

To appear in IEEE-SMC (Sept-Oct-Nov
1991 issue)

AD-A242 553



①

INFORMATION INTEGRATION AND SYNCHRONIZATION IN DISTRIBUTED SENSOR NETWORKS

D.N.Jayasimha*, S.S.Iyengar** and R.L.Kashyap***

DTIC
ELECTE
NOV 18 1991
S D

* Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210-1277

** Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020

*** Department of Electrical Engineering
Purdue University
West Lafayette, IN 47907

This document has been approved
for public release and sale; its
distribution is unlimited.

91-15182



* This author's work was supported in part by the National Research Foundation under Grant No. NSF-CCR-8909189.

** This author's work was supported in part by the Office of Naval Research under Grant No. ONR-N00014-91-J-1306 and in part by the LEQFS- Board of Regents under Grant No. LEQFS-RD-A-04.

*** This author's work was supported in part by the Innovative Science and Technology (IST) program of the SDIO, monitored by the Office of Naval Research under Grant No. ONR-N00014-85K-0611, and by the Engineering Research Center for Intelligent Manufacturing Systems under Grant No. CDR-8803017.

91 11 6 027

ABSTRACT

In recent years, the study of systems with multiple sensors has been an active area of research. In this paper, we focus on the computational, i.e., architectural, algorithmic, and synchronization issues related to competitive information integration in a Distributed Sensor Network (DSN). The proposed architecture of the DSN consists of a set of binary trees whose roots are fully connected. Each node of the tree has a processing element and one or more sensors associated with it. The information from each of the sensors has to be integrated in such a manner that the communication costs are low and that the real time needs are met. We present an information integration algorithm which has a low message cost (linear in the number of nodes of the network) and a low distributed computation cost.

In a distributed environment there is no central clock which regulates the activities of each node. Further, the clock at each node is typically not accurate. The estimates from each of the sensors which need to be integrated have to be temporally "close to each other", however. We consider the problems associated with synchronizing information to be integrated in the presence of imperfect clocks. We derive a) the relationships between the clocks of the processing elements in the network for proper information integration and b) an upper bound on the period between consecutive resynchronizations of a processing element's clock with the central time server. Finally, we discuss the fault tolerant features of the network and the integration algorithm.

KEYWORDS and PHRASES : Distributed Sensor Network, Integration Algorithm, Fault Tolerance, Clock Synchronization, Abstract Sensor Estimate.

RECEIVED	
JAN 10 1994	
DISTRIBUTION	
By <i>per. lth.</i>	
Distribution	
A-1	

1.0 INTRODUCTION

In recent years the study of multisensor systems has been an active area of research [14, 4, 2, 11, 10]. Our interest is in multiple sensor systems which consist of tens or even hundreds of sensors. Such a system is usually organized as a distributed sensor network (DSN) which consists of a set of sensors, a set of processing elements (PEs), and a communication network interconnecting the various PEs. One or more sensors is associated with each PE. A need for DSNs arises in diverse applications such as intelligent robotic systems, aircraft navigation, systems which monitor the activities on an industrial assembly line, etc.[4, 10, 15]. A significant advantage offered by the integration of information from disparate sensors is that the reliability and the fault tolerance of the sensor system are both enhanced. Another advantage is the suppression of the effects of noise. This is because the noise measured by different sensors tends to be uncorrelated, while the signal of interest remains correlated. Also, by careful selection of disparate sensors, one can compensate for the shortcomings and peculiarities of particular types of sensors.

The method of integrating information from the various sensor outputs, called information integration, depends on whether the sensors provide a) competitive information or b) complementary information [4, 2]. In the former case, each sensor ideally provides identical information. In reality, however, the information provided by each sensor may vary widely because of a variety of reasons such as the noise at a particular sensor site, unreliable transmission due to channel failures, loss of accuracy in translating information from disparate sources into a common information medium. It is therefore necessary for the information from the sensors to be combined in a meaningful and efficient manner to obtain a fairly accurate result. Complementary information integration is done when only partial information is available from each sensor; such information is then integrated to build the complete information. Note that in either case, we need an abstract model of the sensor to compare or combine the information from disparate sensor sources.

1.1 SCOPE OF THIS PAPER

From the foregoing discussion, it is clear that the integration of multiple disparate sensors into an effective sensor network requires the solution of several problems. These problems could be broadly categorized into two categories:

- (a) Those related to sensor modeling and
- (b) Those related to sensor information integration.

Examples in the first category include the problems associated with the specification of appropriate sensor parameters [4] or the probabilistic model of individual sensor performance [1]. Examples in the second category include methods needed to abstractly represent the information gained from sensors so that this information may easily be integrated and also methods to deal with possible differences in points of view on frames of references among multiple sensors [1, 8]. For an excellent discussion of the problems and current state of the art in multisensor integration, the reader is referred to the survey paper by Luo and Kay [10].

In this paper we focus on the architectural, algorithmic, and the clock synchronization issues related to competitive information integration needed in the distributed sensor network. Our paper discusses the situation when the sensor outputs are connected intervals on the real line $\mathbb{R}$. Our interval estimates are tolerance zones containing sensors associated with the PEs of the distributed sensor network and our technique of integration focuses on obtaining reliable regions for correct sensor values with fault tolerance. These issues have not received much attention in the context of multisensor systems. The architectural issues are discussed to a certain extent by [5, 15].

1.2 ORGANIZATION OF THE PAPER

The paper is organized as follows: In section 2 we give the architecture of the DSN under consideration. Our abstract sensor model, detailed in section 3, is an extension of the model developed by Marzullo[12]. In section 4, we motivate the need for clock synchronization and present the information integration algorithm. In section 5, we present certain fault-tolerant features of the network and the integration algorithm. Section 6 contains some concluding remarks and the future directions this research could take. The Appendix presents a computational characterization of the integration algorithm using Heaviside functions.

2.0 ARCHITECTURE OF THE DSN

Two types of architectures for distributed networks have been discussed by Wesson, et. al [13], namely the committee (or anarchic) organization and the hierarchical organization. In a committee organization, each node (i.e., the PE and its associated sensors) in the network is autonomous and can broadcast information to all other nodes in the network. In a hierarchical organization, the nodes are assembled as strict hierarchies of abstraction levels. At each level, nodes receive information from the lower level nodes, integrate the information received according to their position in the hierarchy, and send abstracted reports to nodes at higher levels in the hierarchy. The node at the highest level, called the commander, makes appropriate decisions based on the received information, globally interprets the required hypotheses, and controls the network appropriately.

The committee organization, which requires $\Theta(N^2)$ interconnections in an N node DSN, would be expensive and practically infeasible when one considers a large number of nodes. The hierarchical organization, on the other hand, does not suffer from this disadvantage (it requires $\Theta(N)$ interconnections in an N node DSN), but could produce inaccurate estimates since data sharing is not allowed between the low level sensors and since errors accumulate as one goes up the hierarchy.

In this paper we consider a DSN consisting of several clusters. Each cluster is organized in a hierarchical manner as a binary tree of nodes. Each node has a PE and an associated sensor. The leaf nodes are at the first level of the network. The parent of a node is at one level higher than its children. A clock runs on each PE. There is a message buffer associated with each channel as shown in Figure 2.3. After the information is read by a sensor, the associated PE translates that information into an abstract sensor estimate (more detailed description on abstract sensor estimate is given in the next section), time stamps the estimate with the current time and places the abstract estimate in the buffer (see Figure 2.3). The root of each cluster, called the commander, forms a complete interconnection with all its peer nodes. Such an architecture illustrated in Figures 2.1, 2.2 and 2.3 could be thought of as a hierarchical system with a committee organization at the top level. Each PE communicates to its children or its parent over channels through messages.

In addition to the issues already mentioned, we are also interested in the fault tolerance of the system, i.e., the individual sensors could be faulty but the ensemble of sensors should possess reasonable fault tolerant properties. We address the issue of fault tolerance briefly in Section 5. Fault tolerance, however, is not the primary focus of the paper.

Some of the important issues which need to be examined are:

- (i) How should the observations available at a sensor or a sensor cluster be processed so as to recover the relevant signal parameters ?
- (ii) What are the message routing strategies for the network?
- (iii) How does internodal coupling affect obtaining of unbiased estimates of a parameter?
- (iv) How do we structure the signal processing cluster units?

The first issue is discussed in detail in the next two sections. The second issue, message routing involves considering the time delays in message passing in distributed sensor networks. In Section 4, we present a clock synchronization scheme which integrates information considering time delays. The other issues of importance related to message passing, which, however, are not examined in detail in this paper are: a) fail-safe message passing strategies and b) message routing strategies when there exist multiple paths between two nodes in the DSN. To illustrate a problem relating to the third issue, consider Figure 2.4. The estimate that P receives from L could itself be a function of P's estimate since, in addition to the path from L to P, there exists a path from P to L. As a result, the combined estimate calculated at P could be biased estimate favoring P's estimate since the latter has been included twice in the calculation for the combined estimate.

