[13] N. Kumagai, Y. Ishida and H. Tokumaru, "A Knowledge Representation for Diagnosis of Dynamical Systems," *IFAC Real-Time Programming*, Lake Balaton, Hungary, 1986, pp. 27-37.

[14] S. V. Nageswara Rao and N. Viswanadham, "Fault Diagnosis in Dynamical Systems: A Graph Theoretic Approach," *Int. J. Systems Sci.*, 1987, vol. 18, no. 4, pp. 687-695.

[15] N. H. Narayanan and N. Vishwanadham, "A Methodology for Knowledge Acquisition and Reasoning in Failure Analysis of Systems," *IEEE Trans. Syst., Man and Cybernetics*, vol. SMC-17, no. 2, Mar./Apr. 1987, pp. 274-288.

[16] Amit Misra et al., "Diagnosability of Dynamical Systems," The Third International Workshop on Principles of Diagnosis, Oct. 1992, Rosario, Washington.

[17] S. Padalkar et al., "Real-Time Fault Diagnostics," *IEEE Expert*, Vol. 6, No.3, pp. 75-85, June 1991.

[18] G. Karsai et al., "Model-based Intelligent Process Control for Cogenerator Plants," to appear in *Journal of Parallel and Distributed Computing*, vol. 15, 1992.

[19] Csaba Biegl, "Design and implementation of an execution environment for knowledge-based systems," PhD Thesis, Dept. of Electrical Engineering, Vanderbilt University, Nashville, TN (1988).

[20] Li, R, and J. H. Olson, "Fault Detection and Diagnosis in a Closed-loop Nonlinear Distillation Process. Application of extended Kalman filters," *Industrial Engineering Chemistry Research* v. 30 n 5, p. 898-908 (1991).

[21] Mah, R. S. H., G. M. Stanley, and D. M. Downing, "Reconciliation and Rectification of Process Flow and Inventory Data," *Ind. Eng. Chem. Process Des. Dev.*, 15, 175 (1976).

[22] Stanley, G. M., and R. S. H. Mah, "Estimation of Flows and Temperatures in Process Networks," *AIChE J.*, 23, 642 (1977).

[23] Watanabe, K., and D. M. Himmelblau, "Fault Diagnosis in Nonlinear Chemical Processes. I: Theory," *AIChE J.*, 29, 243 (1983a).

[24] Watanabe, K., "Incipient Fault Diagnosis of Nonlinear Processes with Multiple Causes of Faults," *Chem. Eng. Sci.*, 39, 491 (1984).

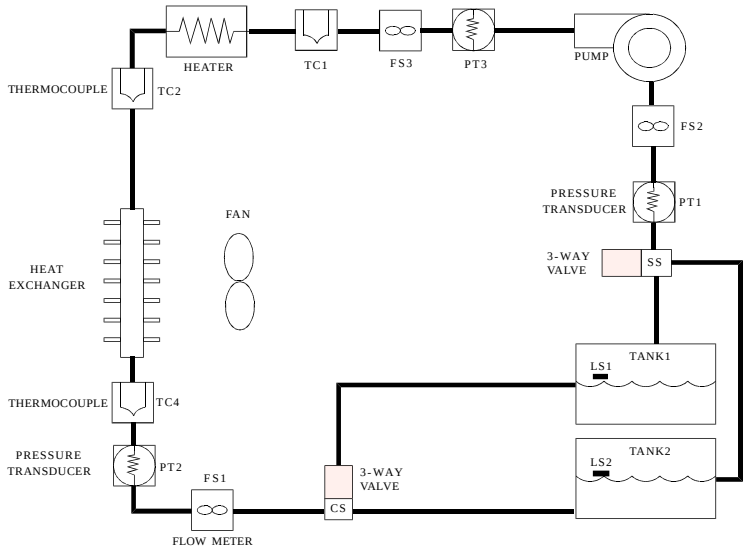

Figure 5: Schematic of Thermal Control System

References

[1] R. Patton, P. Frank and R. Clark, "Fault Diagnosis in Dynamic Systems: Theory and Application," Prentice Hall International (UK), 1989.

[2] R. N. Clark, "State Estimation Schemes for Instrument Fault Detection," Chapter 2 in the book *Fault Diagnosis in Dynamic Systems: Theory and Application*, Prentice Hall International (UK), 1989, pp. 21-45.

[3] F. Hayes-Roth et al., "Building Expert Systems," Addison-Wesley, Reading, Mass. 1983.

[4] E. Shortliffe, "Computer Based Medical Consultations: MYCIN," American Elseiver, New York, 1976.

[5] J. Brown et al., "SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting," BBN report 2970, Bolt Beranek and Newman, Cambridge, Mass., 1974.

[6] W. R. Nelson, "REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents," *Proc. AAAI*, 1982, pp. 296-301.

[7] Y. Peng and J. Reggia, "A Connectionist Model for Diagnostic Problem Solving," *IEEE Trans. Syst., Man and Cybernetics*, vol. SMC-19, no. 2, March/April 1989, pp. 285-298.

[8] T.-H. Guo and J. Nurre, "Sensor Failure Detection and Recovery by Neural Networks," in *Proc. Int. Joint. Conf. on Neural Networks*, July 1991, vol. I, pp. 221-226.

