

Intelligent processing agents that reside at network nodes use adaptive learning methods to detect abnormal network behavior before a fault actually occurs.

In a test at RPI, an agent on a router detected a file server failure 12 minutes before it occurred.

INTELLIGENT AGENTS FOR PROACTIVE FAULT DETECTION

CYNTHIA S. HOOD

Illinois Institute of Technology

CHUANYI JI

Rensselaer Polytechnic Institute

As the Internet becomes a critical component of our society, a key challenge is to maintain network availability and reliability. The increasing number of networks is making downtime more costly. Network management tools and techniques must therefore become better at detecting and identifying network faults.

Fault management is no easy task, however. Fundamental changes to the network occur much more frequently as demand grows and components and applications are developed in an open environment. Although mixing and matching hardware and software from different vendors supports customization and use of the latest technologies, it also increases the risk of faults and other problems.¹

Current fault management implementations generally rely on the expertise of a human network manager, which is translated to a set of rules and then to threshold levels on the measurement variables being collected. As networks become more complex and changes occur more frequently, the human network manager will find it hard to maintain a sufficient level of expertise on a particular network's behavior.

Fault management research has covered approaches such as expert systems, finite state machines, advanced database techniques, and probabilistic methods. Aurel Lazar and colleagues give a good review of communication network fault detection and identification.² The drawback to all these approaches is that they require a specification of the faults to be detected, and it is not feasible to specify all possible faults. Also, changes in network configuration, applications, and traffic can alter the type and nature of possible faults, which makes modeling them impractical in many cases.

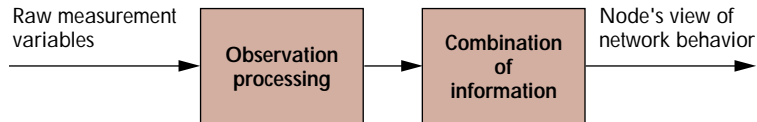


Figure 1. Structural overview of the intelligent processing agent.

Another research approach is to use learning machines that detect anomalies.³ Although this approach does not require fault modeling, neither does it provide a way to correlate the information collected in space or time.

In this article, we describe an intelligent agent that processes information collected by Simple Network Management Protocol agents and uses it to detect the network anomalies that typically precede a fault. The SNMP agents collect information about the network node through their management information base, or MIB, which holds a set of variables pertinent to that particular node. The intelligent agents learn the normal behavior of each measurement variable and combine the information in the probabilistic framework of a Bayesian network. This yields a picture of the network's health from the perspective of the network node, which can be used to trigger local corrective action or a message to a centralized network manager.

The intelligent agent approach has several benefits. It can detect unknown faults and correlate information in space and time. Because it can detect subtle changes that precede an actual fault, the node or a network manager can correct the condition, possibly preventing outages or downtime. The intelligent agents themselves require very little network specification and can be generalized across network nodes and network types. Finally, SNMP agents are included in most pieces of equipment for the Internet, so there is no need for specialized hardware to monitor the network.

In developing the intelligent agent, we collected data from the network used by RPI's Computer Science Department. The network consists of seven subnetworks and two routers. The agent, which was on a correctly operating router, detected a failure in one of the subnetwork's components 12 minutes before it occurred.

STRUCTURAL OVERVIEW

Figure 1 is a diagram of how the intelligent agent approach works. Each network node must compose its own picture of the network's health. To provide the node with this picture, the agent first uses its *observation processing* component to process raw measurement variables and obtain the proba-

bility of each measured variable at a given time. It then uses its *combination of information* component to combine the probabilities using a Bayesian network, which provides a method for estimating probabilities and allows the agent to combine observed information with prior knowledge. The agent can therefore provide the node with a complete, less noisy (relative to using individual variables) picture of the network's health from that particular node.

