Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

SIMULATION OF A REAL-TIME FAULT DETECTION AND ANALYSIS SYSTEM FOR A CIM TOKEN RING NETWORK

Mark D. Pardue

Department of Electrical Engineering
Old Dominion University
Norfolk, Virginia 23529

James D. Palmer

School of Information Technology
and Engineering
George Mason University
Fairfax, Virginia 22030

ABSTRACT

One of the major problems with conventional Manufacturing Automation Protocol (MAP) that is based on ISO/OSI international standards lies in the inability to handle real-time interrupts. The token passing architecture makes no provisions for situations that call for breaking into the token passing operation out of sequence.

Real-time fault detection and correction for CIM operations in a distributed system architecture is introduced in this paper. Initially, an architecture is formulated that addresses problems related to the ability to handle real-time data and uncertain information. The technique developed establishes a secondary channel for reallocation of critical resources and provides for prioritization of channel assignment. An intelligent interface is designed to handle situations involving uncertain information that is initiated by sensors located in the CIM facility. Simulation results show that this approach satisfies 99.1% of all data transfer needs and all of the highest priority data transfers were satisfactorily made.

1. INTRODUCTION

The system architecture that provides the most efficient and effective use of computers in CIM is a distributed system architecture (DSA) with hierarchical control interconnected via local area networks (LANs) [Palmer and Liang, 1987]. DSA provides for operation of multiple simultaneous CIM functions and permits information exchange between compatible devices without end user or application concern with communications detail. DSA interfaces directly with the Manufacturing Automation Protocol (MAP) through use of the international standards and recommendations as specified in the ISO/OSI seven-layer protocols. [Day and Zimmerman, 1983] The layered architecture represents a continuum of functions involved in communications arranged in a logical order to provide inter-connectivity. MAP was developed out of user requirements for non-proprietary, remote management capabilities in a multi-vendor network [Green, 1987] and allows intelligent factory equipment such as robots, computers, machine tools, and programmable controllers to communicate with true plug-in compatibility [Palmer, 1987]. However, this system architecture in its present form of adhering to IEEE Standard 802.4 will not handle real-time interrupts for fault detection and analysis. A study of dynamic assignment in hierarchical networks found that use of a heuristic assignment strategy had qualities similar to those found in probability strategies [Nance and Moose, 1988, and Moose, 1989]. In the approach taken in this work, we utilized a separate communications channel managed by an expert system to provide the assignment strategy for real-time interrupts that contains uncertain information.

The goal of managing information and not simply manipulating data requires an architecture that must be able to address problems related to real-time data and uncertain information. Intelligent interfaces that examine and sort information and assist in the decision making must become a compliment to existing nodes and databases to realize this goal of handling real-time signals that contain information uncertainty. As the use of inferential processes and databases becomes integrated, it becomes feasible to utilize the rich information base in real-time signals and uncertain information available within the manufacturing enterprise. This relationship between information and the manufacturing enterprise [CAM-I Report, 1988], has led to the development of "islands of information". Use of an information backbone unifies the information within the enterprise and enables the enterprise functions to be integrated.

In this paper we address the issues of interfacing individual subsystems to the backbone that serve the distributed information system. Current systems architecture for CIM interface nodes do not allow for:

1. Reallocation of communications resources for handling real-time data transfer requirements; and
2. Uncertain information related to conditions in the distributed information system.

One possible solution to these problems is to add intelligence to the interface nodes used to connect the subsystems to the communications backbone, and then utilize generally accepted interface standards. The intelligent interface node design that we have developed consists of a Knowledge-based Expert System (KBES) integrated with an interface node. The broadband token-passing backbone specified in MAP is the standard for CIM applications, and we have selected this standard approach for the application of intelligent interface nodes to be connected [Allen, 1986]. This node is intended to perform reallocation of communications resources to satisfy real-time data transfer requirements enhancing MAP network performance to handle time-critical functions. It also effectively handles uncertainty in the area of diagnostics in CIM implementations. The node performs reallocation of communications resources to satisfy real-time data transfer requirements and enhances MAP network performance so as to enable handling of time-critical functions. It also effectively allows handling uncertainty in the area of diagnostics in CIM implementations.

