Model-Based Diagnosis for Information Survivability [1]

Howard Shrobe
NE43-839
Artificial Intelligence Laboratory
Massacusetts Institute of Technology
Cambridge, MA 02139
hes@ai.mit.edu

Abstract.

The Infrastructure of modern society is controlled by software systems that are vulnerable to attack. Successful attacks on these systems can lead to catastrophic results; the survivability of such information systems in the face of attacks is therefore an area of extreme importance to society. This paper presents model-based techniques for the diagnosis of potentially compromised software systems; these techniques can be used to aid the self-diagnosis and recovery from failure of critical software systems. It introduces *Information Survivability* as a new domain of application for model-baesed diagnosis and it presents new modeling and reasoning techniques relevant to the domain. In particular: 1) We develop techniques for the diagnosis of compromised *software* systems (previous work on model-base diagnosis has been primarily cconcerned with physical components); 2) We develop methods for dealing with model-based diagnosis as a mixture of symbolic and Bayesian inference; 3) We develop techniques for dealing with common-mode failures; 4) We develop unified representational techniques for reasoning about information attacks, the vulnerabilities and compromises of computational resources, and the observed behavior of computations; 5) We highlght additional information that should be part of the goal of model-based diagnosis.

1 Background and Motivation

The infrastructure of modern society is controlled by computational systems that are vulnerabile to information attacks. The system and application software of these systems possess vulnerabilities that enable attacks capable of compromising the resources used by the software systems. A skillful attack could lead to consequences as dire as those of modern warfare. In every exercise conducted by the government so far, the attacking team has managed to completely the target systems with little difficulty. There is a dire need for new approaches to protect the computational infrastructure from such attacks and to enable it to continue functioning even when attacks have been successfully launched.

Our presmise is that to protect the infrastructure we need to restructure these software systems as *Adaptive Survivable Systems*. In particular, we believe that a software system must be capable of detecting its own malfunction and it must be capable of repairing itself. But this means that it must first be able to *diagnose* the form of the failure; in particular, it must both localize and characterize the breakdown.

Our work is set in the difficult context in which there is a concerted and coordinated attack by a determined adversary. This context places an extra burden on the diagnostic component. It is no longer adequate merely to determine which component of a computation has failed to achieve its goal, in addition we wish to determine whether that failure is indicative of a *compromise* to the underlying infrastructure and whether that compromise is likely to lead to failures of other computations at other times. Furthermore, we wish to determine what kind of *attack* compromised the resource and whether this attack is likely to have compromised other resources that share a vulnerability. This paper focuses on the diagnostic component of self adaptive survivable systems.

2 Contributions of this Work

We build on previous work in Model-Based diagnosis [2, 3, 4, 5, 8]. However, the context of our research is significantly different from that of the prior research, leading us to confront several important issues that have not previously been addressed. In particular, we present several new advances in representation and reasoning techniques for model-based diagnosis:

1. We develop representation and reasoning techniques for describing and reasoning about the behaviors and failures of *software systems* (most previous work has focussed on hardware, particularly digital hardware).
2. We develop mixed symbolic and Bayesian reasoning technique for model-based diagnosis. The statistical component of the technique utilizes Bayesian networks to calculate accurate posterior probabilities.
3. We develop a unified framework for reasoning about the failures of the computations, about how these failures are related to compromises of the underlying resources, about the vulnerabilities of these resources and how these vulnerabilities enable attacks.
4. We develop techniques for reasoning about *common-mode* failures. A common-mode failure occurs when the probabilites of the failure modes of two or more components are not independent. This issue has not been substantially addressed in the previous literature on model-based diagnosis.

[1] This article describe research conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided by the Information Systems Office of the Defense Advanced Research Projects Agency (DARPA) under Space and Naval Warfare Systems Center - San Diego Contract Number N66001-00-C-8078. The views presented are those of the author alone and do not represent the view of DARPA or SPAWAR.

5. We develop diagnostic techniques that lead to an estimate of the trustability of the computational resources that are used in a specific computation. These techniques also help us to assess which attacks have occurred and the likelihood that specific attacks have been successful.

These are crucial issues when failure is caused by a concerted and coordinated attack by a malicious opponent. There are many modes of attacking computational systems but the most pernicious attackers seek to avoid detection; therefore they attempt to scaffold the attack slowly, at a nearly undetectable rate. These scaffolding actions will typically appear as minor misbehaviors (i.e. they will cause the system to behave somewhat outside its normal range), but skillful attackers will space out the attacks so that the misbehaviors are infrequent and they will attempt to make the resulting misbehaviors seem as close to normal behavior as possible. This makes it crucial that our diagnostic techniques be capable of extracting information from low-frequency events that closely resemble normal modes of operation.

Attackers aim at high leverage points of the infrastructure, such as operating systems or middleware. This leads to common-mode faults, because once the operating system has been compromised all application components can be caused to fail simultaneously.