Observation Processing

The observation processing component uses a change-detection method to characterize the behavior of measurement variables. Because network behavior is dynamic, the behavior of the measurement variables changes frequently. Most changes are related to network traffic, so simply detecting that a change has occurred is not enough. The goal is to recognize changes that are important in the context of network behavioral anomalies that correspond to network faults. This process has three aspects: segmenting data in a meaningful way, extracting features that tell whether the variable is exhibiting normal or abnormal behavior, and learning what constitutes a variable's normal and abnormal behavior.

Segmentation. Because a network is dynamic, observations are necessarily made over time. The agent uses an algorithm⁴ that segments data into variable-length

pieces, each of which contains a portion of the time series that is statistically similar. The algorithm assumes that the time series is from a piecewise wide-sense stationary Gaussian process. After segmentation the signals are in stationary pieces.

Segmentation offers three main benefits. First, the statistics calculated from each segment are more representative of the signal. Second, the signal-processing techniques that require a stationary signal can be used within each segment. Third, segmentation temporally correlates the observations. Because many network signals are bursty, the temporal correlation can help the agent distinguish between a burst and a change in the signal's nature.

Feature extraction. Once it has grouped the observations into segments, the agent must extract information, or *features*, pertinent to fault detection from each segment. There are many ways to extract features, most of which have distinct disadvantages. The most common method for detecting

**The intelligent agent can
detect unknown faults and
correlate information in
space and time.**

abnormal behavior is thresholds. The feature is not the value of the threshold itself, but information on whether or not a particular measurement variable has exceeded the threshold. If there are both upper and lower thresholds, the feature would be information that tells if the variable is between the two thresholds. Because thresholds are highly dependent on the traffic level, it is hard to set the proper threshold level. Improperly set thresholds may never be exceeded, thereby letting problems go undetected, or they may be exceeded too often, flooding the network manager with false alarms.

Moreover, even if thresholds are properly set, they tend to detect only large rises and falls in a measurement variable, missing the more subtle behavioral changes. Figure 2 illustrates this. The level of a signal drops off whenever a measured variable changes. This tells nothing about the nature of the change, however. Is it normal or abnormal behavior for that variable?

Ideally, we want to capture only the signal information that will change or become abnormal when a problem is occurring. To do this, we need features that can use network traffic measurements to distinguish abnormal from normal network behavior. Unfortunately, this is an open problem, so we chose features that change along with the network and allow the normal behavior model to continually adapt. Using these features, we are able to detect the more subtle signs of problems, which in turn allows corrective actions that avoid a bigger problem.

Our change-detection method uses the parameters of a second-order autoregressive process AR(2) as features. The AR(2) process is defined as

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + e(t)$$

where $y(t)$ is the value of the signal at time t , a_1 and a_2 are the AR parameters used as features, and $e(t)$ is a white-noise process.

One of our motivations for using AR parameters was simplicity. Ideally, if an underlying stochastic process that governs the data is composed of piecewise Gaussian processes, the extracted AR parameters would be able to characterize the process well, assuming the order of the AR process is properly chosen. Otherwise, our AR parameters would correspond to the best linear approximation in terms of mean-square-error. Others have tried higher order AR processes, but observed no significant gain in detection ability.⁵

Learning behavior. The agent uses the features just described to establish a description of normal behavior in the form of a probability distribution.

To estimate the likelihood of each sample when the network is operating normally *and* when there is a fault, we must know exactly when the network is operating normally from the node's view. However, this information is not typically available. Instead, we use information about the network's health from the log files generated by problem-reporting tools. The log files contain reports on certain types of network problems that have occurred, so from a learning perspective, we can use these reports as measurement "labels" that tell when the faults occurred.

One way to use these labels is to learn the probability distribution of each measurement variable when its related network function is abnormal. This is extremely difficult, however, because with so few examples of network problems, there is not yet the variety (many examples of different problems) or depth (several examples of the same type of problem) for the agent to effectively learn the distribution. Instead, the agent must learn the probability distribution when the variable's related network function is normal. We define normal behavior as the variable's behavior during the *learning window*—the period in which the agent learns the distribution. We assume that the agent will rarely learn problematic behavior as normal behavior.