The intelligent interface node was simulated to provide an indication of performance. In the simulation runs, the intelligent interface node was able to satisfy 99.1% of the real-time data transfer resource shortfalls that otherwise would have resulted without implementation of the KBES, including all of the highest priority data transfers. Also, the KBES was able to correctly deduce problem situations in the manufacturing cell operations and take appropriate corrective action, including the servicing of priority interrupts. These problem situations involved uncertain data and conflict. Thus, the intelligent interface node was able to effectively address the two major problem areas in CIM applications.

2. TECHNIQUE FOR REALLOCATION OF REAL-TIME COMMUNICATIONS RESOURCES

In any environment communications resources are limited, and when communications requirements are greater than the resources available to support them, some requirements will not be met. Many systems provide means for prioritizing these communications requirements and subsequently attempting to satisfy higher priority requirements at the expense of lower priority requirements [Nance and Moose, 1988]. In a CIM environment, the IEEE 802.4 standard, access to the backbone is through token-passing which makes no allowance for prioritizing requirements. The token to allow a station (node) to transmit is passed from one station to another in a predetermined order, with no deviation from that order possible [Weigard, 1987]. Thus, there is no method to request the token in the case of a station suddenly encountering a situation where it needs to transfer data, but does not hold the token.

2.1 Next Station Reallocation Approach

One approach to resolution of this problem in a token bus network environment is the re-prioritization of data transfers through the use of an alternate or secondary channel architecture that provides control information between intelligent interface nodes that have been designed to address this particular problem [Pardue and Palmer, 1988]. This use of a secondary channel for network control is a common approach in circuit-switching applications (common-channel signaling) and has been given consideration in some recent work with IEEE Standard 802.3 which deals with LANs [Spracklen and Smythe, 1987]. Although the token-passing order is determined using IEEE Standard 802.4, it can be altered if the nodes involved cooperate in the reordering. However, the standard for the token-passing protocol requires each node to pass the token to the next station (NS), upon completion of its data transfer which was determined upon network start-up. Following this protocol, the NS of a station (node) is altered only upon the entry of a new node in the network, or the exit of an existing node from the network (including node failure). This is achieved through procedures requiring the use of a data transfer channel.

The next station (NS) reallocation approach that we have taken to resolve these critical issues establishes a secondary channel (control channel) for real-time reallocation of data transfer resources. The media access protocol, CSMA/CD (IEEE Standard 802.3), for the control channel has been chosen because it provides excellent responsiveness, which is essential for control channel operations. CSMA/CD has been found not to operate well under heavy load. Therefore the use of the control channel must be restricted to the transfer of the control information required for real-time reallocation of data transfer resources of the main data transfer channel. This channel uses the token bus (IEEE Standard 802.4) protocol, as specified in MAP.

The minimum information required from each node to determine data transfer priorities is:

1. Interface node ID;
2. Time-to-live (TTL) of the real-time data to be transferred, in number of data transfer opportunities; and
3. Next data transfer opportunity.

With this information, the restrictions of the required data transfer can be calculated, and any surplus communications capacity can be identified and reallocated. This is illustrated using the very simple example shown below in Table 1.

Table 1. Identification and Reallocation of Communications Capacity

Node ID	Time-to-Live	Next Data Transfer	Surplus (+) Shortfall (-)
1	4	3	+1
2	3	4	-1

It is clear from this example that simply switching the order of data transfer opportunities will solve the shortfall for node 2. It is also clear that there is no adverse affect on the data transfer for node 1.

With the approach that we have taken, each node periodically transmits information over the control channel to indicate its data transfer requirements, as discussed above. This transmission is in broadcast mode for all nodes to receive. Both the TTL and NT are in stated terms of "Time Periods" equal to the maximum allowable time between one node starting data transfer and the next node starting data transfer. This time period is equal to the token holding time defined by the network, the delays associated with station (node) delay, and the transmission delay, which is denoted slot time by IEEE Standard 802.4.