The paper first briefly reviews the current state of the art in model-based diagnosis; this work has mainly been concerned with breakdowns caused by the deterioration of hardware components. In particular, we adopt the framework in [4] where each component has models for each of several behavioral modes and each model is given a probability. We will then turn to the question of how to extend these techniques so as to apply them to the diagnosis of *software* systems. We extend our modeling framework to account for the fact that software systems are built in layers of infrastructure, with compromises to one layer affecting all higher levels. A software system has a great deal of hidden state; what we are actually capable of observing is the behavior of a specific *computation*; but this particular computation uses a variety of *resources* (e.g. the operating system and middleware, data-sets, etc.). These resources may have been subject to a variety of *compromises*, each of which might lead to a different misbehavior of the computation. Compromises to the resources occur because the resources possess *vulnerabilities* that allow specific *attacks* to take control of the resources for purposes other than those intended by the original designers.

We will finally present mixed symbolic and statistical diagnostic algorithms for assessing the posterior probabilities of the various behavior modes of each component in the model. We present an implementation and show an example of the reasoning process. Finally, we discuss the demands placed on the diagnostic component by our goal of self-adaptivity and conclude with suggestions for future research.

3 Related Research

Model-Based Diagnosis is a symptom directed technique; it is driven by the detection of discrepancies between the observations of actual behavior and the predictions of a model of the system. Almost all of the reported work in the area [2, 1, 3, 4, 5, 8] has been concerned with the diagnosis of physical systems subject to routine breakdown. Model-based diagnostic systems use simulation models that compute expected outputs given known inputs; they utilize dependency directed techniques to link each intermediate and final value to the selected behavioral model of any component of the system which was involved in producing that value.

The completeness of the diagnostic process is dependent on having bi-directional simulation models for each component of the system. Such models produce both a set of assertions recording what values are expected and a dependency network linking these assertions to one another and to assertions stating which components must be in a particular behavioral mode for those values to appear.

Our work builds on the framework in Sherlock [4] and on the probabilistic techniques in [8]. In Sherlock the description of a component includes multiple simulation models, one for each behavioral mode of the component. One distinguished mode is the normal mode, but behavioral models for known failure modes may also be provided. It is also typical to include a null model to account for unknown modes of behavior. Finally, each of the behavioral modes of a component is assigned an *a priori* probability. Sherlock uses these to guide a best first search for a set of behavioral modes, one for each component, such that the models for those modes predict the observed behavior. This is the most likely diagnosis. However, these techniques i depend on the assumption that the failure modes of the components are independent; as we will see this assumption doesn't hold in our environment. Later work [8] introduced techniques for applying Bayesian networks in the context of model-based diagnosis, allowing dependencies to be modeled; [10] presents techniques within this framework for generating several likely diagnoses in order of decreasing likelihood.

Because our focus is on detecting the intentional compromise of software components we are forced to face a number of new issues. These include: How to model software components in the spirit of model-based diagnosis; How to deal with the fact that a compromise to the computational infrastructure (e.g. the operating system) can manifest itself in the malfunction of many application components; How to deal with the fact that compromised components may behave in ways that are difficult to distinquish from normal behavior; How to reason about the system so as to extract as much information about possible compromises as we can. In particular, we deal with how to use both symbolic and Bayesian techniques.

4 Modeling Software Computations

Model-Based Diagnosis requires completely invertible models of the components in order to guarantee completeness of its analysis. But the components of a complex software system rarely have input-output relationship that are invertible. We therefore look for other, additional properties, that lead to more complete coverage. In particular, we concentrate here on descriptions of computational delay (or other *Quality Of Service* metrics). In our current implementation we use an interval of expected delay times (i.e. the computation should run no slower than x and no faster than y) as the behavioral models. Figure 1 shows the application of such models in a framework similar to Sherlock. When propagating in the forward direction we add the delay interval predicted by the behavioral model to the interval bounding the arrival time of the latest input. In the backward direction, we use interval subtraction (and only update the bounds on the last input to arrive). When more than one component predicts the bounds for a particular value (e.g. when a model for component A and a model for component C both predict bounds for the value labeled MID), we take the intersection of the two intervals to obtain the tightest bounds implied by the overall model. A discrepancy is detected when the lower bound of an interval exceeds the upper bound.

As in Sherlock we provide several behavioral models for each component, one characterizing normal behavior, others characteriz-

ing known failure modes and a null model to cover all other unexpected behaviors.

Notice that in Figure 1 there are six potential diagnoses, only one of which involves a single point of failure (in component C). The others involve multiple failues with one component running slower than expected and other components masking the fault at Out1 by running faster than expected. In the third diagnosis, component A runs in "negative time"! On the surface, such a diagnosis seems physically impossible and we might expect the diagnostic algorithm to reject it. But, the diagnosis algorithm is guided by our representational choices; the reason this diagnosis involves negative time is that the fast behavioral model of component A predicts a delay interval from -30 to +2.

