of task allocation in multi-robot systems", *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939-954, Sept. 2004.
- [12] Harris B., Lewis F., Cook D., "Machine planning for manufacturing: dynamic resource allocation and on-line supervisory control", *Journal of Intelligent Manufacturing*, pp. 413-430, vol. 9, 1998
- [13] Harris, B., Cook, D., and Lewis, F.L., "Automatically generating plans for manufacturing," *J. Intelligent Systems*, vol. 10, no. 3, pp. 279-319, 2000.
- [14] Howard, A.; Mataric, M.J.; Sukhatme, G.S.; "An incremental deployment algorithm for mobile robot teams", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, vol.30, Pages:2849 – 2854, October 2002
- [15] J. Ezpeleta, J.M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robotics and Automation*, vol. 11, no. 2, pp. 173-184, 1995.

- [16] King J., Pretty R., Gosine R., “Coordinated execution of tasks in multiagent environment”, IEEE transactions on Systems, man and cybernetics- Part A: Systems and Humans, vol. 33, no.5, September 2003
- [17] Kusiak A. “Intelligent scheduling of automated machining systems.” In Intelligent design and Manufacturing. A. Kusiak (ed.) Wiley, New York (1992)
- [18] Lee J., Hashimoto H, “Controlling mobile robots in distributed intelligent sensor network”, IEEE Transactions on Industrial Electronics, vol.50, no.5, October 2003
- [19] Lewis F., “Wireless sensor networks”, Smart environments: technologies, protocols, and applications, ed. D. J. Cook and S. K. Das, John Wiley, New York, 2004.
- [20] Lewis F., A. Gurel, S. Bogdan, A. Doganalp, O. Pastravanu, “Analysis of deadlock and circular waits using a matrix model for flexible manufacturing systems,” Automatica, vol. 34, no. 9, pp. 1083-1100, 1998.
- [21] McMickell M., Goodwine B., Montestruque L., “MICAbot: a robotic platform for large-scale distributed robotics”, Proceedings of the International conference of robotics and automation, September 2003
- [22] Mireles J., Lewis F., “Intelligent material handling: development and implementation of a matrix-based discrete event controller IEEE Transactions on Industrial Electronics, , vol. 48 , Issue: 6 , Dec. 2001 Pages:1087 – 1097
- [23] Mireles J., Lewis F., “Deadlock analysis and routing on free-choice multipart reentrant flow lines using a matrix-based discrete event controller” Proceedings of the IEEE International conference on Decision and Control, December 2002

- [24] Murata, T. "Petri nets: properties, analysis and applications." Proceedings of the IEEE, vol.77, no.4, April 1989, pp.541-80
- [25] Parker L., "ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation", IEEE Transactions on Robotics and Automation, vol. 14, no. 2, April 1998
- [26] Petriu E., Whalen T.,Abielmona R., Stewart A., "Robotic sensor agents: a new generation of intelligent agents for complex environment monitoring", IEEE Magazine on Instrumentation and Measurement, vol.7 issue 3, September 2004
- [27] Paruchuri, P., Tambe, M., Ordonez, F., Kraus, S., "Towards a formalization of teamwork with resource constraints," Proc. Third Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 596-603. AAMAS 2004.
- [28] P.J. Gmytrasiewicz, H.-H. Huang, and F.L. Lewis, "Combining operations research and agent-oriented techniques for agile manufacturing system design," Proc. IASTED Int. Conf. Robotics and Manufacturing, pp. 396-402, Cancun, Mexico, June 1995.
- [29] P.R. Kumar and S.P. Meyn, "Stability of queueing networks and scheduling policies," *IEEE Trans. Automat. Control*, vol. 40, no. 2, pp. 251-260, 1995.
- [30] Saridis G., "Intelligent robotic control", IEEE Transactions on Robotics & Automation, vol. 28, no.5, May 1983
- [31] Sibley G., Rahimi M., Sukhatme G., "Robomote: a tiny mobile platform for large-scale ad-hoc sensor networks", Proceedings of the International conference of robotics and automation, May 2002

- [32] S.S. Panwalker and W. Iskander, "A survey of scheduling rules," *Operations Research*, vol. 26, no. 1, pp. 45-61, 1977.
- [33] Steward D. V., "The design structure system: a method for managing the design of complex systems", *IEEE Transactions on Engineering Management*, pp. 45-54, Aug. 1981
- [34] S. Rabin, *AI Game Programming Wisdom*, Charles River Media, Inc, 2002.
- [35] Tacconi D., Lewis F., "A new matrix model for discrete event systems: application to simulation", *IEEE Control System Magazine*
- [36] Tilak S., Abu-Ghazaleh N., Heinzelman W., "A taxonomy of wireless micro-sensor network models," *ACM Mobile Computing and Communications Review*, Vol. 6, No. 2, pp. 28-36, 2002
- [37] Wu H., Siegel M., Stiefelhagen R., Yang J., "Sensor fusion using Dempster-Shafer theory", *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, May 2002

## BIOGRAPHICAL INFORMATION

Prasanna Ballal received his Bachelor of Engineering degree in Electronics and Telecommunication Engineering from Mumbai University in 2002. He then worked for IndiaGames Ltd, India as a software programmer. He started his masters in Electrical Engineering at The University of Texas at Arlington in 2003. Due to his interests in Signal Processing and Wireless systems, he started working on research projects involving Mobile Wireless Sensor Networks in Automation and Robotics Research Institute (ARRI-UTA) under Dr. Frank Lewis. He has been working as a Graduate Research Assistant at Automation and Robotics Research Institute. He has also been a Graduate Teaching Assistant for both undergraduate and graduate courses in UTA.