2.2 Node Prioritization using the Reallocation Approach

The concept of prioritization is an essential feature of our approach and is embedded within the reallocation technique for nodes transmitting critical real-time data. There are several examples that require priority for the transmission of real-time data. One example would be for priority interrupts in life-threatening situations or when damage to equipment is imminent. Another example is when it is critical that all real-time data from a node be transmitted with no interruptions or missing data points, such as is found in situations with sensitive continuous processes (e.g., mixing chemicals, etc.). Another would be fault detection for overheating in bearings that could disrupt the entire manufacturing process and potentially lead to life-threating fires and loss of equipment.

One possible priority scheme designed to address problems such as these is presented in Table 2. Priorities for data transferred are assigned values from 0 to 4, with 0 being the least priority and 4 covering the instance of a life-threating event. Using this scheme, any node with a priority 4 data transfer would have its data transfer requirements met before a node with a priority 3 data trans-

fer, priority 3 before priority 2, priority 2 before priority 1, and priority 1 before priority 0. If two nodes of equal priority vie for the same data transfer opportunity, the node with the highest ID (station address) has priority over the other. This is in concert with the IEEE Standard 802.4 priority scheme and will prevent deadlock situations. In fact, node IDs can be initially assigned based on the relative importance of the information the node will be transmitting, or assignment may be made on any other criteria required by the situation in the CIM facility.

Table 2. Priority Scheme used for Reallocation of Real-Time Data Transfer

Priority	Description
4	Life-Threatening
3	Equipment-Threatening
2	Continuously-Critical Real-Time Data
1	Last Data Transfer Opportunity Missed
0	No Priority

3. TECHNIQUE FOR DEALING WITH UNCERTAINTY

Our approach for handling uncertainty in a CIM environment related to problems in a manufacturing cell indicated by sensor data is based on the methodology for handling uncertainty developed for the MYCIN Model. Sensor readings are used to determine positive or negative evidence instances, as these occur. Positive evidence instances indicate that a problem exists, negative evidence instances indicate the absence of a problem.

Evidence instances are used to calculate Measures of Belief (MBs) and Measures of Disbelief (MDs) in problem hypotheses. The MB and MD for each problem hypothesis are combined to determine a Certainty Factor (CF) for the problem hypothesis. When a CF is above a predetermined threshold for a problem hypothesis, corrective action is taken. The technique also identifies situations where inoperative sensors lead to conflicting hypotheses. Troubleshooting procedures for the involved sensors can then be printed out for the cell operator. This technique effectively handles uncertainty for these areas of diagnostics in CIM implementations.

3.1 Adaptations to the MYCIN Model

3.1.1 Handling Serious Problems

In a manufacturing environment, the cost of not identifying an emergency situation that exists has significantly greater importance than the cost of identifying an emergency situation when in fact it does not exist. Thus, it becomes clear that for a hypothesis that has dangerous consequences, the technique used to determine if the hypothesis is correct should err toward false alarm rather than allowing a dangerous situation to go unidentified. Given the setup procedure of hypotheses for this adaptation of the MYCIN Model, it becomes a simple matter to resolve conflict in potentially dangerous situations. All hypotheses in this adaptation are established in a negative sense, that is, a hypothesis alleges that there is a problem. All MBs support the allegation that there is a problem. All MDs refute the allegation that there is a problem. For those hypotheses with dangerous consequences (e.g., equipment damage, personnel injury) all MDs are removed from the CF calculations. This has the effect of being overly cautious and erring on the conservative side in discounting a possibly dangerous problem. This is exactly the desire in a manufacturing environment. The negative impact in the use of this procedure is that dangerous problems may be identified when in fact they do not exist. However, this is a viable trade-off given the positive effects of the procedure.