Such behavior seems very unlikely, and indeed we assign a low likelihood to this model; however, it is not impossible. Suppose that both computations A and C are running on the same computer and further suppose that the computer has been compromised by an attacker. Under these circumstances, it's not impossible for component C to be delayed (because of a parasitic task inserted by the attacker) while component A has been accelerated, running in less than zero time because it has been hacked by the attacker to send out reasonable answers before it receives its inputs.

What we are able to observe is the progress of a computation; but the computation is itself just an abstraction. What an attacker can actually affect is something physical: the file representing the stored version of a program, the bits in main memory representing the running program, or other programs (such as the operating system) whose services are employed by the monitored application.

Thus, we require a more elaborated modeling framework detailing how the behavior of a computation is related to the state of the resources that it uses. In turn, we must represent the vulnerabilities of these resources and the attacks enabled by these vulnerabilities. Finally, we must represent how such attacks compromise the resources, causing them to behave in an undesired manner.

5 Common Mode Failures

A single compromise of an operating system component, such as the scheduler, can lead to anomalous behavior in several application components. This is an example of a *common mode failure*; intuitively, a common mode failure occurs when a single fault (e.g. an inaccurate power supply), leads to faults at several observable points in the systems (e.g. several transistors misbehave because their biasing power is incorrect). Another example comes from reliability studies of nuclear power plants where it was observed that the catostrophic failure of a turbine blade could sever several pipes as it flies off, leading to multiple cooling fluid leaks.

Formally, there is a common mode failure whenever the probabilities of the failure modes of two (or more) components are dependent. Early model-based diagnostic systems have assumed probabilistic independence of the behavior modes of different components [4] in order to simplify the assessment of posterior probabilities. Later work [8] allows for probabilistic dependence; however, it does not explore in detail how to model the causes of this dependence. We deal with common mode failures by extending our modeling framework to make explicit the mechanisms that couple the failure probabilites of different components.

We first extend our modeling framework, as shown in Figure 2, to include two kinds of objects: computational components (represented by a set of delay models one for each behavioral mode) and infrastructural components (represented by a set of modes, but no delay or other behavioral models). Connecting these two kinds of models are conditional probability links; each such link states how likely a particular behavioral mode of a computational component would be if the infrastructural component that supports that component were in a particular one of its modes (normal or abnormal). Each infrastructural component mode will usually project conditional probability links to more than one computational component behavioral mode, allowing us to say that normal behavior has some probability of being exhibited even if the infrastructural component has been compromised (however, for simplicity, figure 2 shows only a one-to-one mapping).

The model also includes *a priori* probabilities for the modes of the infrastructural components, representing our best estimates of the degree of compromise in each such piece of infrastructure. Following a session of diagnostic reasoning, these probabilities may be updated to the value of the *posterior* probabilities.

We next observe that resources are compromised by attacks. Attacks are enabled by vulnerabilities in the resources. For example, many systems in the Unix family are vulnerable to buffer-overflow attacks; most networked systems are vulnerable to packet-flood attacks. An attack is capable of compromising a resource in a variety of ways; for example, buffer overflow attacks are used both to gain control of a specific resource and to gain root access to the entire system. But the variety of compromises enabled by an attack are not equally likely (some are much more difficult than others). We therefore add a third tier to our model to describe the ensemble of attacks assumed to be available in the environment. We connect the attack layer to the resource layer with Conditional probability links that state the likelikhood of each mode of the compromised resource once the attack has been successful.

Our model of the computational environment therefore includes:

- The components of the computation that is being observed
- A set of behavioral models for each component, representing both normal and failure modes.
- The set of resources available to be used by the computational components
- A set of behavioral modes for each resource, representing both normal and compromised modes.
- A map stating which resources are used by each computational component.
- Conditional probabilties linking the modes of the computations to the modes of the resources employed by that component.
- A list of vulnerabilities possessed by each computational resource.
- A description of which attacks are enable by each vulnerability.
- A list of attack types that are believed to be active in the environment.
- A description of which compromised modes of each type of resource can be caused by a successful execution of each type of attack. This is provided as a set of conditional probabilities of the compromised mode given the execution of the attack.

Given this information, simple rule-based inferencing (implemented in the Joshua inference system) deduces which specific resources might have been compromised and with what probability. This information is then used to construct a Bayesian network (in the IDEAL system).

6 Diagnostic Reasoning

Figure 3 shows a model of a fictitious distributed financial system which we use to illustrate the reasoning process. The system con-

sists of five interconnected software modules (Web-server, Dollar-Monitor, Bond-Trader, Yen-Monitor, Currency-Trader) utilizing four underlying computational resources (WallSt-Server, JPMorgan, BondRUs, Trader-Joe).