Because we are unable to use data to estimate the probability distribution when the network is abnormal, we treat this distribution as unknown⁶ and assume a uniform distribution over a range of the AR(2) parameters.⁷ Any sample that falls outside this range is considered abnormal.

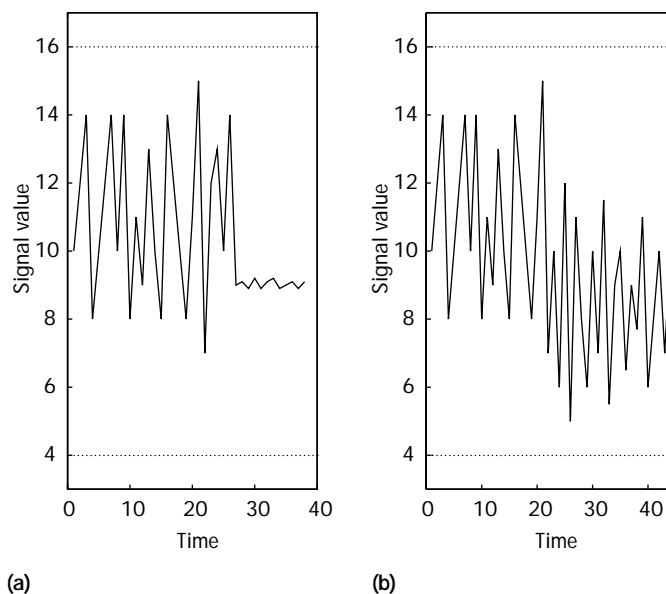


Figure 2. Examples of subtle behavior not detected using thresholds (dotted lines). Thresholds can miss (a) sudden drop-offs in the signal and (b) more subtle variations. Even if thresholds do catch these differences, they do not provide any information that classifies the change as normal or abnormal.

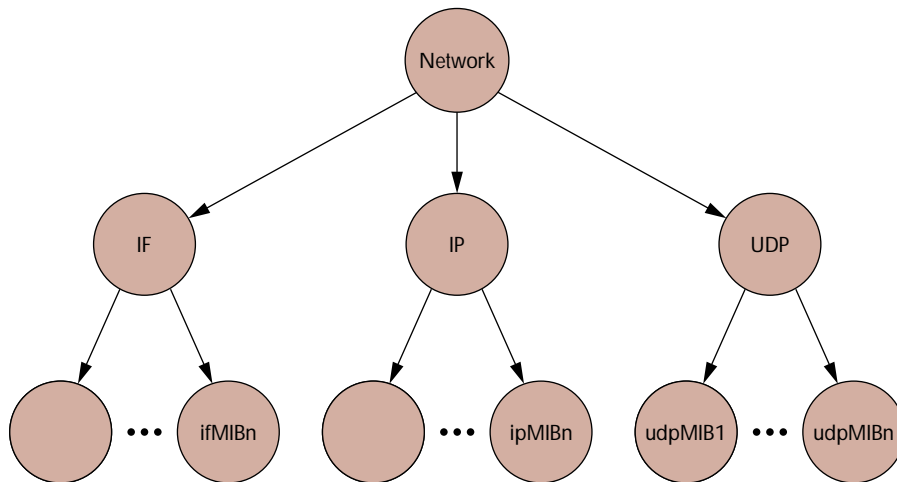


Figure 3. Bayesian network for fault detection. At the top level is the network variable, which corresponds to the network’s entire functionality. At the next level down are the internal network variables, which represent different types of network functionality. At the bottom level are the MIB variables, which are stored in each SNMP agent’s management information base. The arrows between the various nodes go from cause to effect.

Combination of Information

The goal of this component is to combine the processed measurement variable information into higher level measures of network behavior from the network node’s view. These measures, which are the intelligent agent’s outputs, can be used to trigger local control actions or a message to a centralized network manager.

Each measurement variable is combined in the probabilistic framework of a Bayesian network,^{8,9} whose structure is shown in Figure 3. The agent begins the combination process by defining the random variables or nodes in the Bayesian network. Two types of variables must be estimated. The *observed* variables are

- network (top node), which corresponds to all the network functions, and
- MIB variables, which are the features from the SNMP agent’s management information base (bottom row of nodes).