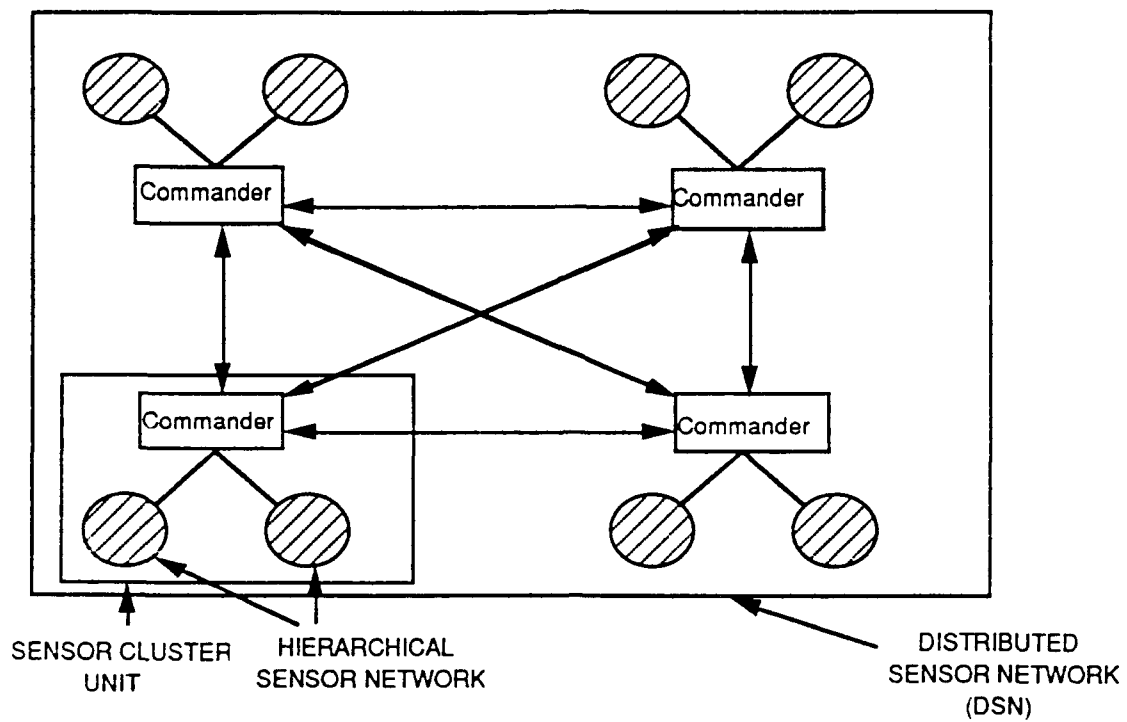


Figure 2.1 Overview of DSN Architecture

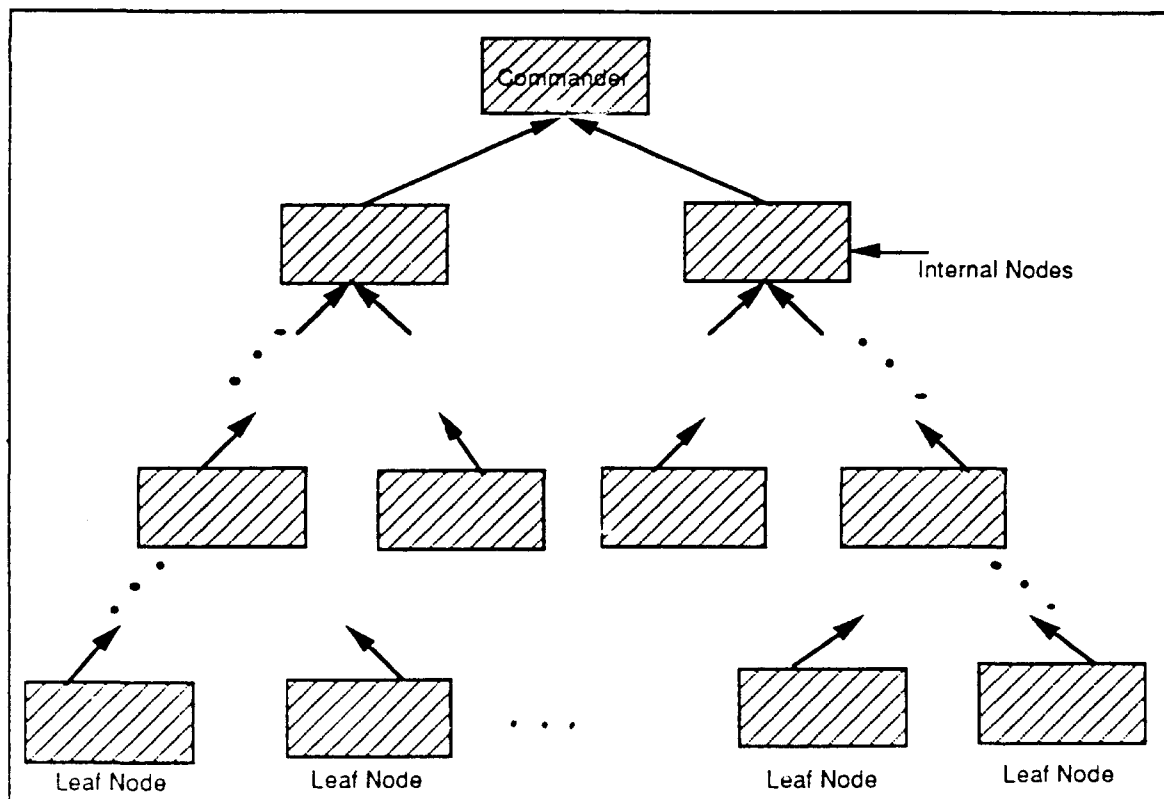


Figure 2.2 Sensor Cluster Unit

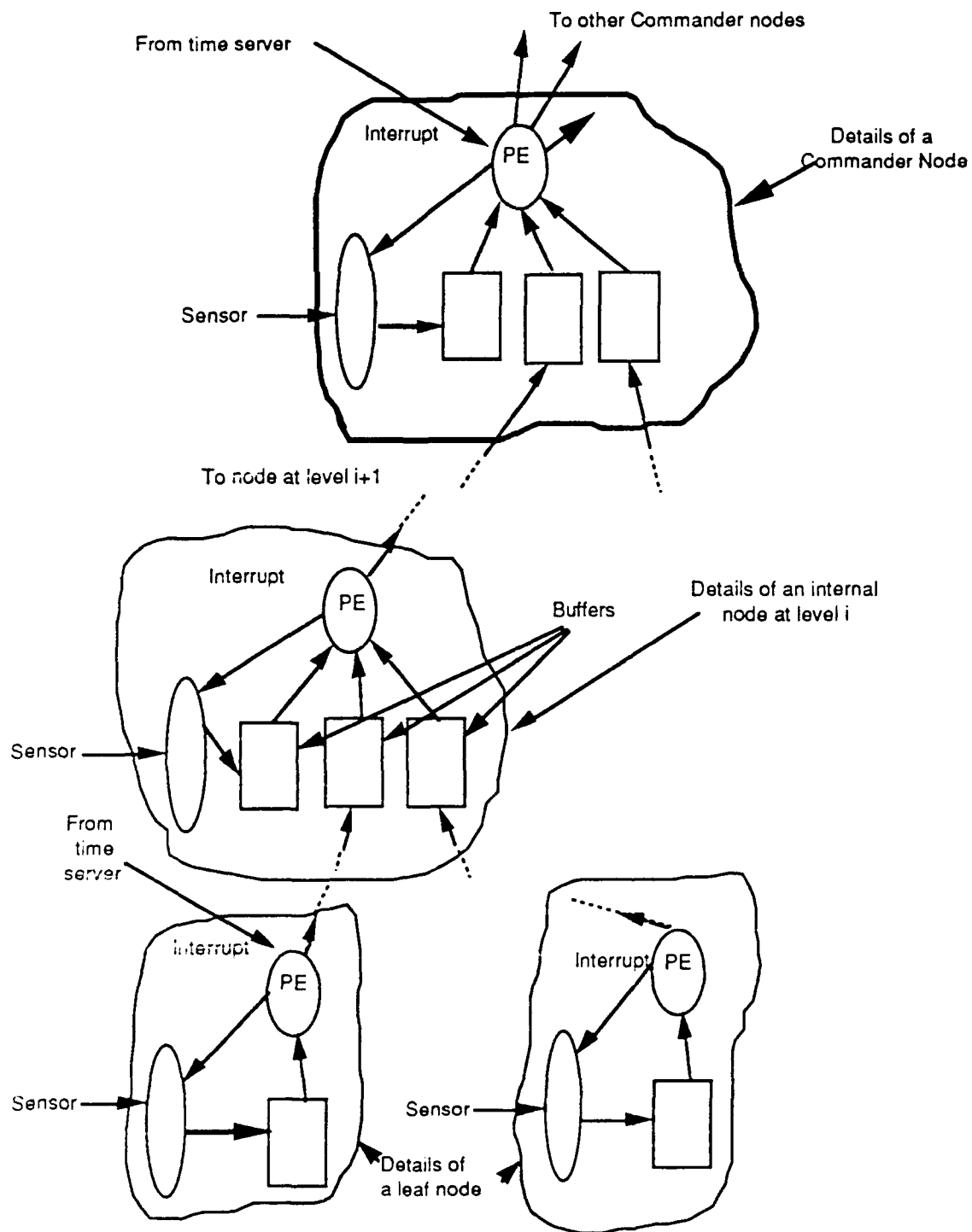


Figure 2.3 Details of the sensor cluster unit

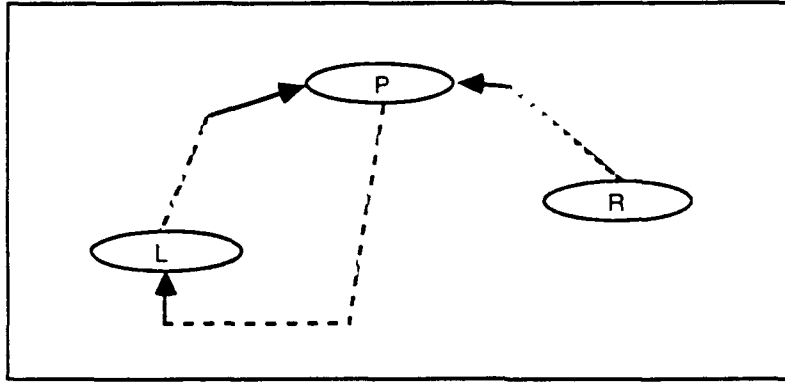


Figure 2.4 L's integrated sensor value that P obtains could itself be a function of P's value since there exists a path from P to L.

We focus mainly on the first two issues in this paper. The other issues do not concern us since we restrict our discussion to a distributed sensor network with a complete binary tree of nodes at each cluster unit and with unidirectional information flow from nodes at the lower level to the nodes at the higher level.

3.0 THE ABSTRACT SENSOR MODEL

A physical sensor is a device which samples a physical variable. For example, a PE controlling a surveillance satellite antenna might have an infra-red sensor as a physical sensor. The PE may obtain measurements either by polling it for its current value or by being asynchronously alerted when a certain signature is detected. Assume for simplicity that each sensor outputs a single value represented by a real number. In reality, however, because of reasons such as a) limited accuracy of the sensor, b) effect of noise signals, c) various delays (the communication delay, for example) between the time t_s that the value is read and the time t_p that the value is processed by the PE, and d) inaccuracy in knowing the correct values of t_s and t_p in distributed systems, it is not meaningful for the sensor value to be represented by a single real value. (Note that to consider the effect of c) mentioned above, the extrapolation characteristics of the process have to be known). Hence we define an abstract sensor to be a piecewise continuous function which maps a physical sensor value v ($v \in \mathbb{R}$) into a dense interval $[a, b]$ ($a, b \in \mathbb{R}$) that contains the physical value. The width of an interval is $(b-a)$. Ideally, this interval $[a, b]$ converges to v but, in reality, the width is finite. This abstract sensor model is based on the work of Marzullo [12]. He also shows that given a physical sensor, it may not be easy to implement

the corresponding abstract sensor, as it requires knowledge of the characteristics of the physical process being monitored. We assume that this knowledge is available. In many cases when the characteristics are only approximately known, a high sampling rate will compensate for this lack of knowledge. We use this model to construct a fault-tolerant abstract sensor.

The abstract sensors representing two or more sensors could be combined to form an abstract estimate. To keep the terminology simple, we refer to the abstract sensor as the abstract estimate also. An abstract estimate could, in turn, be combined with one or more abstract estimates to form a new abstract estimate. A k -interval abstract estimate is defined to be an empty set or a set of up to k intervals $\{(l_1, u_1), \dots, (l_i, u_i)\}$ (where $1 \leq i \leq k$) such that $l_j < u_j$ (where $1 \leq j \leq i$) and $u_j < l_{j+1}$ (where $1 \leq j \leq i-1$). Observe that these intervals do not overlap. If any two did overlap then they would be in one interval. The width of a k -interval abstract estimate is defined to be the value:

$$\sum_{i=1}^k (u_i - l_i)$$

The next section addresses the question of how the abstract sensors or abstract estimates are combined to yield new abstract estimates.

4.0 INFORMATION INTEGRATION OF ABSTRACT ESTIMATES

In this section, we present a method to combine information from multisensor systems which we call information integration. Before we present the actual algorithm, we motivate the need to combine the information from different sensors and also show that the meaningful combining of information necessitates proper clock synchronization which is non-trivial to achieve in a distributed environment such as ours.

4.1 MOTIVATION

Structuring sensor information is a prerequisite for the integration problem in the cluster-tree network. Suppose a node A at an intermediate stage of the network receives abstract estimates (l_1, u_1) and (l_2, u_2) from its two children. Assume that $l_1 < l_2$ and that the two intervals do not overlap. Node A can communicate the 2-estimate $\langle (l_1, u_1), (l_2, u_2) \rangle$ to its parent to signify that the true value lies in one of the two intervals (which of the two