For each computational component we show the conditional probability tables that describe how the behavioral modes of each computational resource probabilistically depend on the modes of the underlying resources (each resource has two modes, normal and hacked). Note that two computations (Dollar-Monitor and Yen-Monitor) are supported by a common resource (JPMorgan) and compromises to this underlying resource are likely to affect both computations. The failure modes of these two computations are no longer independent; this is indicated by the conditional probabilities connecting the behavior modes of the JPMorgan to those of both Dollar-Monitor and Yen-Monitor. The specific conditional probabilites supplied describe the degree of coupling.

Finally we show the *a priori* probabilities for the modes of the underlying resources. However, when attacks are present in the environment what matters is the conditional probabilities of the different modes of the resources given that an attack has taken place. We hypothesize that one or more attack types are present in the environment, leading to a three-tiered model as shown in figure 4. In this example, we show two attack types, buffer-overflow and packet-flood. Packet-floods can affect each of the resources because they are all networked systems; buffer-overflows affect only the 2 resources which are modeled as instances of a system type vulnerable to such attacks.

As in earlier techniques, diagnosis is initiated when a discrepancy is detected; in this case this means that the predicted production time of an output differs from those actually observed after an input has been presented. The goal of the diagnostic process is to infer as much as possible about where the computation failed (so that we may recover from the failure) and about what parts of the infrastructure may be compromised (so that we can avoid using them again until corrective action is taken). We are therefore looking for two things: the most likely explanation(s) of the observed discrepancies and updated probabilities for the modes of the infrastructural components.

To do this we use techniques similar to [4, 8]. We first identify all conflict sets, and then proceed to calculate the posterior probabilities of the modes of each of the computational components. We do these tasks by a mixture of symbolic and Bayesian techiques; symbolic model-based reasoning is used to predict the behavior of the system, given an assumed set of behavioral modes. Whenever the symbolic reasoning process discovers a conflict (an incompatible set of behavioral modes), it adds to the Bayesian network a new node corresponding to the conflict (see below). Bayesian techniques are then used to solve the extended network to get updated probabilities.

This approach involves an exhaustive enumeration of the combinations of the models of the computational components. This allows us to calculate the exact posterior probabilties. However, this is expensive and the precision may not be needed. It would be possible to instead use the techniques in [10] to generate only the most likely diagnoses and to use these to estimate the posterior probabilities; but we have not yet pursued this approach.

We instead follow the following approach: We alternate the finding of conflicts with the search for diagnoses. After each “conflict” node is added to the Bayesian network (see below) the network is solved; this gives us updated probabilities for each behavioral mode of each component. We can, therefore, examine the behavioral modes in the current conflict and pick that component whose current behavioral mode is least likely. We discard this mode, and pick the most likely alternative; we continue this process of detecting conflicts, discarding the least likely model in the conflict and picking its most likely alternative until a consistent set is found. This process is a good heuristic for finding the most likely diagnosis [2].

Our models of computational behavior (the delay models) are used to predict the behavior of the computational components and to compare the predictions with observations. When a discrepancy is detected, we use dependency tracing to find the conflict set underlying the discrepancy (i.e. a set of behavioral modes which are inconsistent). At this point a new (binary truth value) node is added to the Bayesian network representing the conflict as shown in Figure 5. This node has an incoming arc from every node that participates in the conflict. It has a conditional probability table corresponding to a pure ”logical and” i.e. its true state has a probability of 1.0 if all the incoming nodes are in their true states and it otherwise has probability 1.0 of being in its false state.

Since this node represents a logical contradiction, it is pinned in its false state. Adding this node to the network imposes a logical constraint on the probabilistic Bayesian network; the constraint imposed is that the conflict discovered by the symbolic, model-based behavioral simulation is impossible. We continue to explore other combinations of behavioral modes, until all possible minimal conflicts are discovered. Each of these conflicts extends the Bayesian network as before, The set of such conflicts constitutes the full set of logical constaints on the values taken on within the Bayesian network; thus, once we have augmented the Bayesian network with nodes corresponding to each conflict, the network has all the information available. [3].

At this point, we have found all the minimal conflicts and added conflict nodes to the Bayesian network for each. We therefore also know all the possible diagnoses since these are sets of behavioral modes (one for each component) which are not supersets of any conflict set. For each of these we create a node in the Bayesian network which is the logical-and of the nodes corresponding to the behavioral modes of the components. This node represents the probability of this particular diagnosis. The Bayesian network is then solved. This gives us updated probabilities for all possible diagnoses, for the behavioral modes of the computational components and for the modes of the underlying infrastructural components. Furthermore, these updated probabilities are those which are consistent with all the constraints we can obtain from the behavioral models. Thus, they represent as complete an assessment as is possible of the state of compromise in the infrastructure. These posterior estimates can be taken as priors in further diagnostic tasks and they can also be used as a “trust model” informing users of the system (including self adaptive computations) of the trustworthiness of the various pieces of infrastructure which they will need to use.