The *observed* variables represent the network’s health from the intelligent agent’s view. The MIB variables within a group are the measurement variables for that network function.

The *not observed* variables, or internal variables (middle row of nodes in the figure), include

- IF, which represents the network interface,
- IP, which represents the Internet Protocol, and
- UDP, which represents User Datagram Protocol.

These variables correspond to the MIB groups and thus represent different types of network functionality. In our work, we assume that the network variable comprises only the IF, IP, and UDP functions, since these were the functions used at the routers on the network we used in developing the intelligent agent, and the router is the node where the agent resides. It is straightforward to add other network functions.

In our model, the network’s health is the most general information estimated and can be considered an underlying influence on the rest of the nodes in the Bayesian network. As

the figure shows, the network’s overall health directly influences the health of the network’s three functions, IF, IP, and UDP (denoted by the cause-effect arrows going from the network node to the corresponding three nodes). Likewise, for each network function, the health of that function directly influences the values of the individual measurement or MIB variables for that function. One way to interpret this is that a problem in the network will cause symptoms to show up in the MIB variables of that function.

We have designed our Bayesian network model on the basis of these intuitive relationships. The model requires the conditional independence assumptions described in the sidebar “Bayesian Networks” on the next page. We assume that given knowledge of the network’s health, the health of the three network functions is independent. Given the health of a network function, the measurement variables for other network functions do not contribute any additional information. These assumptions are reasonable, since each network function represents an independent functional network component. These components may fail independently, although there is a relationship between the functions, and serious problems in one component can eventually impact the others.

This conditional independence is an assumption we use as the first step in simplifying the problem of combining information. We realize that the MIB variables in general can be dependent, and their dependence can be complicated, changing with time. We plan to address this in future research.

Each internal variable (IF, IP, and UDP) and the variable for network health (network) is discrete and has two states, normal and abnormal. Each MIB variable is continuous.

As a first step in determining the correct state, we must estimate a set of prior, or posterior, probabilities:

$$p(n = \text{normal/abnormal} \mid \text{MIBs}) \tag{1}$$

and

$$p(nf = \text{normal/abnormal} \mid \text{MIBs}) \tag{2}$$

where n is the network and nf is one of the network functions (IF, IP, or UDP). These two probabilities correspond to the health of the network and the internetwork layers given the observations.

Due to the tree or singly connected structure of the Bayesian network, these probabilities can be calculated efficiently either directly or using Pearl's algorithm.⁹ The equations for the direct calculation can easily be derived using the conditional independence assumption stated earlier and Bayes' rule. We can then reduce the task of estimating posterior probabilities to simply estimating the conditional probability:

$$p(\text{MIB variable} \mid nf = \text{normal/abnormal}) \tag{3}$$

These are the conditional probabilities of the observations given the health of the network, and can be directly estimated from data. In addition, we must determine the following prior probabilities to completely specify the probabilities given in Equations 1 and 2:

$$p(\text{network} = \text{normal} / \text{abnormal}); \tag{4}$$

$$p(nf = \text{normal} / \text{abnormal} \mid \text{network} = \text{normal} / \text{abnormal}) \tag{5}$$

We estimate the prior probability in Equation 4 and the conditional probabilities in Equation 5 using prior knowledge of the network behavior gained from observations and conversations with the network managers. These probabilities remain constant throughout the learning process. We estimate the conditional probabilities in Equation 3 using the observed MIB variables.

Since the intelligent agent is monitoring locally, all the probabilities estimated from the observed MIB variables are available to the system. The system can then calculate the desired posterior probabilities using a complete and current set of observations.

DATA COLLECTION

Figure 4 shows the network we used to collect data for the intelligent agent. It is made up of seven subnetworks, or subnets, and two routers. The nodes on the subnets, such as workstations, printers, and file servers, are not shown except for two file servers (fs1 and fs2) and the data collection machine (nm).