intervals contains the true value would be resolved at a higher level) or it could transmit a 1-estimate (l_1, u_2) . Each interval requires the communication of two values. Observe that in the former case, at stage i of the network, $O(2^i)$ number of values need to be communicated by a node. This has a number of disadvantages: a) high communication cost and b) high computation cost at the root node (commander node in our case). Since these estimates are usually a function of time, real time needs may preclude the computation of estimates which have an exponential growth rate at the root node. On the other hand, in the latter case, the communication requirements are lesser (later we will characterize this measure more precisely) and the computation requirements are distributed among all the nodes. The accuracy of the result could suffer, however. In the example considered, we transmit a 1-estimate (l_1, u_2) even though we know that the true value is not in the interval (u_1, l_2) . From the above discussion it is clear that a) there is a trade-off between accuracy and communication and b) the width of an estimate is inversely proportional to the accuracy. This suggests that the width could be used as measure of inaccuracy assuming, of course, that the true value is to be found in one of the intervals of the estimate.

In reality, the disparate sensors may require a set of variables to represent each output. Each of these variables could itself be represented by a vector of values. If there are m values representing the state of the system and each value is represented by a k -interval abstract estimate, then the state of the system (each time the sensors are sampled) could be represented by a $(m \times k)$ matrix.

From the above discussion, the need for combining information at the nodes of the network to reduce the communication and to distribute the computation is clear. Any information combining algorithm should possess the following desirable properties:

- (i) low message complexity,
- (ii) distributed processing of information,
- (iii) reasonable fault tolerance.
- (iv) reasonable accuracy, i.e., the true physical value(s) should be very likely found in the combined estimate with the width of the estimate being low.

A variety of techniques including use of weighted averages, Kalman filter, multi-Bayesian approach, statistical decision theory, and fuzzy logic have been used in multisensor integration [10]. We shall present a simple and intuitively appealing integration algorithm based on estimates represented by intervals. Our method is based on the reasonable assumption that overlapping intervals are likely to contain the value(s)

of interest. This method could be considered as a variation of the use of weighted averages as applied to intervals. As we shall see in Section 4.4, even this simple integration algorithm becomes intricate after it is "clock-synchronized". A future goal of this research is to consider more sophisticated integration algorithms after clock synchronization is well understood. In the next section, we discuss the development of the integration algorithm. For explanatory purposes, we consider 1-interval abstract estimates.

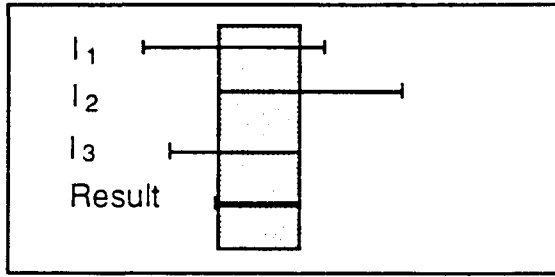
4.2 DEVELOPMENT OF THE INTEGRATION ALGORITHM

The idea behind the one interval integration algorithm is the following: an overlapped interval (if it exists) is more likely to contain the correct value of the physical variable of interest than a non-overlapped interval. This idea is similar to the N-modular Redundancy (NMR) technique used in many passive and hybrid fault tolerant systems ($N = 3$ in our case) [7]. Since we use intervals rather than single values as the abstract sensor estimates, it is possible, in general, to have two or more distinct overlapping intervals as the estimate. With a 1-interval estimate and a binary tree network, each node has to integrate three abstract estimates (one from the associated sensor and one each from the node's two children). The four ways in which the estimates could overlap are shown in Figures 4.1(a)-(d). The only case of distinct overlapping intervals that arises is shown in Figure 4.1(b). It is also possible that no intervals overlap (Figure 4.1(d)). In both these cases, we choose an estimate that *spans* the intervals of interest. By doing so, our estimate would contain sub-intervals that are unlikely to contain the correct value of the physical variable (examples are S_A in Figure 4.1(b), S_B and S_C in Figure 4.1(d)). From these observations it is also clear that the inaccuracy of the estimate is proportional to the width of the estimate. Each estimate, however, is a single interval and is transmitted as one message over the communication link. When information is not integrated, however, the number of messages sent by a PE at level i in the worst case is $(2^i - 1)$. Thus, there is a trade-off between accuracy and communication. The reader is referred to [6] for further discussions on message complexity. Further, note that when the non-root PE p does not perform any information integration, the computation time at the root node increases. The computation time with information integration turns out to be better than that without information integration by a factor of $O(n)$ where n is the number of nodes in the network [6].