7 Results

The sample system shown in Figure 3 was run through several analyses including both those in which the outputs are within the expected range and those in which the outputs are unexpected. Figure 6 shows the results of an analysis in which the outputs are within the expected range. Figure 7 and 8 show the results of an analysis of an

[2] However since the probabilities of the failure modes of different components are not independent, this is only a heuristic

[3] [8] builds logical reasoning directly into the Bayesian network system because the logical inferences needed are simple enough to be accomodated. However, our inference needs are more complex and not easily amenable to this approach

abnormal case. Inputs are supplied at times 10 and 15 for the two inputs of Web-Server; each of the figures shows the times at which the the outputs of Currency-Trader and Bond-Trader are observed. There are four runs for each case, each with a different attack model. In the first, it is assumed that there are no attacks present and the *a priori* values are used for the probabilities of the different modes of each resource. The second run assumes only a buffer-overflow attack; the third run assumes only a packet-flood attack. The fourth run assumes both types of attacks. There are four columns in each of the results chart, one for each of these runs. The top chart in each figure shows the *a priori and posterior* probabilities for each resource being in its "hacked" mode. The middle chart shows the *posterior* probabilities for each mode of each computational component. The bottom bottom chart in each figure shows the *posterior* probabilites that each of the two types of attacks have occurred. [4].

There are more than two dozen possible diagnoses in the abnormal case. It should be noted that even the most likely diagnosis is actually not all that likely; in addition the next several diagnoses are nearly equally as likely. The most likely diagnosis is therefore not particularly informative for our two goals of recovering from the failure and steering away from compromised resources in the future. However, the posterior probabilities of the modes of the infrastructure components are, in fact, useful guides for the second of these goals. The posterior probabilities of the behavioral modes of the computational resources are useful guides for the first goal, because these probabilities aggregate the information contained in the individual diagnoses.

The most significant change is the increase in the probabilities that the resources named JPMorgan and Wallst-server are hacked. This changes the trustworthiness ordering of the resources: JPMorgan is *a posteriori* the least trustworthy resource, while the *a priori* listing ranks Trader-Joe followed by Bonds-R-US as the least trustworthy. This follows from the fact that the JPMorgan resources is utilized by the computations Yen-Monitor and Dollar-Monitor both of which are very likely to be in abnormal modes and the most likely explanation is that that JPMorgan causes a common-mode failure. Notice that in the last two columns when packet-flood attacks are possible, all the resources are much more likely to be hacked. Qualitatively, this is because all the resources are vulnerable to the packet-flood attack. The misbehavior of the computational components provides evidence that JPMorgan is hacked which in turn provides evidence of a packet flood attack. But since packet-flood attacks affect all the resources, this increases the likelihood that other resources are hacked as well. The Bayesian network carries out the quantitative version of this argument.

It is worth noting that this propagation of trust can carry over to resources not used in the misbehaving computation. For example, assume that the environment contains another resource (call it "newbie") that is subject to the same attacks as the ones (e.g. JPMorgan) that participated in the faulty computation. The misbehavior in the computation is evidence that JPMorgan is "hacked" and this, in turn, is evidence that an attacked succeeded. But this would lend weight to the conclusion that other resources (e.g. Newbie) subject to this same attack had also been compromised. The Bayesian network would propagate probabilities in exactly this fashion leading to a posterior assessment that Newbie has been hacked (although this probability will be lower than the probability that JPMorgan is hacked).

8 Conclusions and Future Work

The example above illustrates how model-based reasoning techniques can be used to extract information from a single run. Our example is intentionally fanciful since we are at the present concentrating on the development of the representational and reasoning frameworks. In future work we will explore realistic models of real systems.

The information extracted is probabilistic and it sheds light both on the question of where the computation might have failed, on what underlying resources might have been compromised and on what attacks might have succeeded.

It is notable that the identification of the most likely diagnosis is not particularly informative. For example, in the most likely diagnosis Yen-Monitor is in its Normal mode. However, the most likely behavioral mode for Yen-Monitor is its Slower mode which occurs in many of the remaining diagnoses. The posterior probabilites of the behavioral modes aggregate the probabilites from each of the possible diagnoses, producing an overall assessment that is more informative than any individual diagnosis. Of course, if there are very few diagnoses, or the most likely diagnosis is extremely probable, then the probabilities of its behavioral modes will approximate the overall posterior probabilities.