BAYESIAN NETWORKS

A Bayesian network, also called a belief network or a causal network, is a graphical representation of relationships within a problem domain. Bayesian networks provide a theoretical framework for combining statistical data with prior knowledge about the problem domain, which makes them particularly useful in practical applications. Indeed they have been widely used for medical diagnosis, troubleshooting, and in communications.¹

In formal terms, a Bayesian network is a *directed acyclic graph*, in which certain conditional independence assumptions hold.² The DAG's nodes represent random variables. The conditional independence assumptions are as follows:

Given a DAG $G = (N, E)$, where $n \in N$ is a node in the network, and $e \in E$ is a directed arc. For each $n \in N$, let $p(n) \subseteq N$ be the set of all parents of n , and $d(n) \subseteq N$ be the set of all descendants of n . For every subset $W \subseteq N - (d(n) \cup \{n\})$, W and n are conditionally independent given $p(n)$.³

In other words, for any node in the DAG, given that node's parents, that node is independent of any other node, not its descendant.

These assumptions let us use algorithms² to estimate the conditional probabilities of any of the nodes (or random variables) in the Bayesian network, given the observed information or evidence. The efficiency of the algorithm depends on the structure of the Bayesian network.

REFERENCES

1. D. Heckerman, "A Tractable Algorithm for Diagnosing Multiple Diseases," *Proc. Workshop Uncertainty in Artificial Intelligence*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 174-181.
2. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufman, San Mateo, Calif., 1988.
3. R. Neapolitan, *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley & Sons, New York, 1990.

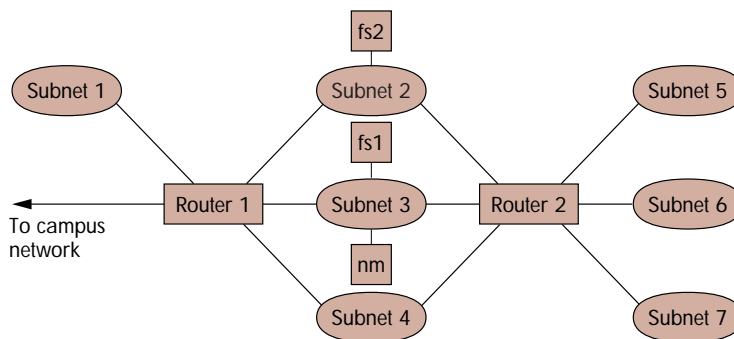


Figure 4. Configuration of the network for data collection. For simplicity, no subnet nodes are shown except the file servers (fs1 and fs2) and the data collection machine (nm).

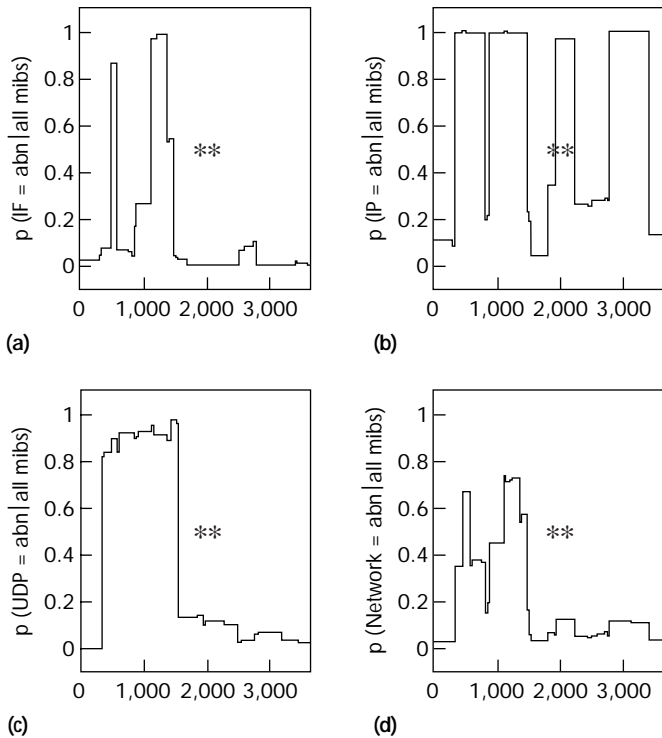


Figure 5. Network results reported by the intelligent agent. The horizontal axes are time in seconds. The vertical axes are the posterior probabilities of abnormal behavior in the (a) IF variable, (b) IP variable, (c) UDP variable, and (d) network variable. By combining information from (a) through (c), the agent was able to detect abnormal behavior 12 minutes before the server was reported unreachable (denoted by asterisks).