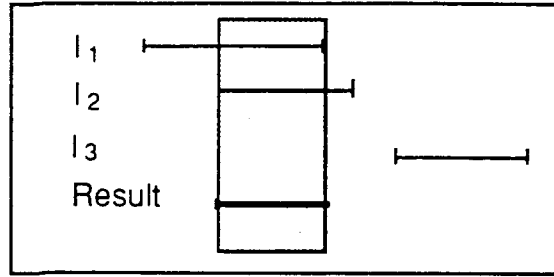
Consider two 1-interval estimates given by the intervals $\langle l_1, u_1 \rangle$ and $\langle l_2, u_2 \rangle$. Let $|u_1 - l_2| \equiv 0$. If $u_1 > l_2$ then the integration algorithm chooses the intersecting interval of width $(u_1 - l_2)$. Otherwise (i.e., $u_1 < l_2$) the algorithm chooses the spanning interval of width $(u_2 - l_1)$. Clearly, this is undesirable since small errors could result in choosing estimates with large derivations. We therefore require that for two intervals to be considered intersecting, the width of the intersecting interval be at least Δ (this width is a design parameter chosen depending on the sensor characteristics, the roundoff errors introduced by the PE, etc.)

4.3 CLOCK SYNCHRONIZATION IN DSN

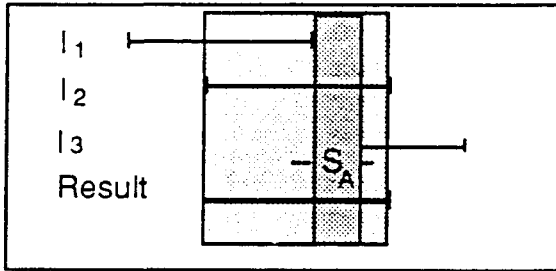
Since the physical sensor outputs typically change as a function of time, it is necessary for each of the estimates that are integrated to be "close to each other" temporally in order for the integration process to yield meaningful results. This is achieved by time-stamping each estimate. If the estimates from different sensors are to be synchronized to obtain a final estimate at each of the commander nodes, then it is necessary that the sensors be sampled at approximately the same time.



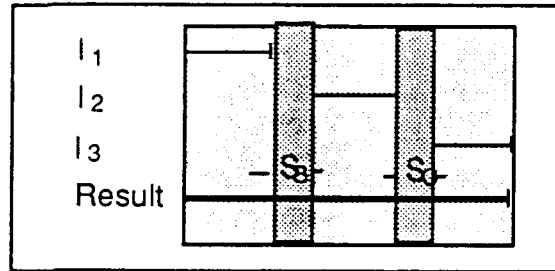
(a)



(c)



(b)



(d)

Figure 4.1 l_1, l_2, l_3 are the abstract estimates. Result is the 1-interval abstract sensor computed by the algorithm. Note that S_A, S_B, S_C are intervals included in the Result even though they are not intersecting intervals.

In a distributed environment such as ours, there is no central synchronized clock which regulates the activities of each node. Instead, each node is under the control of its own clock. Since a sensor responds to real time activities, it is convenient for the clock at each node to provide the real, i.e., physical time. Further, since the estimates from different sensors have to be integrated, it is convenient to have the time provided by each of the sensor nodes to be close to each other. The clock at each node may not be accurate because of a variety of reasons such as clock shift, variations in temperature, etc. Each clock therefore has to periodically synchronize with a more accurate clock. We assume the existence of a central time server on one PE of the network which, when requested for the time at t , provides the time $C(t)$. It is possible, though unlikely when fast real time communication is achievable, that in the case of some DSNs which are geographically spread out, there will be a need for many time servers. Since the time provided by two time servers may themselves not agree, clock synchronization with multiple time servers is more complicated than with a single time server (for a detailed treatment, see [9]). The

central time server itself periodically synchronizes with a universal time server. The latter is always accurate and lies outside our environment.

4.31 Clock Behavior and Synchronization

In this section we formalize the notion of integrating abstract estimates which are temporally "close to each other" by deriving an upper bound on the time interval between the arrivals of two abstract estimates at a node. Let $C_p(t)$ be the time provided by the clock on PE p at time t . Note that t itself is not observable by PE p , nor by any of the other processing elements.

Let us assume that the time $C_p(t_1)$ provided by PE p be greater than $C(t_1)$, the time provided by the central time server. Let p now synchronize with the central time server. It is now possible that the new time $C_p(t_2)$ is less than $C_p(t_1)$. An abstract estimate sent out later at $C_p(t_2)$ seems as though it was sent earlier. If the abstract estimates are integrated in a non First-in First-out fashion at a node, this could lead to problems. Hence, we require that the time on a PE to always increase monotonically. This monotonicity could be achieved by speeding up or slowing down a PE's clock each time a correction is required on synchronizing with the central time server [3].

From the foregoing discussion, we can now state the following requirements for proper synchronization (the subscript p refers to the PE p):

Requirement 1: Correct Time: The deviation in time of each clock is bounded.

$$|t - C_p(t)| \leq \epsilon_p \quad (1a)$$

$$|t - C(t)| \leq \alpha \quad (1b)$$

Where ϵ_p : is the maximum allowable deviation in time of a clock on a PE.

α : is the maximum allowable deviation in time of the clock on a central time server.

Requirement 2: Correct Rate:

Between resynchronizations, the drift rate of the clock is bounded,

$$\left| \frac{dC_p(t)}{dt} - 1 \right| \leq \kappa_p \quad (2a)$$

$$\left| \frac{dC(t)}{dt} - 1 \right| \leq \sigma \quad (2b)$$

Where κ_p : is the maximum allowable drift rate in time of a clock on a PE.

σ : is the maximum allowable drift rate of the clock on the central time server.

Requirement 3:

The clock on each of the PE p and the central time server increase monotonically.

We have assumed that the quantities $\epsilon_p, \kappa_p, \alpha, \sigma$ are all fixed and are known. (These quantities could be obtained from the specifications of the manufacturer's handbook.) If these quantities are time varying, then the analysis becomes very complicated. For simplicity we will assume that the constants ϵ_p is the same for all PEs and equals ϵ . Similarly it is assumed that $\kappa_p = \kappa$. From this simplification and Requirement 1, the following inequality follows. Let $C_q(t)$ be the time provided by the clock on PE q at time t .

Requirement 4: Synchronization Bound

$$|C_p(t) - C_q(t)| \leq 2\epsilon \quad (3)$$

Let $\xi_{\min}$ and $\xi_{\max}$ be the minimum and maximum values of the delay in receiving the message sent by the central time server to any PE (the message contains the time $C(t)$ at time t). Let $\delta_{\min}$ and $\delta_{\max}$ be the corresponding values for a message sent by a PE to its neighbor. Let γ be the maximum tolerance in time that a node can tolerate between intervals that can be integrated. This value of γ has to be derived from the sensor characteristics and the longest path between the leaf nodes and the commander node ($\lceil \log n \rceil$ in our case where n is the total number of PEs).

The following lemma and theorem are used to state precisely the conditions for combining abstract sensor estimates which are temporally "close to each other."

Lemma 1: Let a message be received by PE p at $C_p(t)$. Then this

message was sent by PE p or its neighbors in the interval

$$[C_p(t) - 2\epsilon - \delta_{\max}, C_p(t) + 2\epsilon - \delta_{\min}]$$

Proof: The proof is trivial if the message was sent by PE p's sensor since $C_p(t)$ lies in the interval.

Whenever a node sends a message, it time stamps the message with the current time. Because of the delay characteristics of the channel connecting PE p to its neighbor, the message was sent in the interval

$$[C_p(t) - \delta_{\max}, C_p(t) - \delta_{\min}]$$

according to p's clock. Let TS be the time stamp on the message. Then from (3), it follows that

$$TS \in [C_p(t) - 2\epsilon - \delta_{\max}, C_p(t) + 2\epsilon - \delta_{\min}] \quad \blacksquare$$

The time stamp may not belong to the interval, say, if the channel becomes faulty.