[9] M. S. Fox, S. Lowenfield and P. Klienosky, "Techniques for Sensor-Based Diagnosis," in *Proc. 8th Int. Joint. Conf. Artificial Intelligence*, 1983, pp. 158-163.

[10] B. Chandrasekaran and W. Punch III, "Data Validation During Diagnosis, A Step Beyond Traditional Sensor Validation," *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, July 1987.

[11] E. Scarl *et al.*, "Diagnosis and Sensor Validation through Knowledge of Structure and Function," *IEEE Trans. Syst., Man and Cybernetics*, vol. SMC-17, no. 3, May./June 1987, pp. 360-368.

[12] Y. Ishida, N. Adachi and H. Tokumaru, "A Topological Approach to Failure Diagnosis of Large-Scale Systems," *IEEE Trans. Syst., Man and Cybernetics*, vol. SMC-15, no. 3, May./June 1985, pp. 327-333.

Faults	Time in seconds
TS	0.390315
PS	0.687290
PS+TS	0.466055
TS+CS	0.489567
TS+CS+PS	0.484502

Table 2: Sensor faults (false alarms)

Faults	Time in seconds
PS1	1.223147
PS1+PS2	0.967154
PS1+PS2+PS3	0.842267
PS1+PS2+PS3+PS4	0.767575

Table 3: Sensor Faults (silent alarms)

from the pump such that the sensors did not indicate a drop in pressure when they were supposed to (missing alarm). The results are shown in Table 3. With the first three sensor failures, the diagnoser returned correct diagnostic results – failure of pump and failures in the three sensors which did not detect a drop in pressure. When the fourth sensor was failed, the diagnoser returns the following result – the pressure sensor and flow meter in the pump assembly which are reading low values are faulty, the pump is all right and none of the four pressure sensors mentioned above are faulty. This is a case when the diagnoser gets misled by sensor failures but it requires four sensor failures to cause the diagnoser to go astray. As long as there is sufficient evidence for diagnoser to work with, it returns the correct results. Note that as the number of missing alarms increases, the diagnosis time goes down. This is because the diagnosis is event driven and is triggered every time that an alarm rings. Thus with a smaller number of ringing alarms, the diagnosis gets triggered less number of time and takes less cumulative time for diagnosis. The second example is that of a Thermal Control System (TCS) whose schematic is shown in figure 5. This system has the simple task of maintaining the temperature of water within certain limits. There are two tanks in the system, only one of which is on-line at a time. The water is pumped through a heater and then a heat exchanger. By controlling the amount of heating in the heater and cooling in the heat exchanger, the temperature of the water is maintained. The three way valves before and after the tanks are used to put a tank on-line or off-line.

We built the models for TCS and conducted tests with the actual hardware. The run-time system consisted of the diagnostic system running on a SPARC 1 workstation, talking to a data acquisition unit over a serial line. The data acquisition unit collected the signals provided by the sensors in the system and sent them over to the diagnostic system. The alarm generator part of the diagnostic system read these signals and rang and/or shut off alarm(s). The diagnoser then updated the fault status of the components in the system and displayed the results. We tested leaks in the tanks, fan failure, pump failure etc. and an assortment of sensor failures. The diagnoser returned correct results in case of component failures. Sensor failures were also diagnosed properly. In fact, even when all the three pressure transducers were failed, the diagnoser deduced the sensor failures correctly since none of the flow meters indicated any abnormal values.

8. CONCLUSION

In this paper we have presented a model-based diagnostic system which diagnoses the faults in the system correctly if the number of observation errors is not too large. Furthermore, the diagnostic algorithms used in the system are of polynomial complexity and hence the system can deliver real-time performance. The use of explicit fault models cuts down the search space and the number of hypotheses that need to be generated and validated. The use of spatial and temporal constraints (structural redundancy) in the model makes the detection of inconsistent observations computationally feasible and hence we can hypothesize about sensor failures without having to consider every possible combination of sensor and component failure(s).

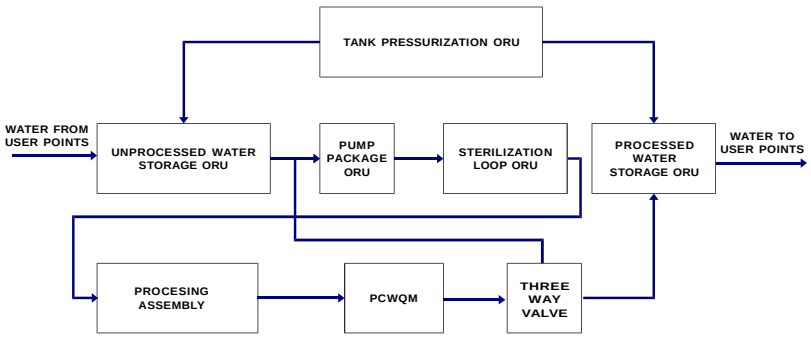

Figure 4: Schematic of Hygiene Water Processor

Faults	Time in seconds
TNK	0.450163
HTR	0.874088
PMP	1.004877
TNK+HTR	1.451494
TNK+PMP	1.365014
HTR+PMP	1.249900
TNK+HTR+PMP	1.572464
HTR+PMP+WQM	1.652971
TNK+HTR+PMP+DST	2.148210
TNK+HTR+PMP+WQM	2.106865

Table 1: Single and multiple component faults

of sterilization, bacteria removal and some other processing. The PCWQM senses the amount of contaminants in the water and controls the 3 Way Switch to either send it back to the purification circuit or to the Processed Water Storage tank. We built the models for the HWP and tested the diagnostic system using these models. For testing purposes, we've developed a Fault Pattern Simulator (FPS). One can specify the components and/or sensors to fail and FPS generates the alarm pattern in real-time, using the failure propagation information in the models.