Router 1 is the gateway between this network and the campus network, with all traffic to and from the campus and the outside world flowing through it. Router 2 routes mainly the local traffic between the subnetworks. A large part of this traffic is access from workstations to the file servers. Data collectors gathered data from Router 2, the internal router, by polling it every 15 seconds using SNMP queries. Among the 45 available MIB variables, only 14 were consistently active. The group collected data from the router continuously from October 1995 to March 1996. During this time, they also saved the log files generated by the `syslog` function so that we could label the faults within the data. Because `syslog` is not targeted specifically for network errors, it reports only a subset of all network problems. Its reports do not provide any information about the cause of the problem.

EXPERIMENTAL RESULTS

We tested the intelligent agent on a set of 10 faults observed on the network diagrammed in Figure 4 during data collection. Nine of the 10 faults were recorded as “server not responding,” which indicates a severe problem that could be due to the server being down or the path being unavailable.

The remaining fault was reported as excessive Ethernet collision on one of the subnets. Because of the limitations of `syslog`, we were able to observe only faults in which a service the network provides was not operational.

Although nine of the faults reported by `syslog` messages were the same, we did not observe the same types of data changes from fault to fault—possibly because the faults were caused by different sets of circumstances or root causes. However, we were not able to determine this because we did not know the implementation details of the network nodes.

Sample Fault

One of the faults reported as “server not responding” involved the file server on subnet 2. A total of 13 workstations reported this problem (seven on Subnet 2, four on Subnet 3, and two on Subnet 4). Using a one-hour learning window, we estimated the results for the posterior probabilities as $p(\text{network} = \text{abnormal} | \text{MIBs})$, $p(\text{IF} = \text{abnormal} | \text{MIBs})$, $p(\text{IP} = \text{abnormal} | \text{MIBs})$, and $p(\text{UDP} = \text{abnormal} | \text{MIBs})$.

Figures 5a through 5d show these, respectively. The asterisks mark the file server’s downtime. As the figures show, anomalies are present before the problem in all three network functions (an anomaly was any abnormal probability of more than 0.5), but only IP (Figure 5b) detects the anomaly during the crash. By combining information from the three MIB variables, the agent was able to detect the abnormal behavior approximately 12 minutes before the server was reported unreachable. We obtained similar results when we used a four-hour learning window, which means that detection was not very sensitive to window size once we arrived at a reasonable size.

Moreover, although the file server being down was not a router problem, the intelligent agent at the router was able to detect that there was an anomaly because of changes in its MIB variables. Therefore, the results are from the router’s view of the network.

We compared our results to those we obtained with a feature set of upper and lower thresholds. We calculated the thresholds by adding or subtracting the (sample) variance from the sample mean calculated from the data within a learning window. If a sample point fell between the lower and upper thresholds, it corresponded to the normal situation; otherwise, it corresponded to an abnormal situation. To combine the information from each MIB variable, we counted the total number of variables that exceeded their