Definition: Let an abstract estimate time stamped at TS be received by PE p at time $C_p(t)$. The estimate is said to be proper if

$$TS \in [C_p(t) - 2\epsilon - \delta_{\max}, C_p(t) + 2\epsilon - \delta_{\min}].$$

Theorem 2: Let the three proper abstract sensor estimates I_1, I_2 and I_3 be received by PE p at times

$$C_p(t_1) < C_p(t_2) < C_p(t_3) \quad \text{respectively}$$

Then I_i ($i = 2, 3$) can be integrated, iff

$$(C_p(t_i) - C_p(t_1) + 4\epsilon + \delta_{\max} - \delta_{\min}) \leq \gamma \quad (4)$$

Proof: Since the estimates are proper estimates, we can deduce, using Lemma 1, that these estimates originated in the interval

$$[C_p(t_j) - 2\epsilon - \delta_{\max}, C_p(t_j) + 2\epsilon - \delta_{\min}] \quad (5)$$

$$(1 \leq j \leq 3)$$

If I_2 is to be integrated with I_1 , the spanning time interval of estimates I_1 and I_2 should at most equal γ , the maximum tolerance in time that a sensor can tolerate. The spanning interval is obtained from (5) with $j = 1, 2$ to be

$$C_p(t_2) - C_p(t_1) + 4\epsilon + \delta_{\max} - \delta_{\min}$$

Similar argument can be made for integrating I_3 with I_1 . Inequality (4) then follows. ■

4.32 Clock Resynchronization

Since the clocks on the central time server and each of the PE s drift, they have to be periodically reset. In this section, we derive an upper bound on the time period between resynchronizations.

Let T_s be the time period between resynchronization of the central time server.

Let T_c be the time period between resynchronization of the clock on a PE p .

The central time server resynchronizes itself every T_s seconds with a perfect universal time server which exists outside the environment of our DSN. The central time server also resynchronizes the clock on a PE every T_c seconds.

Let T_s^i and T_c^i be the periods corresponding to T_s and T_c as observed by the central time server.

Lemma 3: The period, as observed by the central time server, between synchronizations of the central time server is bounded by

$$T_s^i \leq \alpha \left(\frac{1}{\sigma} - 1 \right)$$

Proof: From (1b) and (2b), we can derive

$$\sigma T_s \leq \alpha$$

From (2b), T_s is in the interval $\left[\frac{T_s^i}{1 + \sigma}, \frac{T_s^i}{1 - \sigma} \right]$

Hence
$$\sigma \left(\frac{T_s^i}{1 - \sigma} \right) \leq \alpha$$

i.e.,
$$T_s^i \leq \alpha \left(\frac{1}{\sigma} - 1 \right)$$
 ■

Observation: When a PE p is resynchronized by the central time server at time t , there is a transmission delay of at least $\xi_{\min}$ before the value $C(t)$ reaches p . Hence

$$C_p(t') \geq C(t) + \xi_{\min}$$

Because of the drift in the central time server and the variable transmission delay $C_p(t')$ lies in the interval $[C_p(t) - \sigma T_s + \xi_{\min}, C_p(t) + \sigma T_s + \xi_{\max}]$. Hence, there could be an error of $(2\sigma T_s + \xi_{\max} - \xi_{\min})$ at the time that p 's clock is resynchronized. From (1a) and (2a),

$$2\sigma T_s + (\xi_{\max} - \xi_{\min}) + \kappa T_c \leq \epsilon \quad (6)$$

A restatement of (6) is Theorem 4.

Theorem 4: The time period between resynchronizations of the clock on a PE is bounded by

$$T_c \leq \frac{\epsilon - 2\sigma T_s - (\xi_{\max} - \xi_{\min})}{\kappa} \quad (7)$$

Proof: Arguments in the previous paragraph. ■

Observation: A restatement of (7) with observable times on the central time server is

$$T_c^i \leq \frac{\epsilon - \frac{2\sigma T_s^i}{1 - \sigma} - (\xi_{\max} - \xi_{\min})}{\kappa} - \alpha \quad (8)$$

Using the clock and network parameters, we can rewrite the equation (8) as follows:

$$T_c^i \leq \frac{\varepsilon - 2\alpha - (\xi_{\max} - \xi_{\min})}{\kappa} - \alpha \quad (9)$$

4.4 A CLOCK SYNCHRONIZED INFORMATION INTEGRATION ALGORITHM

This section presents a clock synchronized integration algorithm using the ideas presented in the previous two sections. At each node the integrated information is a 1-interval abstract sensor I_p computed from the three abstract estimates I_1, I_2, I_3 received at each node. Thus

$$I_p = D(I_1, I_2, I_3)$$

The computation of the function D is the information integration algorithm. As we have mentioned earlier, we consider only a 1-interval information integration algorithm. Further, we assume that the state of the system at any instant can be represented by a single value. (The algorithm can be extended in a straightforward manner for the case of a vector of values (see the Appendix) with each value being represented by a k -interval estimate.) The algorithm, called `One_Interval_Integrate` and shown in pseudo-Pascal on the next page, computes the 1-interval abstract estimate at a node p and sends it to its parent time stamping it with the current time CT . Procedure `FORM_I1_I2_I3`, which is repeatedly called as a procedure by `One_Interval_Integrate`, integrates the 1-interval abstract estimates that are temporally close to each other using lemma 1 and theorem 2.

Algorithm One_Interval_Integrate;

/ Δ The minimum width of overlap for two intervals considered intersecting. */*

/ CT Current time as seen by a processing element clock. */*

/ l_p Integrated abstract sensor estimate. */*

```

1.  Begin FORM_I1_I2_I3
2.    if (width ( $l_1 \cap l_2 \cap l_3$ )  $\neq \Delta$ ) then    /* See Fig. 4.1(a) */
3.       $l_1 \leftarrow$  lowest value in the intersecting interval;
4.       $h_1 \leftarrow$  highest value in the intersecting interval;
5.    else if (width ( $l_1 \cap l_j$ )  $\geq \Delta$ ) and (width ( $l_j \cap l_k$ )  $\geq \Delta$ ) then ( $i \neq j, j \neq k, i \neq k, 1 \leq i, j, k \leq 3$ )
        /* See Fig. 4.1(b) */
6.       $l_1 \leftarrow$  lowest value in the left intersecting interval;
7.       $h_1 \leftarrow$  highest value in the right intersecting interval;
8.      else if (width ( $l_i \cap l_j$ )  $\geq \Delta$ ) then ( $i \neq j, 1 \leq i, j \leq 3$ )    /* see Fig. 4.1(c) */
9.         $l_1 \leftarrow$  lowest value in the intersecting interval;
10.        $h_1 \leftarrow$  highest value in the intersecting interval;
11.      else    /* no intersecting intervals - see fig. 4.1(d) */
12.         $l_1 \leftarrow$  lowest value in the spanning interval;
13.         $h_1 \leftarrow$  highest value in the spanning interval;
14.      fi
15.    fi
16.  fi
17.   $l_p \leftarrow [l_1, h_1];$ 
18.  if  $p$  is not the root then
19.    send( parent( $p$ ),  $l_p$ , CT);
20.  fi
21. end.

```

Procedure FORM_I1_I2_I3

/* Computes the K-interval abstract sensors associated with a PE's abstract sensors and the PE's children*/

/* Buf is a FIFO buffer into which messages of the form < abstract sensor, time stamp > are sent by the abstract sensors and received by the PE p */

/* empty (buf) is true when there is no message in the buffer */

/* The variable CT in each PE has the current time */

```

1.  begin
2.      done := false;
3.      while not done do
4.          receive(l1, t);
5.          if t in  $[CT - 2\epsilon - \delta_{max}, CT + 2\epsilon - \delta_{min}]$  then /* from lemma 1 */
6.              done := true;
7.              /* i.e. l1 is a proper K-interval abstract sensor */
8.              first_sensor_time := CT;
9.              /* store the time at which first proper K-interval abstract sensor read */
10.         fi
11.     od
12.     l2, l3 ← 0;
13.     if p is the leaf PE then
14.         goto L3; /* exit a leaf PE that does not have any children */
15.     fi
16.     L1: empty_2&3 := false;
17.     while empty(buf) do
18.         /* if current time exceeds the max transmission delay d then set l2, l3 ← 0 */
19.         if ((CT - first_sensor) +  $4\epsilon + \delta_{max} - \delta_{min}$ ) >  $\gamma$  then /* from theorem 2 */
20.             l2 ← 0; l3 ← 0;
21.             empty_2&3 := true;
22.         fi
23.         if not (empty_2&3) then /* a K-interval abstract sensor was read */
24.             receive(l2, t);
25.             if not (t in  $[CT - 2\epsilon - \delta_{max}, CT + 2\epsilon - \delta_{min}]$ ) then /* from lemma 1 */
26.                 /* i.e., this abstract sensor is not a proper abstract sensor */
27.                 go to L1;
28.             fi
29.             L2: empty_3 := false;
30.             /* the 2nd proper abstract sensor has been read. Wait for the third one */
31.             while empty ( buf ) do
32.                 if ((CT - first_sensor) +  $4\epsilon + \delta_{max} - \delta_{min}$ ) >  $\gamma$  then /* from theorem 2 */
33.                     l3 ← 0;
34.                     empty_3 := true;
35.                 fi
36.             od
37.             if not (empty_3) then /* third K-interval abstract sensor was read */
38.                 receive( l3, t );
39.                 if not (t in  $[CT - 2\epsilon - \delta_{max}, CT + 2\epsilon - \delta_{min}]$ ) then /*from lemma 1*/
40.                     /* i.e., this abstract sensor is not a proper abstract sensor */
41.                     go to L2;
42.                 fi
43.                 go to L3;
44.             fi
45.         fi
46.     od

```

41. end procedure.