The results from simulations using HWP models are shown in Tables 1, 2 and 3. The abbreviations used are: (1) `TNK` is the tank in the Unprocessed Water Storage that supplies the water to the purification circuit, (2) `HTR` is the heater in the sterilization assembly, (3) `PMP` is the pump assembly, (4) `WQM` is the water quality monitor (PCWQM), (5) `DST` is the distribution assembly which supplies the clean water to user points, (6) `TS` is the temperature sensor in the heater, (7) `PS` is the pressure sensor in the pump assembly, (8) `CS` is the pressure sensor in the pressurization assembly, (9) `PS1` is the pressure sensor after the 3-way valve, (10) `PS2` is the pressure sensor after PCWQM, (11) `PS3` is the pressure sensor after the UF/RO filter assembly, and (12) `PS4` is the pressure sensor in the Sterilization assembly. The fault column shows the fault(s) that were simulated. Multiple faults are indicated by `+`, e.g., `TNK+HTR` means that a leak in the tank and a fault in the heater (fail high) were the two faults simulated.

The time column shows the time spent on diagnosing the faults. Note that the diagnosis system consists of an ILC and a number of instances of functionality diagnosers (FD), as discussed before. Also, the diagnostic system is implemented using the Multigraph model of parallel, distributed computation.[19] The ILC and all the FDs are implemented as actors which exchange information, commands and results by using message passing. Thus, time shown in this column includes the time spent in ILC, in the various FDs and in the message passing.

Table 1 shows the results of single and multiple fault scenario simulations. Table 2 shows the results with single and combinations of sensor failures. The sensor failures in this experiment gave rise to false alarms. They were all diagnosed as sensor failures since the evidence against an actual fault and for a false alarm was quite strong. In the next experiment, we failed the component pump and also failed one, two, three and four pressure sensors downstream

non-ringing alarms and checking the propagation times from each ringing alarm to these non-ringing alarms to find the earliest expected alarm time.

6.4. Complexity analysis

There are m $fNode$s and n $dNode$s. The total number of nodes in the graph is $(m+n)$. Some of the above algorithms use DFS. The order of DFS in our case is $O((m+n)^2)$ since the graph is represented using adjacency matrix. The alarm sets P, S etc. are maintained as arrays and the operation of adding or deleting the alarms is $O(n)$ since the maximum number of alarms is n. Many of the loops iterate on the number of hypotheses n_h. Theoretically, the number of hypotheses can be $O(mn)$, however, for any reasonable model and alarm pattern, the number of hypotheses will be $O(m)$.

Updating Hypotheses : The first step in the algorithm involves, given an alarm, checking if an $fNode$ is ancestor of the alarm. If it is an ancestor, we then check if the alarm is a primary alarm of the $fNode$. This takes $O(1)$ time since the information is maintained in matrices. Next, if the alarm *is* a primary alarm of the $fNode$, we check whether some other alarms, which were marked as primary, have become secondary alarms of the $fNode$ because of the new alarm. This takes $O(n)$ time since there are at most n alarms. Since we do the above for every $fNode$ and there are m $fNode$s, this takes $O(mn)$ time.

The next step, recomputing the estimated time of occurrence of failure modes, involves going through the list of all the hypotheses that an $fNode$ may have. Since an $fNode$ can be incident upon at most n $dNode$s, there can be at most n such hypotheses. For each hypothesis, the new alarm is added to either P or SP set. Thus this step takes $O(n^2)$ time for each failure modes giving $O(mn^2)$ time overall.

Next we search through the graph (using DFS) and mark the alarms for inclusion into P, SP etc. for a hypothesis, which takes $O(n^2)$ time. This is repeated for each hypothesis, hence this step takes $O(n_h.n^2)$. The overall complexity for updating hypothesis then is $O(n_h.n^2)$ since n_h can theoretically be greater than m.

Finding Missing Alarms : Since the DFS is performed for all the alarms and all the hypotheses, the complexity of this step is $O(n_h.n.(m+n)^2)$.

Locating the fault sources : The sorting the hypotheses according to their rank takes $O(n_h\ log n_h)$ time. Next we examine the hypotheses until all the alarms are explained. The number of hypotheses examined is always less than or equal to the number of ringing alarms. This is because each hypothesis that we look at will explain at least one alarm. If, however, the next hypothesis does not explain any alarm, i.e., its rank is less than 1, we will not examine any more hypotheses, instead we will label the remaining unexplained alarms as false alarms. And since the number of ringing alarms can not be more than the number of alarms, we will examine at most n hypotheses. Examining each hypothesis involves going through its alarm sets, which is $O(n)$. Thus, examining the hypotheses takes $O(n^2)$ time. Thus, locating the fault sources takes $O(n_h log n_h\ +\ n^2)$.