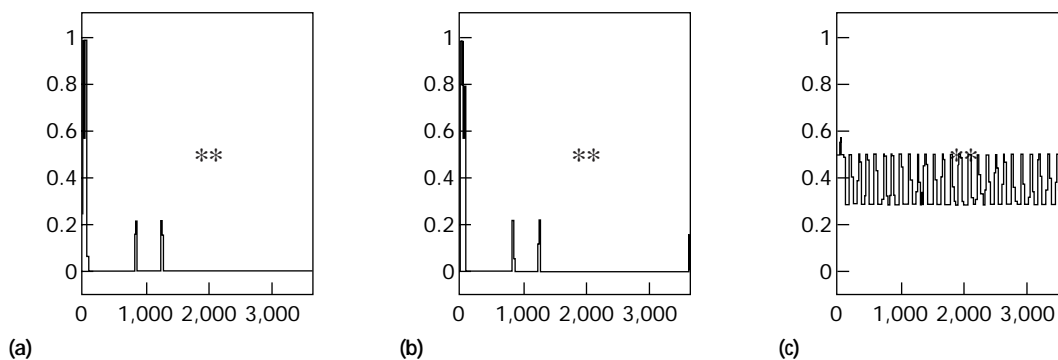


Figure 6. Results from comparing the intelligent agent with the use of thresholds: (a) intelligent agent with a one hour learning window; (b) the intelligent agent with a four-hour learning window; and (c) upper and lower thresholds with a one week learning window. The horizontal axes denote time in seconds, the vertical axes denote the probability that all variables exceed their thresholds at each instance, and the asterisks denote downtime.

thresholds at each instance using thresholds from three learning windows: one hour, four hours, and a week. Figure 6 shows the results of this comparison. Figures 6a and 6b represent results with the intelligent agent for the one-hour and four-hour windows, respectively. Figure 6c represents the results using the upper and lower thresholds. The one-week interval reflects the common practice of determining threshold levels using a large amount of data. The asterisks denote the period where the file server was not responding.

As the figure shows, the results for the one- and four-hour learning windows are almost identical. Both have small peaks where thresholds have been exceeded by three of the 14 MIB variables. Because the corresponding probabilities are much smaller than 0.5, these results may not be sufficient to clearly indicate a problem. The results using a one-week learning window are essentially the same for the entire hour, thereby providing no useful information. This is an intrinsic limitation of threshold methods.

Composite Results

The intelligent agent detected seven of the 10 faults studied (all the faults, not just the “server not responding” faults) using a one-hour learning window, and five of the 10 faults using a four-hour learning window. We considered a fault detected if the posterior probability of abnormal network variable behavior was greater than 0.5.

These results need to be put into perspective. Ideally, we could calculate the number of times a fault is detected when there is no fault (false alarms). However, because we had labels from the syslog file only for the severe faults, we did not know what other faults should or should not be detected. Therefore, to get some measure of sensitivity to characterize the false alarm rate, we calculated the percentage of time the posterior probability of the Network, IF, IP, and

UDP variables are abnormal. Table 1 gives these results. The percentages reflect what would have been false alarms had all faults been reported through syslog messages.

As the table shows, abnormalities at the agent’s highest level (Network) are being detected only a small percentage of the time. This means fault detection at the lower levels is significant. On the other hand, the faults not detected may not have had symptoms present at the router, or the features we used may not have captured the information needed to detect the symptoms that were present.

The results in the table indicate a larger than desired number of false alarms. We need to further investigate feature extraction to reduce the false-alarm rate, yet still maintain accurate detections.

FUTURE WORK

Our work on the intelligent agents can be viewed as a first step toward automated fault management using intelligent agents. One direction of our future work is to improve the agent’s performance. To that end, we plan to further investigate the feasibility and limitations of AR parameters, as well as the combining schemes. For example, more sophisticated learning methods are available to deal with the unob-

Table 1. Percentage of time the posterior probability of Network, IF, IP, and UDP variables are abnormal.

	One-hour learning window	Four-hour learning window
Network abnormal	6.29%	3.85%
IF abnormal	35.97%	24.85%
IP abnormal	42.03%	33.32%
UDP abnormal	42.24%	30.29%

servable variables.⁸ These methods increase computational complexity, something we did not want in our initial investigation. We will also study in more depth how to better incorporate the prior knowledge available at a network node. These efforts should reduce the number of false alarms.