3.1.2 Conflict and Incorrect Evidence

A potential inconsistency can be seen whenever both a substantial value of MB and a substantial value of MD exist for a hypothesis H_i based upon evidence instances E_j (generating an MB) and $\sim E_k$ (generating and MD). This situation is caused by conflict and could indicate an invalid evidence instance. To resolve the apparent inconsistency in this case, hypothesis H_i would be presented to the operator with instructions to check the sensors associated with the applicable evidence instances. If hypothesis H_i is correct and the sensor providing evidence instance $\sim E_k$ is bad, the operator would easily be able to determine the problem and take corrective action replacing the sensor. If hypothesis H_i is incorrect, then the sensor providing evidence E_j is bad and the operator would take corrective action by replacing this sensor and correcting the problem indicated by H_i. Given any other similar problem situation, this same procedure will be able to isolate the faulty operation.

3.1.3 Missing or Incomplete Evidence

Given the evidence structure developed for this technique, incomplete evidence results from neither E_i or $\sim E_i$ existing. This state in itself means that there is some problem in the system. This could be caused by a bad sensor producing no output, by a problem in the communications link from the sensor (e.g. broken wire, protocol error), or because of noise making the sensor reading undecipherable.

For any situation involving missing or incomplete evidence E_i or $\sim E_i$, the worst-case evidence instance E_i is assumed. This allows all possible problems to be identified. This may lead to belief in one or more hypotheses, but with the caveat that these hypotheses may not really be correct. In this situation, the sensors associated with these problem hypotheses would be investigated through troubleshooting procedures supplied to the operator. The operator would also be given instructions to check the sensors and communications links from the sensors associated with E_i or $\sim E_i$ because of missing or incomplete evidence. This would also apply to situations where a sensor continues to indicate a problem, when in fact there is no problem. The troubleshooting procedures should be able to isolate this type of sensor failure. As before, this approach is extremely conservative, but is much preferable to one that does not identify system problems.

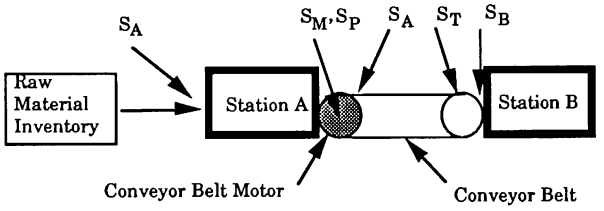
3.2 Implementation

The technique described above for handling uncertainty is incorporated into a rule-based expert system that provides support for problem identification and correction. This system is used to handle problems with the control of the particular manufacturing cell served by the

interface node containing the production system. Given the well-defined domain of a CIM implementation, the precise definition of the rules to be used in the production system for each cell should be possible. In fact, production rules can be pieced together (and even modified for flexible manufacturing operations) in a modular fashion. This requires sets of data for each sensor type and type of machine in use to be combined in a manner defined by the layout of the manufacturing cell.

4. SIMULATION OF INTELLIGENT INTERFACE NODE

The approach that we have taken for reallocation of real-time data transfer resources was simulated to demonstrate the effectiveness of our solution to this essential problem with token passing networks. The sample manufacturing cell shown in Figure 1 was selected as being typical of situations found in small factories or in islands of automation. Figure 1 depicts an island of automation with raw material being fed to two separate operations represented by Stations A and B. Sensor locations and applications are designated by the letter S. We will use this information during the simulations that we perform to analyze or approach to real-time fault detection and correction.