Determining the next alarm : The complexity of this algorithm is $O(n^2)$ since there are at most n alarms and for each ringing alarm we examine, in the worst case, all the other alarms exactly once.

From the complexity of the steps described above, the complexity of the whole algorithm is $O(n_h log n_h + n^2 + n_h.n.(m+n)^2+n^2)$. In the worst case, with $n_h = m.n$, the complexity reduces to $O(m.n^4)$. In the more common case, n_h will be $O(m)$ and then the complexity reduces to $O(m.n^3)$. Since the step for finding missing alarms dominates the other two in terms of complexity, the time taken when the other two events, alarm becoming silent and timeout, come in, is still $O(m.n^3)$.

7. TESTS AND DISCUSSION

In this section some examples of multiple faults, sensor faults and combinations thereof are given and the robustness of the diagnostic system is examined. At first we'll discuss the results obtained by testing using simulation of faults. Then the test with an actual system is described.

The first example is in the context of the models for Hygiene Water Processor for the Space Station Freedom. Figure 4 shows a schematic of the HWP system. The tanks in the Unprocessed Water Storage (1) collect the water from user points, (2) store it, and (3) supply it to the purifying circuit. The tanks in the Processed Water Storage (1) collect water after purification, (2) store it and (3) supply it to user points. The purification circuit itself consists

1. a_i might not have any ancestor alarm that is ringing (it is a primary alarm).
2. a_i might have one or more ancestor ringing alarm(s) (it is a secondry alarm)
3. a_i might or might not be a descendant of one or more hypotheses already in the hypotheses table (if it is a descendant of an already existing hypothesis we add it to one of the alarm sets).
4. a_i might or might not be a descendant of one or more hypotheses that are not already in the table (if it is not a descendant of an already existing hypothesis we add a new hypothesis).
5. a_i might be an ancestor of some other ringing or silent alarms (in which case these alarms may need to be moved to S, SS or MS sets).

If there is a hypothesis h for which a_i is a primary alarm, we have to recompute the estimated time of occurrence of $f(h)$ and check if this new alarm is a spurious primary or a normal primary. Note that one f may have more than one hypotheses in the table. This will happen if two or more primary alarms of the failure mode conflict with each other temporally. In such cases we will divide the primary alarms into subgroups such that the alarms in one group all agree with each other temporally. Then we will have as many number of entries in the hypotheses table for this failure mode as there are groups. These entries will have the corresponding primary alarms in set P and all other primary alarms in set SP. For example, let there be a failure mode f_1, with primary alarms a_1, a_2, and a_3. Out of these primary alarms, let a_1 and a_2 agree with each other temporally, but not with a_3. In this case we will have two entries in the hypotheses table, h_1 and h_2, with

$$f(h_1) = f(h_2) = f_1, \quad P(h_1) = \{a_1, a_2\}, \quad SP(h_1) = \{a_3\}, \quad P(h_2) = \{a_3\}, \quad SP(h_2) = \{a_1, a_2\}$$

and the secondary alarms will be descendants of the corresponding primary alarms. Thus, we check to see if a_i is consistent with h or not and adds a_i to $P(h)$ or $SP(h)$. If it can not find any hypothesis for $f(h)$ which is consistent with a_i, it adds a new hypothesis for $f(h)$.

6.3.3. Finding the missing alarms

For each hypothesis h, there might be some alarms that should be ringing, given the estimate of time of occurrence of $f(h)$ and the present time, but are not ringing. The (primary or secondary) *missing* alarms of a hypothesis h are identified by performing a depth first search (DFS) of TFPG starting from $f(h)$ $\forall\, h \in H$.

6.3.4. Locating the fault sources

Next we identify the faulty components and sensors by trying to explain the alarm pattern in terms of faults in components and/or sensors. Now, we can try to take combinations of various faults that explain the alarm pattern and choose the combination that gives us the minimum number of components+sensors faults. But this will result in exponential number of combinations. So, we use the following heuristic : examine hypotheses in decreasing order of their rank stop once we reach a hypothesis whose rank is less than a particular threshold.

When we examine a hypothesis, we will accept the hypothesis only if it consistently explains a previously unexplained alarm. If we accept a hypothesis, it will explain the alarms that are in $P(h)$ and $S(h)$. But then we have to also believe that the missing alarm(s) in $MP(h)$ and $MS(h)$ are a result of failure of the sensor(s) associated with these missing alarms. Also, since we are using threshold for degree of confidence, there might be some ringing alarms that are not explained by any hypothesis whose degree of confidence is higher than the threshold. These unexplained ringing alarms will be treated as false alarms and as a consequence of sensor failure.

The rank of a hypothesis is given by the number of alarms consistent with the hypothesis minus the number of inconsistent alarms. Thus, the higher the rank of a hypothesis, the more confidence we have in the hypothesis. Also, $rank(h)$ will be 1 or greater only if there are more alarms supporting it than opposing it. Thus we choose $rank = 0$ as our threshold for examining and/or accepting a hypothesis.