We would also like to expand the scope of our current work. The intelligent agent can be generalized to different types of network nodes with minimal network-specific information required beforehand. This makes the intelligent agents a feasible mechanism for managing heterogeneous networks. We could extend the Bayesian formulation of the problem to combine observations of several intelligent agents from multiple network nodes in the network, thereby tackling the problem of fault detection in large networks. ■

ACKNOWLEDGMENTS

We are indebted to Dave Hollinger, Roddy Collins, and Nathan Schimke for their generous help with the data collection. We also thank John Miller, Ken Vastola, and George Nagy for helpful discussions and suggestions for related references. Finally, we thank the anonymous reviewers for their helpful comments. We gratefully acknowledge the support of the National Science Foundation (ECS-9312594 and (Career) IRI-9502518). This work was presented in part at the 1997 IEEE Information Communication Conference (Infocom 97) and the 1997 International Symposium on Network Management (IM 97).

REFERENCES

1. Y. Yemini, "A Critical Survey of Network Management Protocol Standards," in *Telecommunications Network Management into the 21st Century*, S. Aidarous and T. Plevyak, eds, IEEE Press, Piscataway, N.J., 1994.
2. A. Lazar, W. Wang, and R. Deng, "Models and Algorithms for Network Fault Detection and Identification: A Review," *Proc. Int'l Conf. Communications*, IEEE CS Press, Los Alamitos, Calif., 1992.
3. R. Maxion and F. Feather, "A Case Study of Ethernet Anomalies in a Distributed Computing Environment," *IEEE Trans. Reliability*, Vol. 39, Oct. 1990, pp. 433-443.
4. U. Appel and A. Brandt, "Adaptive Sequential Segmentation of Piecewise Stationary Time Series," *Information Sciences*, Vol. 29, 1983, pp. 27-56.
5. C. Hood, "Intelligent Detection for Fault Management of Communication Networks," doctoral dissertation, Rensselaer Polytechnic Institute, Troy, N.Y., 1997.
6. P. Smythe, "Markov Monitoring with Unknown States," *IEEE J. Selected Areas in Communications*, Vol. 12, 1994, pp. 1,600-1,612.
7. C. Hood and C. Ji, "Proactive Network Fault Detection," *Proc. Int'l Conf. Communications*, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 1,149-1,157.
8. D. Heckerman, "A Tractable Algorithm for Diagnosing Multiple Diseases," *Proc. Workshop Uncertainty in Artificial Intelligence*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 174-181.
9. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, Calif., 1988.

Cynthia S. Hood is an assistant professor of computer science and engineering at the Illinois Institute of Technology. Her research interests include fault management, intelligent agents, intrusion detection, and network reliability. At the time she did the work reported in this article, she was at Rensselaer Polytechnic Institute. Hood received a BS in computer and systems engineering from RPI, a Master of Engineering from Stevens Institute of Technology, and a PhD in computer and system engineering from RPI. She is a member of the IEEE.

Chuanyi Ji is an assistant professor of electrical, computer and systems engineering at Rensselaer Polytechnic Institute. Her research interests include stochastic modeling and analysis of multimedia traffic, network management, and adaptive learning systems. Ji received a BS from Tsinghua University, an MS from the University of Pennsylvania, and a PhD from the California Institute of Technology—all in electrical engineering. She is a member of the IEEE and a recipient of the NSF's Career award (1995).



CALL FOR
SUBMISSIONS

DIGITAL LIBRARIES: TECHNOLOGICAL ADVANCEMENTS AND SOCIAL IMPACTS

Submissions Due: May 15
Acceptance By: November 1
Publication Date: February 1999

Specific areas of interest include:

- ❖ Large-scale projects to develop real-world electronic testbeds for actual users and that aim to develop new, user-friendly technology for digital libraries.
- ❖ Research projects that examine the broad social, economic, legal, ethical, and cross-cultural contexts and impacts of digital library research.

Guest Editors:

Hsinchun Chen
University of Arizona
hchen@bpa.arizona.edu

Bruce R. Schatz
University of Illinois
bschatz@ncsa.uiuc.edu

For more information, see the detailed author guidelines at
<http://computer.org/computer>

COMPUTER
Innovative technology for computer professionals