Observe that in our integration algorithm, the abstract sensor estimate associated with a PE at level i could affect the information integration in the same manner as the aggregate information produced by the children of the node rooted at level i . This method of integration could be altered by assigning weights to the intervals such that the individual sensor's abstract estimate is not as important as the abstract estimate produced by the aggregate. A sensor's abstract estimate could be weighted, for example, in inverse proportion to its level in the hierarchy.

In the Appendix, we present a computational characterization of the combination function for 1-interval abstract sensor estimate and for k -interval estimates using Heaviside functions.

5.0 FAULT TOLERANT ISSUES

Assuming that the PEs in the network are reliable, the main sources of faults in our system are the sensors, the channels connecting the PEs, and the various clocks including the central time server. We look at each of the faults and their effects.

Sensor Faults: An abstract estimate is *faulty* if the associated sensor produces a faulty output. Observe that if c ($c \geq 2$, $c = 2$ or 3 in our case) abstract estimates are non-faulty, then the intervals represented by the estimates must intersect. On the other hand, if two abstract estimates are faulty, then it is quite unlikely that the faulty estimates overlap because of the unpredictable nature of the associated sensor faults. Consequently, with the presence of even one non-faulty abstract estimate, the 1-interval abstract estimate is highly likely to contain the correct abstract estimate. Of course, with the presence of two faulty abstract estimates, the inaccuracy of the 1-interval abstract estimate increases since the interval of the estimate is wider. This reasoning carries over to the higher level nodes where the 1-interval abstract estimates are also integrated. Our network, hence, tolerates faults well and *degrades gracefully* as the number of faults increase. Further, the effect of a fault is *not* propagated but *localized* to that node and all its children. The network is thus resilient to failures. In the unlikely case of *all* the faulty abstract estimates having overlapping intervals, the number of faulty sensors that the network can tolerate is approximately $n/8$ (n is the number of nodes in each cluster; for a derivation of this result see [6]).

Observe that from a computational point of view, noise signals which significantly alter the output of a physical sensor and faulty sensors are indistinguishable- both result in a faulty abstract estimate. The arguments which we made to justify the fault tolerance of our scheme would now hold in a situation where there is a low signal to noise ratio and the noise signals picked up by the various sensors are uncorrelated.

Channel Faults: Observe that if the channel fails or is very slow, i.e., when the transmission time exceeds $\delta_{\max}$, a proper abstract estimate will not be received by the PE. The information integration algorithm essentially ignores the message passing through the channel but continues with the integration of the other estimates. Again, the DSN does not fail but degrades gracefully.

Clock Faults: Even though the clocks are imperfect, the integration algorithm works as long as the requirements specified in section 4.31 are met. When a clock on a PE fails, however, it would then transmit only the abstract estimate associated with its sensor (this is because the PE has no way of knowing that its clock has failed). Since the estimate is timestamped, it would not constitute a proper estimate to its parent node (unless the parent node has also failed in an identical manner which is highly unlikely!). Hence, the effect of a clock failure is limited to the node and the nodes rooted at that node. Further, since every clock is periodically resynchronized, the effect of a clock failure lasts only between resynchronizations.

In our DSN, we assumed the existence of a single central time server. The failure of the central time server is problematic, however. One solution is to have a number of central time servers. These are several intricate problems that need to be solved in using many central time servers. For a discussion, the reader is referred to [9].

Observe that in our DSN, the effect of the failure of a node or channel at the higher levels of the network is more drastic than a failure at the lower levels. It would be worthwhile to investigate DSNs in which these failures have the same effect irrespective of their position in the network.

6.0 CONCLUSION

The effective use of multisensor systems requires the solution of various problems relating to sensor models, the architecture of the sensor network, the integration of information at each node of the network, the cost of information transmission, and the fault tolerance of the network. The integration of information in real time requires the clocks at each of the nodes be synchronized. Synchronization of clocks is a non-trivial task in such distributed sensor networks. In this paper we have proposed an architecture based on binary trees for the sensor network and considered a) how information could be integrated in real time when clocks at the nodes are not perfect; b) how information could be efficiently transmitted without incurring heavy communication costs; c) how the network is tolerant to certain types of faults. As far as we are aware, this is the first study in the area of distributed sensor networks that looks at the various computational issues in such networks. Since our focus has been on the issues just mentioned, we have chosen to represent information with a simple and elegant model based on real valued interval due to Marzullo [12]. We are aware that sensor modeling is itself a detailed area of study [2, 10]. We have assumed that the output of each sensor is a physical value. The above discussion and results easily extend to the case when the output of a sensor is a vector rather than a single value.

This study could be extended in several directions. A straightforward extension, as suggested at the end of Section 4, is to assign weights to the abstract estimates produced as a function of its level in the hierarchy. A future goal of our project is to investigate more sophisticated integration algorithms and also other sensor models. Depending upon the application that a DSN gets used for, other DSN architectures could be studied. An architectural study could also be based on the fault tolerant properties desired. Finally, this study can be extended to the integration of complementary information.

ACKNOWLEDGMENT

The authors would like to thank the three anonymous referees for their comments which helped improve both the form and content of the paper.

APPENDIX

COMPUTATIONAL CHARACTERIZATION

A.1 1-Interval Abstract Sensor Estimate

We define a useful and familiar function which will form the germ of the algebra of information integration, to characterize the output abstract sensor estimate I_p which depends on the input abstract sensor estimates I_1, I_2, I_3 . We first consider the case of 1-interval abstract estimates and then go on to generalize the idea to k-interval abstract estimates.

$$\text{Define } H_a(x) = \begin{cases} 0 & \forall x < a \\ 1 & \forall x \geq a \end{cases}$$

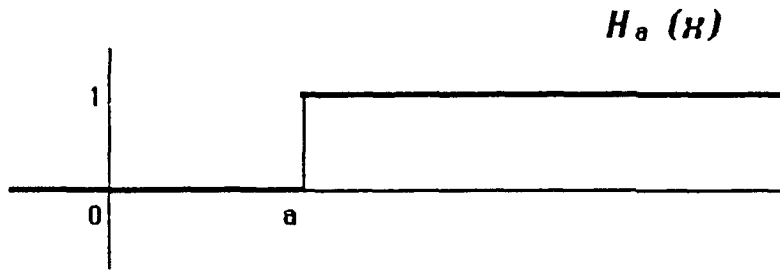


Figure A.1 The Heaviside function.

$H_a(x)$ is called the Heaviside function (Figure A.1). Using this function we can construct a function which takes the value 1 over a desired interval say $[a,b]$ and zero everywhere else as follows :

Set

$$H_{[a,b]}(x) = H_a(x) (1 - H_b(x))$$

This clearly is the required function (Figure A.2).

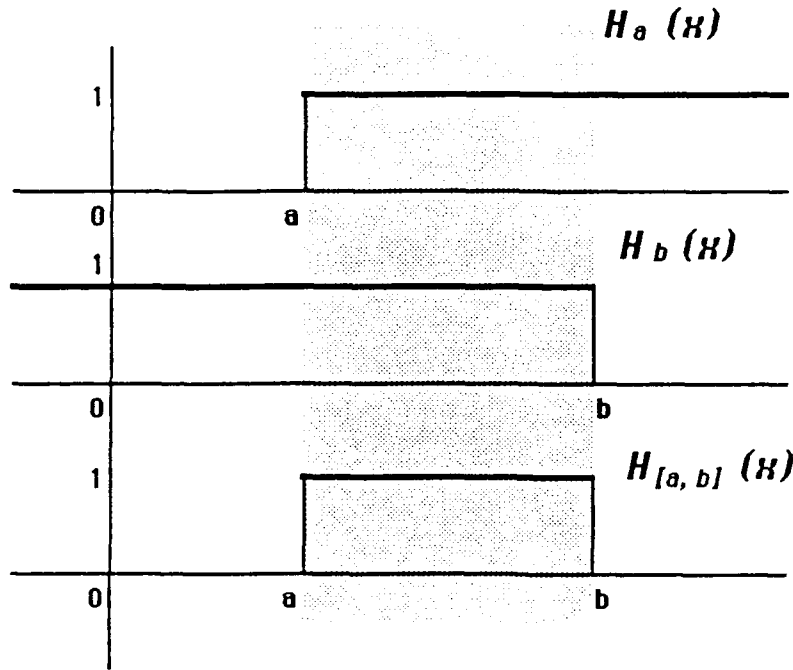


Figure A.2 Construction of the function of the desired interval.

Indeed :

$$H_{[a, b]}(x) = \begin{cases} 1 & a \leq x < b \\ 0 & \forall x \geq b \end{cases}$$

The set of values for which a function is non-zero is called the support Supp of the function.

$$\text{Thus, } \text{Supp}[H_{[a, b]}(x)] = [a, b).$$

It is easy to see that

$$\int_{-\infty}^{\infty} H_{[a, b]}(x) dx = b - a.$$

This gives the width of the interval $[a, b)$.