6.3.5. Determining the next alarm

When there are some alarms ringing, we need to predict the next alarm that should ring and the latest time by which it should ring. This will help us in determining the silent and false alarms. This is done by examining all the

of failure mode f. The *rank* of a hypothesis is a number which gives an the measure of plausibility in the hypothesis. The higher the *rank*, the more likely we are to believe in the hypothesis. P, MP, SP, S, MS and SS are all sets of alarms. When a hypothesis is first created, its rank is set to 0. Whenever an alarm is added to the P or the S set of the hypothesis, its rank is incremented by one. Whenever an alarm is added to any of the other sets, the rank is decremented by one. (Remark : This way of computing the measure of plausibility has been choosen because it works well with the fault models used. However, it can be changed if required by the system characteristics. For example, we can include the *a priori* probabilities of component failures, or the probabilities of sensor failures in the measure.)

Notation : $f(h_i)$ means the failure mode of hypothesis h_i. $rank(h_i)$ means the *rank* of hypothesis h_i. $time(h)$ means the estimated time of occurrence of f. $P(h_i), SP(h_i)$ etc. will mean the corresponding alarm sets of hypothesis h_i. We will say that an alarm a_i *supports* hypothesis h if a_i is a descendant of $f(h)$ and the time of ringing of a_i and $time(h)$ satisfy temporal constraints.

The P, MP, SP, S, MS and SS are defined as follows

- P is the set of *primary alarms* which are reachable from $f(h_i)$ and also provide support for h_i. Note that there may be some primary alarms that $f(h_i)$ reaches but does not get any support for. Such cases are discussed later.
- MP is the set of *missing primary alarms* for h_i. Missing primary alarms are those primary alarms of $f(h_i)$ that are silent but should have been ringing, given the estimated time of occurrence of the failure mode.
- SP is the set of *spurious primary alarms* for h_i. These are the primary alarms that are either not reachable from $f(h_i)$ or, if reachable, do not support h_i.
- S is the set of *secondary alarms* for h_i that are reachable from, and are temporally consistent with, the alarms in set P.
- MS is the set of *missing secondary alarms* for h_i. These are the secondary alarms that should have been ringing, given the time stamp of alarms in set P.
- SS is the set of *spurious secondary alarms*. These are the secondary alarms that are either not reachable from h_i or, if reachable, are not temporally consistent with the alarms in set P.

6.3. Diagnostic reasoning

Whenever an event comes in, the hypotheses table will be updated and the fault source components and/or the faulty sensors are located. This involves four steps

1. Updating the hypotheses.
2. Finding the missing alarms.
3. Locating the fault sources.
4. Determining the next alarm to ring.

These steps are described in the next subsections. These four steps are explained for an alarm ringing event, i.e., when a new alarm starts ringing. For the other events, i.e., alarm becoming silent and timeout, only the last three of the above steps are performed.

6.3.1. Updating the hypotheses

When an event comes in (i.e., when an alarm starts ringing), we have to update the hypotheses in the table (recompute the estimated time of occurrence of the f represented by the hypotheses, add the alarm to one of the alarm sets described above and perhaps move other alarms from one set to another), maybe add new hypotheses and compute the rank of the hypotheses again. For this, we might make a few observations about the new alarm a_i :

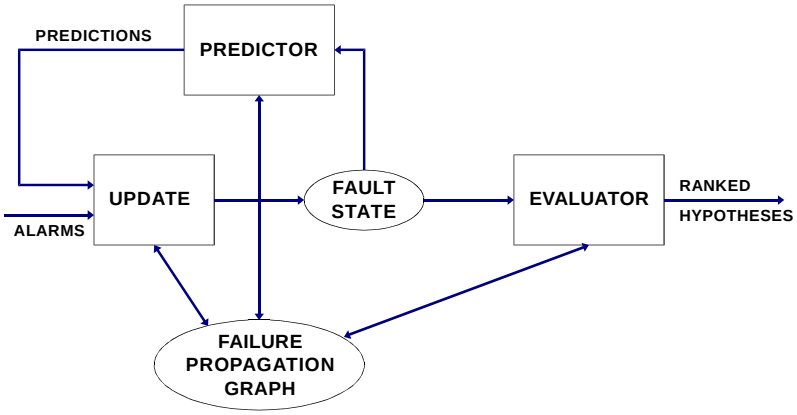

Figure 3: Conceptual structure of the robust reasoning method

failure propagation, it can be used to predict future failures based on the current fault state. The function of the Predictor is to derive deadlines for the future incoming alarms. The event comparator compares the incoming alarms and the predicted deadlines, and drives the Update module. The Update module processes the received event (alarm or passed deadline) and creates a new update of the fault state. The output is a ranked list of hypotheses which is generated by the Evaluator module. Ranking occurs according to the plausibility of the individual hypotheses. The Results module then selects the hypotheses that explain the current observations, identifies the fault source functionality and sends the results to ILC.

6. DIAGNOSTIC ALGORITHMS AND DATA STRUCTURES

6.1. Data structures related to TFPG

The TFPG of a system is a directed graph in which the nodes may represent either a failure mode ($fNode$) or a discrepancy ($dNode$). Let the set of failure modes in a TFPG be denoted by F and the set of discrepancies be denoted by D. Let there be m failure modes $f_1, ..., f_m$ and n discrepancies $d_1, ..., d_n$. Each edge $e_{i,j}$ in the graph is weighted with T_{min} and T_{max} as defined previously.