It is important to keep in mind why we are interested in the diagnoses at all. The goal of the system is to recover from the failure and to steer away from future trouble. To do this it needs to know how much of the computation has been completed successfully and how much remains to be done. Given such information the system would pick a rollback point for recovery that includes no failed part of the computation. Furthermore, the chosen rollback point would maximize the probability of continuing the restarted computation to completion. As we just saw, an individual diagnosis, even the most likely diagnosis, does not give us the information we need to do this. When the available evidence supports multiple diagnostic hypotheses, then our interest should shift from individual diagnoses to aggregate failure probabilities and this information is conveyed completely by the posterior probabilities of the failure modes. I.e. if the posterior probability that Yen-Monitor failed is high, then we don't actually care that there are multiple (multiple point of failure) diagnoses involving this failure nor do we care how likely each of these diagnoses is. Instead what we do care about is that it's very likely that Yen-monitor didn't do its job and that we should select a rollback point prior to its execution. Similarly, in choosing a recovery plan we should avoid using those resources whose posterior failure probabilities are highest. [5].

This is to say that the goal of the diagnostic process should be to assess the overall posterior probabilities of the behavioral modes of the computational and infrastructure components. These give us evidence for which computational resources are to be to be trusted during the recovery process and during subsequent computations. This is

[4] The implementation is in CommonLisp and uses the Joshua [7] rule-based reasoning system as well as the Ideal system [9] and in particular its implementation of the algorithm described in [6]. On a 300 MHz powerbook, the total solution time is under 1 minute. By far, the most expensive part of this is calculating the probabilities of the complete set of diagnoses. The most likely diagnosis and all conflict sets are located in less than 10 seconds)

[5] Of course, gathering further evidence might reduce the number of possible diagnoses leading to greater resolution. However, in our context there are two difficulties with attempting to do this. First, it would take time and there might be tight timeliness contraints on the failed computation (e.g. suppose the computation was processing sensor data which must be acted on very quickly). Second, any attempt to gather more data would involve running the same, or similar, computations again when we know that something is compromised; this might lead to loss or destruction of data. Making this tradeoff correctly involves estimating the expected cost of new information and it expected benefit. It is possible that such and analysis would suggest that acting on the available diagnostic evidence is the best course of action

a different definition of the goal of diagnostic activity than has been used in previous research on model-based diagnosis.

We have not yet addressed the details of how the system should use this information in forming a recovery plan. The general outline is that when assigning a computation to a resource it should choose that resource which is most likely to be in n mode that will successfully complete the computation. But the probabilities of the modes of different resources are not independent; they are linked by the Bayesian network. Having decided to use a particular resource because it's likely to be in an acceptable mode, the system should pin the Bayesian network into a state where the resource is believed to be in the desired state and re-solve the network. Subsequent choices should be made in light of the updated probabilities.

We have also not yet addressed the question of what actions the system might take to obtain more information in future runs. The Minimum Entropy approach in [3] provides a useful framework. However, the current context provides more degrees of freedom; in addition to making new observations, we can also change the assignment of resources to computational components in a way that will maximize the expected gain in information. The details of this remain for future research.

REFERENCES

[1] Randall Davis, 'Diagnostic reasoning based on structure and behavior', *Artificial Intelligence*, **24**, 347–410, (December 1984).

[2] Randall Davis and Howard Shrobe, 'Diagnosis based on structure and function', in *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 137–142. AAAI, (1982).

[3] Johan deKleer and Brian Williams, 'Diagnosing multiple faults', *Artificial Intelligence*, **32**(1), 97–130, (1987).

[4] Johan deKleer and Brian Williams, 'Diagnosis with behavior modes', in *Proceedings of the International Joint Conference on Artificial Intelligence*, (1989).

[5] Walter Hamscher and Randall Davis, 'Model-based reasoning: Troubleshooting', in *Exploring Artificial Intelligence*, ed., Howard Shrobe, 297–346, AAAI, (1988).

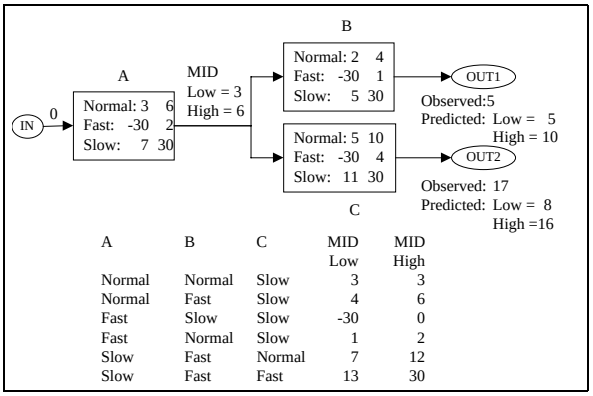
[6] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen, 'Bayesian updating in causal probablistic networks by local computations', *Computational Statistics Quarterly*, **4**, 269–282, (1990).

[7] S. Rowley, H. Shrobe, R. Cassels, and W. Hamscher, 'Joshua: Uniform access to heterogeneous knowledge structures (or why joshing is better than conniving or planning)', in *National Conference on Artificial Intelligence*, pp. 48–52. AAAI, (1987).

[8] Sampath Srinivas, 'Modeling techinques and algorithms for probablistic model-based diagnosis and repair', Technical Report STAN-CS-TR-95-1553, Stanford University, Stanford, CA, (July 1995).