Sensors
- S_A - Parts Input to Station A (Binary)
- S_M - Parts Input to Station B (Binary)
- S_P - Motor Stator Temperature
- S_A - Power to Motor
- S_T - Conveyor Belt Tension
- S_B - Conveyor Belt Velocity

Figure 1. Manufacturing System Example

Since much of the information from real-time fault detection contains a high degree of uncertainty, we have introduced this aspect into the simulation so as to be as close as possible to actual practice as possible. The technique for handling uncertainty was simulated to demonstrate the effectiveness of our approach under uncertain conditions such as those found in general operating situations. Three separate simulations were developed to analyze the intelligent interface node operation. These are real-time fault detection using deterministic information; real-time fault detection under uncertainty conditions; and a combination of these two. The last simulation was accomplished by combining these. It was developed to demonstrate the entire operation of the intelligent interface node. The simulations are developed on an IBM-compatible personal computer in the object-oriented programming language Objective-C and the programming language C. Objective-C was chosen because it can easily model objects such as nodes and sensors, and when used with Microsoft C provides executable C code that runs on IBM PCs or compatibles. Together these hardware and software selections provide very good portability and will allow future work to build upon these efforts.

4.1 Simulation for Reallocation of Real-Time Communications Resources

4.1.1 Assumptions Used in Simulation

A broadband communications backbone using the IEEE Standard 802.4 token-passing protocol such as that commonly found in factory operation is assumed to be the network of choice. The simulation was developed to handle 100 interface nodes, with each having a total potential of nine subsystems involved in operations. It is our observation that a small factory may have five to ten interface nodes, and the selection of 100 nodes for the simulation represents a situation that might occur in a medium-sized factory or a worst-case situation for a small factory. The number of interface nodes was restricted for purposes of efficiency of simulation, rather than being a limitation on the ability to handle larger situations.

Five priority levels of data transfer were assumed, although the simulation could have been expanded to handle many more priority levels. The priority of the data transfer for each node is determined randomly with the distribution presented in Table 3. It was further assumed that the majority of data traffic on a CIM backbone would be routine and that higher priority emergency traffic typically represents no more than 5% to 10% of total data traffic. This distribution of data transfer priority may be adjusted to suit any particular situation. A sensitivity analysis was performed to note effects of changing the distribution.

Table 3. Selected Distribution of Data Transfer Priorities for use in Simulation

Priority	% of Data Transfers
4	2%
3	8%
2	20%
1	30%
0	40%

The Time-to-Live (TTL) of the data to be transferred by an interface node was also determined randomly using a uniform distribution and a maximum TTL of 200 time slots. This represents a more realistic distribution than a simple Gaussian distribution, because each type of data transfer in a CIM implementation (real-time, transaction based, or batch) will have widely different critical perishability (TTL). However, for completeness Gaussian distributions for TTL were examined. Distributions were varied from means equal to 100 and standard deviations equal to 25 to means equal to 50 and standard deviations equal to 15. The results from these sensitivity runs showed that all data transfer shortfalls were easily satisfied using the reallocation approach that we have selected. The Gaussian distribution for TTL represented no challenge for our approach nor did it represent a realistic situation. As a result of these preliminary investigations, the uniform distribution with maximum TTL equal to 200 was used for simulation runs.

4.1.2 Description of Simulation

The simulation consists of two programs; a baseline program (BASELINE) that was run without reordering of the token-passing sequence, and the same program (REALTIME) with reordering of the token-passing sequence based upon the reallocation approach. For both programs, input data on the nodes to be included in the simulation was read from the same data file. The initial token-passing sequence for the nodes was determined as specified by the IEEE Standard 802.4 protocol, with the higher address nodes appearing further up in the token-passing sequence.

The program, BASELINE, was run through 25 token-passing sequences collecting statistics on the resulting data transfer resources shortfalls. REALTIME was also run through the same 25 token-passing sequences, however this program first reorders the token-passing sequences based upon our approach for the reallocation of real-time data transfer resources. Statistics were collected on the resulting data transfer resource shortfalls after the reordering.

The simulation of nodes was made extremely easy through the use of object-oriented programming. A subclass of objects named "Node" was created to model an interface node. Another benefit of using object-oriented programming was the ability to use the library of complex pre-defined groups of objects that Objective-C provided, as do most object-oriented programming languages. To model the communications order for the nodes (called the token-passing sequence), the nodes inherited the properties of an ordered collection, called OrdColl in Objective-C, to allow the correct sequencing and re-sequencing of nodes to take place. Because this research effort involved extensive use of dynamic token-passing sequences, this use of the library definition saved a tremendous amount of effort in simulating the token-passing protocol for the communications backbone.