A $dNode$ may have an alarm associated with it. The occurrence of the discrepancy represented by node i is indicated by ringing of an alarm notated as a_i. The maximum number of alarms in a graph is n. An alarm may have a sensor associated with it. Sensor fault detection proceeds by identifying the wrong alarm(s) (false or missing) and locating the sensor associated with the alarm(s). In the following discussion, an $fNode$ may be referred to as f_i, a $dNode$ as d_i or a_i. We will use a $(m+n) \times (m+n)$ matrix, $Reach$, to represent the reachability in TFPG. $Reach_{i,j} = 1$ if there exists at least one path from node i to node j else it is 0. Note that, since TFPG is a directed graph, $Reach_{i,j} = 1$ does not imply $Reach_{j,i} = 1$

An alarm a_i is an ancestor of alarm a_j if $Reach_{i,j} = 1$ and it is a descendant of a_j if $Reach_{j,i} = 1$. Note that it is possible for a_i to be both ancestor and descendant of a_j (in case of loops). Also, we say that a_i and a_j are consistent with each other if $Reach_{i,j} = 1$ or $Reach_{j,i} = 1$ and the times when the alarms ring satisfy the temporal constraints.

6.2. Structure of hypotheses table

A hypothesis h represents exactly one failure mode of one component in the TFPG under consideration. A component may have more than one failure modes which may have occurred. In that case, each of the failure modes will have a distinct hypothesis representing it in the hypotheses table. It is also possible that one failure mode may have more than one entry in the hypotheses table, as is explained later.

The hypotheses table is a set H of hypotheses $\{h_i\}$. Each hypothesis is an n-tuple

$$h_i = < f, time, rank, P, MP, SP, S, MS, SS >$$

where f is the failure mode of a component for which this entry stands and $time$ is the estimated time of occurrence

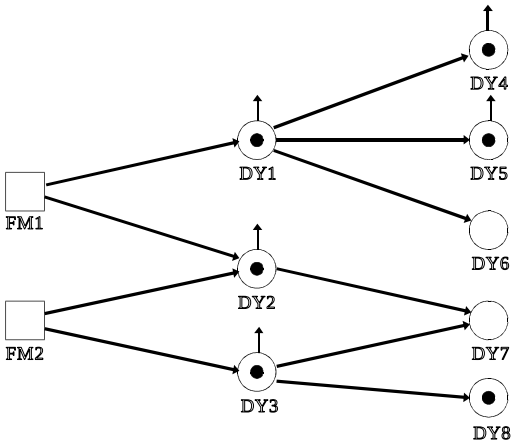

Figure 2: Use of structural redundancy for sensor fault detection

the alarm. The alarms associated with $DY1$, $DY2$, $DY3$, $DY4$ and $DY5$ are ringing (indicated by up arrow), while the alarm assigned to $DY8$ is silent. The simplest explanation for the alarms at $DY1$, $DY2$, $DY4$ and $DY5$ is that $FM1$ is a fault source (parsimony). Possible explanation for alarm $DY3$ is that $FM2$ is also a fault source. However, if $FM2$ is a fault source, the alarm at $DY8$ should also ring (structural redundancy), therefore this hypothesis implies that the sensor at $DY8$ must be faulty as well. An alternative explanation is that the sensor associated with $DY3$ is faulty and is giving rise to a spurious alarm. This hypothesis is more plausible than the previous one, since it explains the alarm scenario with 2 fault sources ($FM1$ and the sensor associated with $DY3$) instead of 3 ($FM1$, $FM2$ and the sensor associated with $DY8$) (parsimony). A number of other explanations can be found for the alarm pattern, that are all less plausible than the previous one.

Clearly, structural redundancy means the use of the interdependence among the alarms in the reasoning algorithm. The actual reasoning method is considerably more complex than the illustrative example due to the temporal aspect of the TFPG.

5. THE REASONING METHOD

Since a system is modeled hierarchically and since each functionality in the system has its own TFPG, the TFPGs are also linked up hierarchically. Thus the diagnostic algorithm has to 1) diagnose faults in the context of individual functionalities and 2) combine the results from different functionalities and guide the search through the hierarchy.

The diagnostic reasoning is divided into two modules – Functionality Diagnoser (FD) and Inter-Level Coordinator (ILC). The ILC receives the external events, provides an interface to the user and controls the diagnostic search. The FD operates on the TFPGs of individual functionalities and generates *local* results. These local results and some control information is passed on to the ILC. The ILC looks at the local results and decides which functionality to diagnose next, registers timeout requests and coordinates the diagnostic search.

5.1. Functionality diagnoser

The Functionality Diagnoser is triggered by one of these events – (1) an alarm ringing, (2) an alarm becoming silent after ringing for a while, or (3) a timeout occurring, i.e., a predicted alarm did not ring. The ILC receives the events from the external world and passes them on to the FD. It also tells FD which functionality to operate on. The FD locates the failure mode(s) and/or sensor faults that have occurred. It also identifies next alarm that should ring and then passes on all this information to the ILC.