[9] Sampath Srinivas and Jack Breese, 'Ideal: A software package for analysis of influence diagrams', in *Proceedings of CUAI-90*, pp. 212–219, (1990).

[10] Sampath Srinivas and Pandurang Nayak, 'Efficient enumeration of instantiations in bayesian networks', in *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 500–508, Portland, Oregon, (1996).



A	B	C	MID Low	MID High
Normal	Normal	Slow	3	3
Normal	Fast	Slow	4	6
Fast	Slow	Slow	-30	0
Fast	Normal	Slow	1	2
Slow	Fast	Normal	7	12
Slow	Fast	Fast	13	30

Figure 1. Reasoning About Software Component Delay

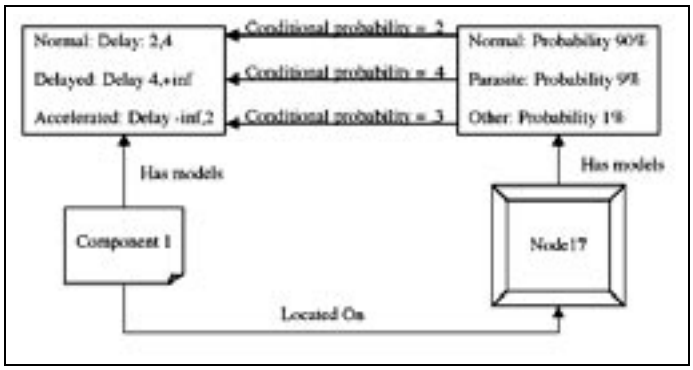


Figure 2. Modeling Computational and Infrastructure Components

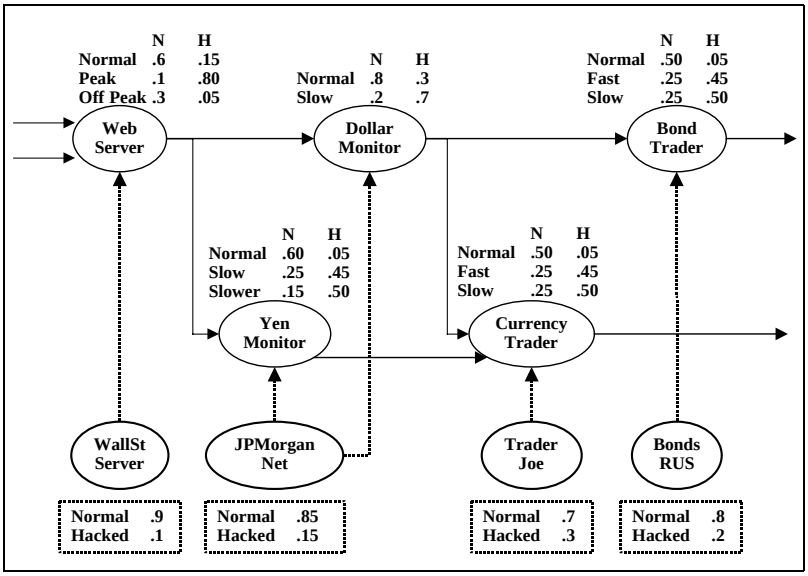


Figure 3. An Example of the Extended System Modeling Framework

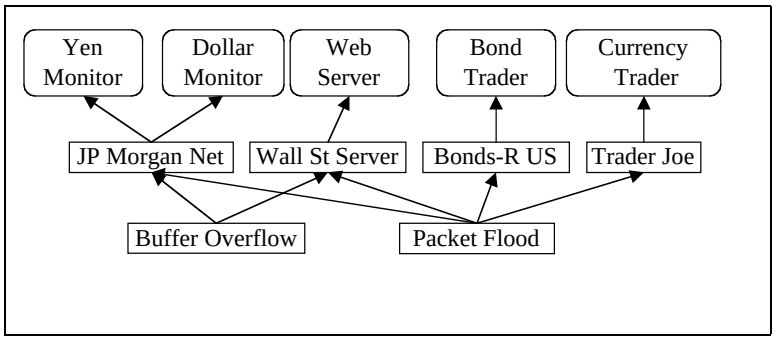


Figure 4. An Example of the Three Tiered System Modeling Framework

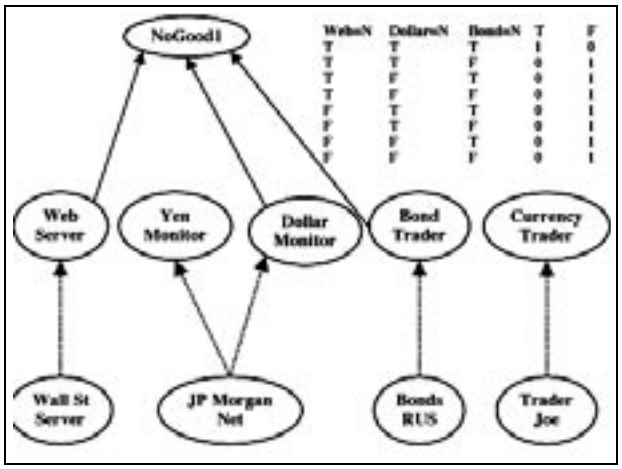


Figure 5. Adding a Conflict Node to the Bayesian Network

Normal Case: 48 “Diagnoses”
30 Minimal Conflicts
Output of Bond-Trader Observed at 25
Output of Current-trader Observed at 28