Let $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$, $I_3 = [a_3, b_3]$ be the 1-interval abstract sensor estimates entering the processor p . We wish to integrate the above inputs into an output which is a reliable index of the sensor values. To achieve this we follow the prescription of the

integration algorithm and obtain a functional realization using the Heaviside function and other auxiliary functions.

$$\text{Define } H_{I_j}(x) = H_{a_j}(x) (1 - H_{b_j}(x)) \quad \forall 1 \leq j \leq 3.$$

$$\text{Thus } H_{I_j}(x) \text{ has support } I_j, \text{ i.e., } \text{Supp}[H_{I_j}] = I_j = H_{I_j}^{-1}(1)$$

We consider three possibilities for the 1-interval sensor inputs with regard to their relative positions inducing overlap or reinforcement of sensor information and integrate in each case the inputs in accordance with the integration algorithm to get a fault tolerant output I_p .

Case (1):

$$I_j \cap I_k = \emptyset \quad \forall j \neq k, 1 \leq j, k \leq 3.$$

Here the function $H_{I_1} + H_{I_2} + H_{I_3}$ has its support as a reliable estimate of the possible region of the value of I_p : the output abstract estimate. The support is the union of the disjoint intervals I_1, I_2, I_3 (Figure A.3).

$$H_{I_p}(x) = H_{I_1}(x) + H_{I_2}(x) + H_{I_3}(x)$$

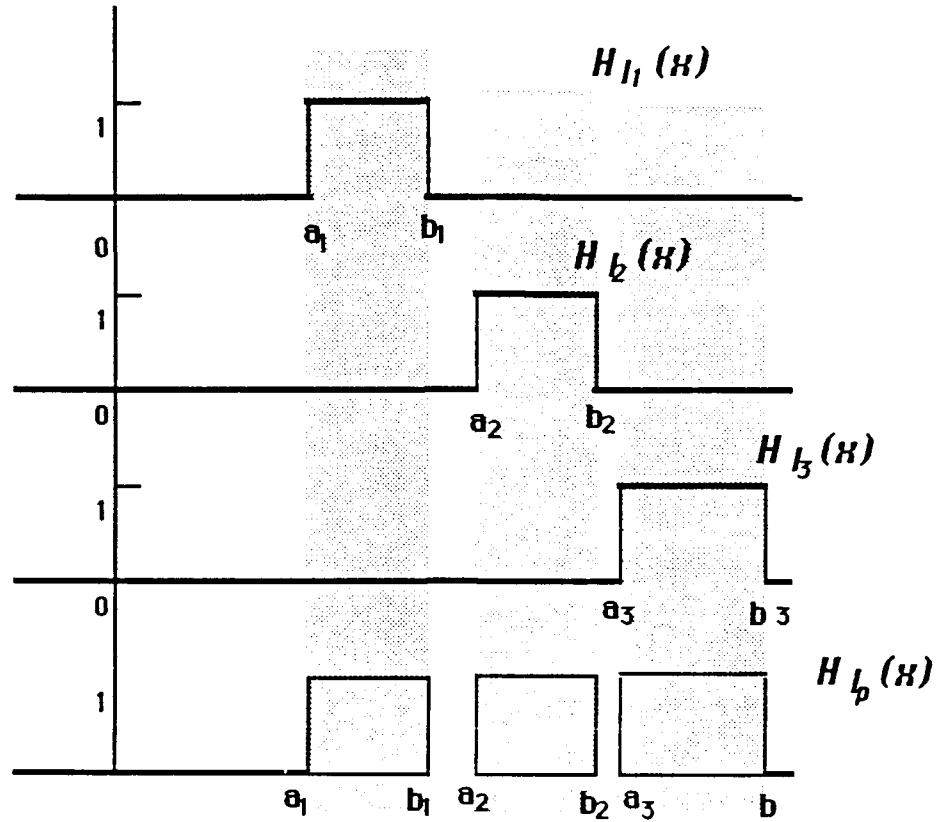


Figure A.3 Support function characterization for the union of disjoint intervals.

Case (2) :

$$(I_j \cap I_k \neq \emptyset \text{ for some } j \neq k \ 1 \leq j, k \leq 3) \wedge (I_1 \cap I_2 \cap I_3 = \emptyset)$$

Here the function $H_{I_1} H_{I_2} + H_{I_2} H_{I_3} + H_{I_3} H_{I_1}$ has its support as a reliable estimate of the possible region of the value of I_p . The support is the overlap regions of the disjoint intervals I_1, I_2, I_3 (Figure A.4).

$$H_{I_p}(x) = H_{I_1} H_{I_2} + H_{I_2} H_{I_3} + H_{I_3} H_{I_1}$$

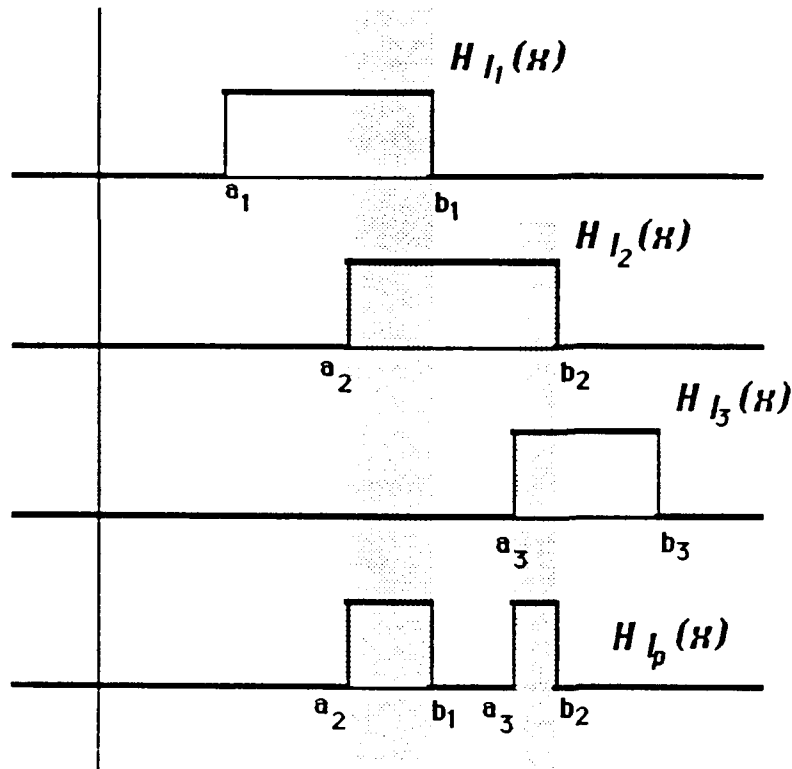


Figure A.4 Support function characterization of overlap regions of the disjoint intervals I_1, I_2, I_3 .

Case (3) :

$$I_1 \cap I_2 \cap I_3 \neq \emptyset.$$

Here the function $H_{I_1} H_{I_2} H_{I_3}$ has its support as a reliable estimate of the possible region of the value of I_p . The support is the common overlap region of the intervals I_1, I_2, I_3 (Figure A.5).

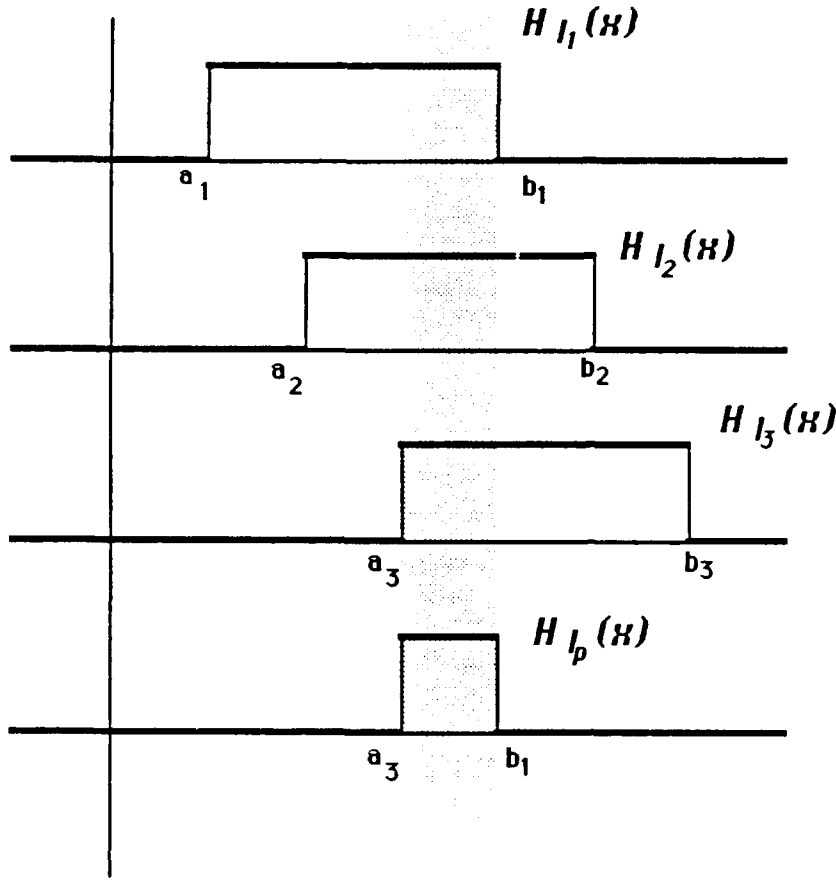


Figure A.5 Common overlap regions of the intervals I_1 , I_2 and I_3 .