4.1.3 Results of Simulation

A comparison of the statistics on the data transfer resource shortfalls as collected by the two programs is shown in Table 4. As shown in this table, our approach for reallocation of data transfer resources results in the system satisfying 99.1% of the shortfalls encountered during the 25 token-passing sequences. The shortfalls that could not be satisfied by the reallocation approach involve only the two lowest priorities of data transfers.

Table 4. Simulation Results: Comparison of Data Transfer Shortfalls by Priority

Priority[1]	Without Reallocation[2]	With Reallocation[3]
4	14	0
3	39	0
2	114	0
1	195	2
0	275	4
Total	637	6

1 Priority 4 is the highest priority, priority 0 is the lowest

2 Data from 25 simulation runs of program BASELINE

3 Data from 25 simulation runs of program REALTIME

A sensitivity analysis was performed by varying the distribution of priorities for a wide range of TTL distributions. The reallocation approach satisfied 100% of the top 30% to 35% highest priority data transfers in all cases, and satisfied 99% of the remaining 65% to 70% lower priority data transfers. This indicates that our approach will be able to satisfy all situations for real-time fault detection in a small to medium sized factory and will be able to handle peak emergency data traffic, as well.

4.2 Simulation of Technique for Dealing with Uncertainty

4.2.1 Assumptions Used in Simulation under Uncertain Information

The sample manufacturing cell shown in Figure 1 was simulated to demonstrate that our approach for handling uncertainty was applicable to general situations found on the factory floor. Four problem hypotheses, shown in Table 5 and the ranking of these hypotheses shown in Table 6 and the six sensors (with their resulting evidence instances) from Figure 1 of the sample cell were simulated. This represents a realistic configuration, although the simulation can be expanded to handle more than those numbers. As the number of sensors grows, the problem can be partitioned to handle the greater complexity.

Table 5. Possible Hypotheses given Evidence E_3 (No parts received at Station B) Description of Hypotheses

Hypothesis	Description
H_1	No Raw Material Inventory
H_2	Slippage in Conveyor Belt
H_3	Conveyor Motor is Seized
H_4	No Electrical Power to Conveyor Belt Motor

Table 6. Ranking of Hypotheses in Table 5 by Seriousness of Consequences

Hypothesis	$MB'(H_1,E_3)$	Seriousness (1 = Most)
H_3	0.10	1
H_2	0.30	2
H_1	0.40	3
H_4	0.25	4

4.2.2 Description of Simulation

The positive evidence instances for the cases used in this simulation (KBES) are read from an input data file. These positive evidence instances were then used to drive the procedures described in the implementation of the uncertainty technique. These procedures calculate Measures of Belief (MBs), Measures of Disbelief (MDs), and Certainty Factors (CFs) for the problem hypotheses. If the CF of a hypothesis is over the threshold of believability for the hypothesis, the correction procedure for that hy-

pothesis is called. If both the MB and MD are over the threshold indicating a conflict, the evidence instances leading to that conflict are identified, and troubleshooting procedures for the sensors leading to those evidence instances are called. The thresholds used in the procedures are adjustable in the program modules. The correction procedures and sensor troubleshooting procedures as implemented in this simulation consist of display messages indicating which procedures would be activated.