Name	Prior	Posterior			
Wallst	.10	.04	.07	.09	17
JPMorgan	.15	.07	.08	.11	.19
Bonds-R-Us	.20	.18	.18	.15	.17
Trader-Joe	.30	.28	.28	.16	.18

Computations Using Each Resource					
Web-Server	Off-Peak	.40	.40	.40	.40
	Peak	.04	.05	.05	.08
	Normal	.55	.56	.55	.52
Dollar-Monitor	Slow	.23	.23	.25	.29
	Normal	.77	.77	.75	.71
Yen-Monitor	Really-Slow	.03	.04	.04	.05
	Slow	.21	.21	.21	.25
	Normal	.76	.75	.75	.70
Bond-Trader	Slow	.29	.29	.27	.26
	Fast	.23	.23	.24	.26
	Normal	.48	.48	.49	.48
Currency-Trader	Slow	.09	.09	.07	.06
	Fast	.52	.52	.48	.51
	Normal	.40	.39	.45	.43

Attack Types		Attacks Possible			
Name	Prior	None	Buffer-Overflow	Packet-Flood	Both
Buffer-Overflow	.4	0	.28	0	.30
Packet-Flood	.5	0	0	.23	.25

Figure 6. Updated Probabilities

Slow Fault on both outputs 25 “Diagnoses”
34 Minimal Conflicts
Output of Bond-Trader Observed at 35
Output of Current-trader Observed at 45

Name	Prior	Posterior			
Wallst	.1	.27	.58	.75	.80
JPMorgan	.15	.45	.62	.74	.81
Bonds-R-Us	.20	21	.20	.61	.50
Trader-Joe	.30	.32	.31	.62	.50

Computations Using Each Resource					
Web-Server	Off-Peak	.03	.02	.02	.02
	Peak	.54	.70	.78	.80
	Normal	.43	.28	.20	.18
Dollar-Monitor	Slow	.74	.76	.73	.76
	Normal	.26	.24	.27	.24
Yen-Monitor	Really-Slow	.52	.54	.56	.58
	Slow	.34	.35	.34	.34
	Normal	.14	.11	.10	.08
Bond-Trader	Slow	.59	.57	.76	.70
	Fast	0	0	0	0
	Normal	.41	.43	.24	.30
Currency-Trader	Slow	.61	.54	.62	.56
	Fast	.07	.11	.16	.16
	Normal	.32	.35	.22	.28

Attack Types		Attacks Possible			
Name	Prior	None	.Buffer-Overflow	Packet-Flood	Both
Buffer-Overflow	.4	0	.82	0	.58
Packet-Flood	.5	0	0	.89	.73

Figure 7. Updated Probabilities

Prob ability	Currency Trader	Bond Trader	Yen Monitor	Dollar Monitor	Web Server
.0898	Slow	Slow	Normal	Normal	Peak
.0876	Slow	Normal	Slow	Slow	Normal
.0855	Normal	Normal	slower	Slow	Normal
.0762	Slow	Normal	Really-Slow	Slow	Normal
.0641	Slow	Slow	Slow	Slow	Normal
.0626	Normal	Slow	Really-Slow	Slow	Normal
.0557	Slow	Slow	Really-Slow	Slow	Normal
.0468	Normal	Slow	Slow	Normal	Peak
.0416	Slow	Slow	Slow	Normal	Peak
.0321	Slow	Normal	Normal	Slow	Peak
.0306	Normal	Slow	slower	Normal	Peak
.0301	Normal	Normal	Slow	Slow	Peak
.0276	Slow	Slow	slower	Slow	Off-Peak
.0272	Slow	Slow	slower	Normal	Peak
.0268	Slow	Normal	Slow	Slow	Peak
.0262	Normal	Normal	slower	Slow	Peak
.0260	Fast	Slow	slower	Normal	Peak
.0235	Slow	Slow	Normal	Slow	Peak
.0233	Slow	Normal	slower	Slow	Peak
.0223	Fast	Normal	slower	Slow	Peak
.0221	Normal	Slow	Slow	Slow	Peak
.0196	Slow	Slow	Slow	Slow	Peak
.0192	Normal	Slow	slower	Slow	Peak
.0171	Slow	Slow	slower	Slow	Peak
.0163	Fast	Slow	slower	Slow	Peak

Figure 8. diagnoses