The basic structure of the reasoning algorithm of FD applies the *predictor–corrector principle* well known in recursive estimation, system identification algorithms, such as Kalman filtering or observers. The conceptual block diagram of the reasoning method is shown in Figure 3.

The reasoning algorithm uses two basic data structures, the *fault state* and the TFPG. The fault state includes the actual list of fault hypotheses and the corresponding evidences. Because the TFPG expresses the dynamics of

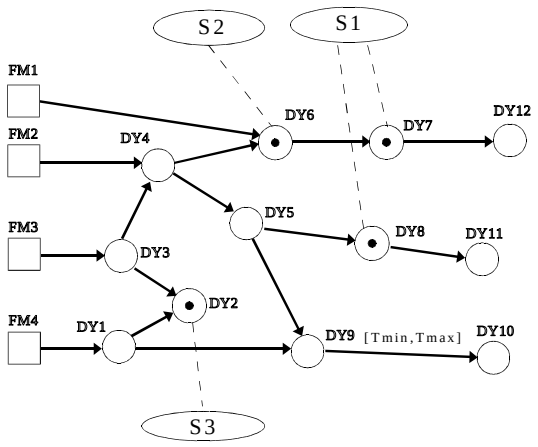

Figure 1: Timed Failure Propagation Graph

mapping between sensors and the alarms they generate.

The edges in the graph represent the propagations of failures and capture the interactions between different failures. Thus, an edge between two nodes means that the failure represented by the source node will propagate and cause the failure represented by the destination node. Each edge is labeled with a time interval $[T_{min}, T_{max}]$, which gives the minimum and maximum time that the source failure can take to propagate to the destination.

4. DIAGNOSTIC STRATEGY

The diagnostic system operates on the models discussed in the previous section and locates faults. The input to the diagnostic system are events that arrive asynchronously and are interpreted in the context of the fault models. The system (1) receives events, (2) generates hypotheses, and (3) selects some hypothesis(es) according to how consistently they explain the observations. For this it uses the principles of *parsimony* and *structural redundancy*.

4.1. Parsimony

A particular hypothesis is said to be consistent with the received events (observations) if the spatial and temporal constraints imposed by the propagation models are satisfied. If observation errors are possible, not all of the received events have to comply with the spatial and temporal constraints. Consequently, the number of hypotheses that are plausible under the given set of observed events will become larger.

The principle of parsimony suggests that *the simplest explanation is the best.* If a hypothesis can explain consistently all of the observed events, it should be considered more plausible than another one, which additionally requires the assumption of a sensor fault as well. Application of the principle of parsimony means that the different hypotheses are ranked according to some measure of their plausibility.

4.2. Structural redundancy

As we have discussed before, the physical interactions in dynamic systems impose spatial and temporal constraints on the observed events. This means that a fault in one part of the system may *propagate* to another part after some time. The timed failure propagation model captures these interrelationships and provides an explicit representation of the propagation paths. Consequently, in those parts of the system where failure propagation occurs, a single fault cause results in multiple manifestations. Obviously, these manifestations are not independent of each other, they provide a *redundant* observation of the fault cause. Because the failure propagation models primarily show structural relationships in the systems, we call this redundancy *structural redundancy.*

Due to the structural redundancy, events can confirm or contradict other events in a propagation model, therefore a concept similar to that of the analytical redundancy approach can be developed. The idea is illustrated in a simplified TFPG shown in Figure 2. The graph includes only failure propagations. For the sake of simplicity, all of the monitored discrepancies are associated with a unique sensor, whose output signal is used by a monitoring algorithms to generate

2. PREVIOUS WORK

The problem of observation errors in complex automated systems has been extensively investigated. There has been considerable research, particularly in the field of chemical process engineering, to use quantitative methods to detect and diagnose faults in a system. The two most common techniques used for fault diagnosis are filtering and parameter estimation.[20–24] Some quantitative methods have been developed to handle sensor failures as well, using the set of constraints among the observable physical variables in the system. The basic idea in this approach is the use of *analytical redundancy*[1] for detecting sensor failures. If the model of the plant (or a part of the plant) is known, than by observing any of the observable physical variables, the actual state of the plant can be calculated, which in turn can be used to derive the expected value of the other, related physical variables. Then, by comparing the computed values to the actual values read by the sensors, the health of a sensor can be determined.[2] The main drawback of these methods is that they are useful only for smaller, stable subsystems.

Researchers have addressed the problem of sensor failures using the association-based (rule-based or knowledge-based expert systems) techniques. Fox et al.[9] describe a sensor-based system, called PDS, in which the level of confidence in a sensor is by performing a "meta-diagnosis" of sensors before the actual diagnosis. But this results in just an additional rule base for sensor failures. Another association-based diagnostic system is described by Chandrasekaran et al.[10] The system diagnoses faults by pattern matching on failure manifestations to determine the likelihood of the faults. For handling sensor failures, the system uses pattern matching on *diagnostic expectations*, i.e., the expected values of sensors, given a fault. Thus it has to store a large number of patterns that will correspond to the different component fault(s) and sensor failure(s) combinations. This has been one of the biggest stumbling blocks for association-based diagnostic systems. Using "pre-compiled" knowledge about the failures in the system and their symptoms makes diagnosis of observation errors computationally prohibitive.