As part of the expert system implementation, several Implementation Tables were used containing parameters for testing hypotheses, among other things. Some of these tables were fairly simple and could be programmed very easily, others are quite complex and require considerable time and effort to program. The Implementation Table shown in Table 7 is an example of a complex table for which there was not a simple programming implementation other than object-oriented programming. There are four hypotheses (H_1 through H_4) listed in Table 7. The threshold value above which the hypothesis is considered true is listed for each hypothesis. Each hypothesis also has a an MB (Measure of Belief) column and an MD (Measure of Disbelief) column. In each MB and MD column a variable number of applicable evidence instances (E_i) are listed. Across from each of these evidence instances, the associated MB or MD value associated with that hypothesis is given for that evidence instance. The complexity of programming for this table arises from the interrelationships between the evidence instances and hypotheses and the sub-column divisions. The table was implemented by creating a subclass of "Object" called "Table_D". That class has a separate instance for each subsystem that utilizes the expert system. The characteristics of the class include threshold values, the applicable evidence instances, and MB and MD values. The programming of the Implementation Table was performed easily and very naturally using Objective-C. Standard techniques for programming the table without object-oriented techniques failed due to difficulties in writing all of the possible relationships in standard high level languages. Standard techniques proved cumbersome for the Implementation Table and hid the important characteristics and relationships within the table.

4.2.3 Results of Simulation

Seven cases representing the worst cases confronting this configuration were run using this simulation program. In all seven cases, the results were identical to those calculated manually for the same cases. All believable hypotheses were identified as believable and their associated correction procedures were called. All evidence instances leading to conflict were identified and their associated sensor troubleshooting procedures were called.

4.3 Combined Simulation for Reallocation of Real-Time Data Transfer

4.3.1 Assumptions Used in Simulation

All of the assumptions discussed earlier were presumed to hold true for the combined simulation. The correction procedures for each problem hypothesis assumes the values of TTL and priority presented in Table 8 remain as initially selected.

Table 8. Parameters used in Combined Simulation for Corrective Action for each Problem Hypothesis

Hypothesis	Corrective Action Required TTL	Corrective Action Required Priority
H_1 - No Inventory	40	2
H_2 - Belt Slippage	20	2
H_3 - Seized Motor	10	3
H_4 - No Power	50	2

4.3.2 Description of Simulation

This simulation (SIMULATION) used program modules from each of the two previous simulations (REALTIME and KBES). Input data on the 100 nodes was read in from a data file, and the initial token-passing sequence was determined as in the REALTIME program above. At this point, the approach for handling uncertainty was simulated for one of the nodes, Node 50 of the

Table 7. Implementation Table "D" for use in Methodology for Dealing with Uncertainty

*H_3 (0.25)				H_2 (0.40)				H_1 (0.55)				H_4 (0.75)			
MB		MD		MB		MD		MB		MD		MB		MD	
E_i	Value	$\sim E_i$	Value	E_i	Value	$\sim E_i$	Value	E_i	Value	$\sim E_i$	Value	E_i	Value	$\sim E_i$	Value
E_1	0.60	$\sim E_1$	1.00	E_2	0.50	$\sim E_2$	1.00	E_3	0.35	$\sim E_1$	0.60	E_2	0.25	$\sim E_1$	0.40
E_2	0.50	$\sim E_2$	1.00	E_3	0.30	$\sim E_3$	1.00	E_4	1.00	$\sim E_2$	0.60	E_3	0.25	$\sim E_2$	1.00
E_3	0.10	$\sim E_3$	1.00	E_5	0.80	$\sim E_5$	1.00			$\sim E_3$	1.00	E_4	1.00	$\sim E_3$	1.00
										$\sim E_4$	1.00			$\sim E_6$	1.00

NOTES: (1) This table is a static table, changing only as cell configuration changes
(2) The asterisk with a hypothesis denotes a serious consequence
(3) The numbers in parentheses are the threshold values for each hypothesis

100 nodes. Any one of the nodes could have been selected. The simulation could, in fact, be modified to implement KBES for all of the 100 nodes. However, this is unnecessary to demonstrate the validity of the approach. The output of KBES, as modified for this simulation, was used only to determine the TTL and Priority for the particular node for which KBES is implemented. The real-time reallocation approach simulation (REALTIME) operates on given TTL and Priority values regardless of how these values are determined. The new TTL and Priority values required for correction of the problems identified by the KBES program module are used in the combined simulation as the values for node 50. The token-passing sequence is then modified to alleviate data transfer resource shortfalls, as noted in the previous section.