However, it is desirable to have just one function subsuming the above three cases so that the job of checking for the kind of overlap of intervals can be avoided. To effect this computationally we define a selector function f on functions as follows :

If $g : \mathbb{R} \rightarrow \mathbb{R}$ is any single valued real function, then define :

$$f(g(x)) = \begin{cases} 0 & \forall x \in \mathbb{R} \text{ if } g(x) > 1 \text{ for some } x \in \mathbb{R} \\ 1 & \forall x \in \mathbb{R} \text{ if } g(x) \leq 1 \text{ for } \forall x \in \mathbb{R} \end{cases}$$

This function chooses the zero function or the function $g(x)$ at hand accordingly as to whether $g(x)$ takes value greater than 1 or stays strictly less than or equal to 1. We need to choose in cases 1,2 and 3 the appropriate function constructed under the case. While amalgamating these functions by means of a sum we need to suppress the other two functions since they are irrelevant. Hence we operate the selector function f taking

advantage of the fact that the irrelevant functions are themselves zero or take values greater than 1.

Now define $H_{Ip}(x)$ as follows :

$$H_{Ip}(x) = f(H_{I1} + H_{I2} + H_{I3}) + f(H_{I1}H_{I2} + H_{I2}H_{I3} + H_{I3}H_{I1}) + f(H_{I1}H_{I2}H_{I3})$$

$I_p = H_{Ip}^{-1}(x)$ is the desired output abstract sensor estimate. This is not in general a 1-interval estimate, but gives a sharper and narrower estimate than the associated single interval estimate in the algorithm. We however can obtain a single interval estimate also by filling up the gaps as shown later.

The total interval width of I_p is given by

$$W(I_p) = \int_{-\infty}^{\infty} H_{Ip}(x) dx$$

We generalize these ideas and results to obtain a functional characterization of the combination function for a k-interval abstract sensor estimate.

A.2 The Combination Function for k-Interval Estimates

Suppose $I_p = \{I_m^j\}_{m=1}^k$, where $I_m^j = [a_m^j, b_m^j]$, $1 \leq j \leq k$ are the proper k-interval

abstract sensor estimate inputs to the processor p. I_j is clearly a vector of k components.

If $I_1, \dots, I_s$ are s disjoint intervals and $I = \bigcup_{j=1}^s I_j$, then define

$$I^\wedge = \{\alpha t + \beta(1-t) \mid \alpha, \beta \in I \text{ and } t \in [0, 1]\}$$

Thus if

$$I_j = [a_j, b_j], 1 \leq j \leq s \text{ and } a_1 < b_1 < \dots < a_s < b_s, \text{ then } I^\wedge = [a_1, b_s]$$

$\wedge : I \rightarrow I^\wedge$ can be called the filler function and from its definition, it is clear that it fills up the gap between disjoint intervals to give one single connected interval.

We now carry out the integration of the vectorial input sensor estimates I_1 I_2 and I_3 componentwise. The componentwise integration is the same as the 1-interval integration except that we use the filler function to get a 1-interval output estimate, so that the vectorial output is also a k component vector. To this effect :

$$\text{Define } I_p = \left\{ I_m^\wedge \right\}_{m=1}^k$$

$$\text{where } I_m^p = \text{Supp} \left[H_{I_m^p} \right]$$

$$= \text{Supp} \left[f(H_{I_m^1} + H_{I_m^2} + H_{I_m^3}) + f(H_{I_m^1} H_{I_m^2} + H_{I_m^2} H_{I_m^3} + H_{I_m^3} H_{I_m^1}) + f(H_{I_m^1} H_{I_m^2} H_{I_m^3}) \right]$$

Thus I_p is the output k -interval estimate where the integration has been performed component-wise following the example of the 1-interval estimates discussed above. The width vector of the k -interval estimate output I_p is given by :

$$W(I_p) = \left\{ \int_{-\infty}^{\infty} H_{I_m^p}(x) dx \right\}_{m=1}^k$$

$W(I_p)$ is a vector with k -components with each component being a non-negative real number and not an interval as in the case of I_1 , I_2 , I_3 and I_p . We may further obtain a scalar measure of the width by integrating in some manner desirable (e.g., taking a weighted sum) the components of $W(I_p)$

Thus we have described functionally the scheme of integration of the abstract sensor inputs I_1 I_2 and I_3 to obtain an abstract sensor output in the case of 1-interval and then generalized the idea to k -interval estimates by componentwise integration. This integration is in accordance with the integration algorithm. We have treated here the case of the sensors giving a uniformly distributed signal over the associated interval.

However, if the sensor can weight the values of its dense output, then we can introduce the appropriate weight function in place of products of Heaviside function. The rest of the analysis may be carried out in an analogous manner with a few modifications. The interval width measures in the case of non-uniform distribution of signal values are genuine integrals and not mere measures of rectangular areas as in our present discussions. The reason for writing interval width measures in this discussion as integrals is to suggest this generalization.

The number of inputs considered so far is three as dictated by the architecture of the sensor network. However, our functional representation is easily extendible to the case when the architecture is not just binary but n-ary at each processor.

REFERENCES

- [1] Durrant-Whyte, H.F., "Consistent Integration and Propagation of Disparate Sensor Observations," Int. J. Robot. Res., V. 6, No. 3,4, 1987, pp. 3 - 24.
- [2] Durrant-Whyte, H.F., "Sensor Models and Multisensor Integration," Int. J. Robot. Res., V. 7, No. 6, 1988, pp. 97-113.
- [3] Gusella, R., Zatti, Stefano, "The Accuracy of the Clock Sync. Achieved by TEMPO in Berkeley Unix 4.3 BSD", IEEESE, July 1989.
- [4] Henderson, T.C., Allen, P.K., Mitchie, A., Durrant-Whyte, H., Snyder, W., Eds., "Workshop on Multisensor Integration in Manufacturing Automation," Dept. of Comp. Scie., Univ. of Utah, Snowbird, Tech. Rpt. UUCS-87-006, Feb. 1987.
- [5] Henderson, T.C. and Weitz, E., "Multisensor Integration in a Multiprocessor Environment," in Proc. ASME Int. Comput. in Engr. Conf. and Exhibition, R. Raghavan and T.J. Cokons, Eds., New York, NY, Aug. 87, pp. 311-316.
- [6] Jayasimha, D.N., Iyengar, S.S., "Information Integration and Synchronization in Distributed Sensor Networks," Tech. Rpt., OSU-CISRC-2/91-TR9, Dept. of Comp. and Inf. Sc., The Ohio State University, Columbus, OH.
- [7] Johnson, B.W., "Design and Analysis of Fault Tolerant Digital Systems," Addison-Wesley, 1989.

- [8] Kashyap, R. L., Oh, S. G., Madan, R. N., "Estimation of Sinusoidal Signals with Colored Noise Using Decentralized Processing," IEEE Trans. on ASSP, Vol. 38, No. 1, 1990, pp. 91-104.
- [9] Lamport, L., "Synchronizing Time Savers," DEC SRC TR, June, 87.
- [10] Luo, R.C. and Kay, M.G., "Multisensor Integration and Fusion in Intelligent Systems," IEEE Trans. on SMC, V. 19, No. 5, 1989, pp. 901-931.
- [11] Luo, R.C., Lin, M. and Scherp, R.S., "Dynamic Multi-sensor Data Fusion System for Intelligent Robots," IEEE J. Robot. Automat., V. RA-4, No. 4, 1988, pp. 386-396.
- [12] Marzullo, K., "Implementing Fault-Tolerant Sensors," TR89-997, Dept. of Comp. Science., Cornell Univ., May 1989.
- [13] Wesson, R. et. al., "Network Structures for Distributed Situation Assessment", IEEE Trans. on SMC., Jan. 1981, pp. 5-23.
- [14] Yemini, Y., "Distributed Sensors Networks(DSN): An Attempt to Define the Issues," Proc. of the Distributed Sensors Network Workshop, Carnegie Mellon Univ., 1978, pp. 53-60.
- [15] Zheng, Y.F., "Integration of Multiple Sensors into a Robotic System and its Performance Evaluation," IEEE Trans. on Robot. Automat., V. 5, No. 5, Oct. 1989, pp. 658-669.

LIST OF SYMBOLS

Δ	The minimum width of overlap for two intervals to be considered intersecting.
ϵ_p, ϵ	Maximum allowable deviation in time of a clock on a PE.
α	Maximum allowable deviation in time of clock on the central time server.
κ_p, κ	Maximum allowable drift rate in time of a clock on a PE.
σ	Maximum allowable drift rate of the clock in the central time server.
$\delta_{\min}, \delta_{\max}$	Minimum and Maximum channel transmission delay.
$\xi_{\min}, \xi_{\max}$	Minimum and Maximum delay in receiving the message sent by the central time server to any PE.
γ	Maximum tolerance in time that a node can tolerate between intervals that can be integrated.
CT	Current time as seen by a processing element's clock.
I_i	i^{th} abstract estimate.
$(u_i - l_i)$	Width of an interval.