Model-based systems offer a way out of the problem of having to pre-compile and store such large number of associations. The presence of the model makes the detection of inconsistent observations feasible, hence, conceivably, observation errors can be diagnosed by using a suitable algorithm.

There are primarily two kinds of models that have traditionally been used for diagnosis – *functional* models and *fault* models. Functional models describe the "correct" behavior of the system, i.e., how the system is supposed to behave when no faults are present. Diagnostic algorithms using functional models, such as the one developed by Scarl,[11] suffer from two basic problems (1) function inverses may not be available (the mapping from inputs to outputs may not be invertible), and (2) the mapping between inputs and outputs may be too complex to uniquely determine faults.

Fault models, on the other hand, describe system behavior in the presence of faults. Fault models have been used for diagnosis in work done by many researchers.[12–15] Padalkar et al.[17] have further developed the work presented in[15] and integrated the diagnosis with a process control system .[18] The research presented in this paper uses similar fault models.

3. FAULT MODELS

In our previous work on the diagnosability of large-scale, dynamical systems, we introduced a hierarchical multiple-aspect modeling paradigm,[18] which captures the system dynamics in terms of a *timed fault propagation model* (TFPG).[16] In this section the diagnosis related concepts are briefly summarized.

The faults in a system and their interactions are modeled by using a labeled digraph to represent the dynamics of the system under fault conditions. For the sake of brevity, we will not present here the definitions and concepts involved in TFPG; instead we illustrate the concepts with the example shown in Figure 1. The square boxes in the figure represent the *failure modes*, e.g., `valve stuck open`, of *components* in the system. The circles represent the anomalies in the system behavior, called *discrepancies*, e.g., `loss of flow`. The dotted circles represent those discrepancies that have *alarms* associated with them. These are called *monitored* discrepancies, which means that the occurrence of these discrepancies can be observed and will be indicated by "ringing" of the associated alarm. The empty circles represent discrepancies that do not have alarms associated with them and are called *non-monitored* discrepancies. The ellipses represent the *sensors* in the system. Sensors measure the values of physical variables. The signals provided by these sensors are used to decide if a discrepancy exists and to generate (ring) the associated alarm. The dotted lines between the sensors and monitored discrepancies represent the (possibly many-to-many)

Robust diagnostic system: structural redundancy approach

Amit Misra and Janos Sztipanovits

Department of Electrical Engineering
Vanderbilt University
Nashville, TN 37235

James Ray Carnes

Advanced Computing Group
Boeing Missiles & Space Division
PO Box 240002, M/S JN-55
Huntsville, AL 35824

ABSTRACT

The complexity of diagnosis tasks in large-scale, heterogeneous dynamic systems has turned the attention in this area toward model-based methods. Most model-based diagnostic systems try to find a set of failed components that can explain observed discrepancies in system behavior consistently with constraints represented by the models. Observations are usually sensor-based and discrepancies are detected by fault detection algorithms that compare expected and observed system behavior. We have developed and field tested a real-time robust diagnostic system, which uses hierarchical, multiple-aspect models of plants. The models include the functional structure, timed failure propagation graphs, physical component structure and component failure modes. The diagnostic reasoning applies structural and temporal constraints for the generation and validation of fault hypotheses using the "predictor-corrector" principle. The diagnosis is generated in real-time amid an evolving alarm scenario, and uses progressive deepening control strategy. The robust diagnostic system has been tested and demonstrated using ECLSS models obtained from the Boeing Company.

1. INTRODUCTION

The continued operation and safety of a large scale heterogeneous dynamical system depends upon speedy and correct diagnosis of faults. In these systems, the sensory input signals provide information to diagnostic programs. When the information provided by the signals indicates incorrect system behavior, the diagnostic program should explain the incorrect behavior by identifying the component(s) in the system that have failed. In other words, the diagnostic program explains the *observed* system behavior in terms of a hypothesis about the state of the components in the system.

Since the diagnoser has to locate the fault(s) by interpreting the observations (information provided by sensor signals), the correctness and completeness of diagnostic results depend upon the reliability of observations. By correctness of the results we mean that only those components that are actually faulty are identified as faulty and no healthy component is part of the diagnostic result. Completeness of results means that *all* the components that are faulty are indicted. In the real world, the observations can be erroneous because (1) sensors can fail and thus provide the diagnostic system with wrong data values, (2) even if the sensors work correctly, the signals read by them could be interpreted wrongly (in model-based systems, this would be a modeling error), and, (3) in case of a major fault, the assumptions about the system can become invalid (in model-based systems this means a loss of model validity). Such erroneous observations can lead the diagnoser astray. Thus, the results returned by the diagnoser may be incomplete and/or incorrect. This leads to a need for a diagnostic system that can handle observation errors.

This paper describes a robust diagnostic reasoning method that is able to diagnose faults properly in the presence of observation errors and degrades gracefully as the number of observation errors increases. The diagnoser uses the structural and temporal constraints imposed by system dynamics on the fault propagations to reason about the faults. The computational complexity of the reasoning method is polynomial, hence it can generate the diagnosis in real-time.