Differentiating between those nodes that have the expert system implemented and those that do not, requires two subclasses of the class "Node" to be developed for later phases of the effort. Selected characteristics of the parent class "Node" will be inherited by each of the two subclasses, the intelligent subclass and the non-intelligent subclass. The selected characteristics may be different for each subclass, and additional characteristics may be specified for one subclass or the other. This ability is extremely powerful but very simple to implement, and the most important factor is that it will be completely compatible with the existing simulation.

4.3.3 Results of Simulation

The simulation was run for four different problem states in node 50, corresponding to the four problem hypotheses. In each case, the problem in the manufacturing cell served by node 50 was correctly identified by the KBES program module and the appropriate correction procedure was invoked. The required data transfer TTL and Priority requirements for the correction procedures were in each case satisfied by the real-time data transfer reallocation procedure.

5. SUMMARY

The two approaches developed for use in an intelligent interface node (approach for reallocation of real-time data transfer resources and approach for handling uncertainty) were simulated on an IBM-compatible personal computer. The approach for reallocation of real-time data transfer resources was shown to satisfy 99.1% of all data transfer resource shortfalls that would exist without the benefit of implementation of our approach. Most importantly, the highest priority data transfers were always satisfied. The technique for handling uncertainty was shown to correctly deduce problem situations in the sample manufacturing cell even in situations involving conflict. The concurrent simulation of the two approaches showed that problems in the manufacturing cell could be correctly deduced, and the resulting interrupt messages for the corrective action were able to be transmitted within the given time constraints. Thus, the results of the simulations demonstrated the efficiency of both techniques and the effective interaction of the two techniques.

REFERENCES

Allen, R. (1986), "Factory Communication: MAP Promises to Pull the Pieces Together," *Electronic Design, 34*, 11,102.

"An Architecture of CIM" (1988), CAM-I ATPC Report R-88-ATPC-01, Arlington, TX,

Day, J.D. and Zimmerman, H., (1983), "The OSI Reference Model," *Proceedings of the IEEE, 71*, 12, 1334.

Green, Lee. "Gearing Up for CIM, (1987)," *Information WEEK, 112,* 24.

Iyer, V. and Joshi, S., (1985), "FDDI's 100 M-bps Protocol Improves on 802.5 Spec's 4 M-bps Limit,"*Electronic Design News, 30,* 10, 151.

Moose, R.L., Jr., (1989), "Modeling Networks with Dynamic Topologies," *ORSA Journal of Computing 1*: 4 223-231.

Nance, R.E. and Moose, R.L., Jr., (1988), "Link Capacity Assignment in Dynamic Hierarchical Networks," *Computer Networks and ISDN Systems, 5*: 1, 189-202.

Palmer, J.D. (1987), "Information Backbone Characteristics in Contemporary and Future CIM Operations," *In Proceedings, SMC International Conference.,* Washington, D.C., 666-670

Palmer, J.D. and Liang Y., (1987), "Distributed Systems Architecture and Decision Support Systems in Computer Integrated Manufacturing," *In Proceedings, , IEEE Computer Society Conference on Computer and Applications,* Beijing, PRC, 537-543.

Pardue, M.D. (1987), "Fine-Tuning the OSI Model: Layer Functions and Services," *In MILCOM 87 Conference Record, 1,* 10.1.1.

Pardue, M.D. and Palmer, J.D., (1988), "Real-time Data Transfer in an Intelligent CIM Mode," *Proceedings IEEE International Conference on Intelligent Control,* Arlington, VA., 730-735.

Spracklen, C.T., and Smythe, C., (1987), "Direct Sequence Spread Spectrum Access to Local Area Networks," *In MILCOM 87 Conference Record, 1,* 10.6.1.

Wiegard, J., (1987), "LAN ICs for IEEE-802 Networks," Electronic Design News, 32, 9, 130.