Adaptive Thresholding for Proactive Network Problem Detection

Marina Thottan and Chuanyi Ji
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute, Troy, NY 12180
e-mail: thottm@ecse.rpi.edu, chuanyi@ecse.rpi.edu

Abstract—
The detection of network fault scenarios has been achieved using the statistical information contained in the Management Information Base (MIB) variables. An appropriate subset of MIB variables was chosen in order to adequately describe the function of the node. The time series data obtained from these variables was analyzed using a sequential Generalized Likelihood Ratio (GLR) test. The GLR test was used to detect the change points in the behavior of the variables. Using a binary hypothesis test, variable level alarms were generated based on the magnitude of the detected changes as compared to the normal situation. These alarms were combined using a duration filter resulting in a set of node level alarms, which correlated with the experimentally observed network faults and performance problems. The algorithm has been tested on real network data. The applicability of our algorithm to a heterogeneous node was confirmed by using the MIB data from a second node. Interestingly, for most of the faults studied, detection occurs in advance of the fault (at least 5 min) and the algorithm is simple enough for potential online implementation; thus allowing the possibility of prediction and recovery in the future.

I. Introduction

Fault management is one of the most importanat aspects of network management. Fault situations could arise from defective components, their transient failure, software failures or from operational errors [8]. As the demand for and dependence on communication networks increases, the variety in network devices and software also increases. Along with the variety come the problems of compatibility and coexistence of the network's different pieces. These problems make fault management a critical part of providing network reliability and Quality of Service (QOS) guarantees. The goal of this work is to detect potential network problems through network traffic measurements. Specifically, for the Internet we use the Management Information Base (MIB) variables [17].

Several network management software packages are commercially available; however these at best can only detect severe failures or performance issues such as a broken link or a loss of link capacity [18]. These methods do not capture the subtle changes in network traffic that are indicative of many common network problems including file server crashes. Rule based methods have also been developed to detect certain subsets of faults. Unfortunately these methods require a data base of fault scenarios and rules for detection which often rely heavily on the expertise of the network manager . The rules thus developed are too specific to characterize all network fault scenarios that evolve with time. Thus most schemes based on Artificial Intelligence suffer from being dependent on prior knowledge about the fault conditions on the network and the rules developed do not adapt well to a changing network environment [9]. Research in the field of fault management is primarily focussed on fault localization by alarm correlation [20][14]. Fault localization is usually based on the assumption that the alarms provided by the network nodes are true and the relevance of temporal information in the alarms is ignored or assumed accurate. The issue of temporal information has been partially addressed [2] by introducing time correlated MIB variables TMIB. However, to the best of our knowledge the generation of time correlated alarms related to a fault at a network node and related issues is still an open problem. The open problems we tackled in this work are the generation of time correlated proactive alarms using a modification of the adaptive thresholding scheme on the MIB variables [11], choice of an optimal set of variables, study of the statistical properties of the MIB variables, and the generalization capability of the algorithm to another node.

The potential use of MIB variables in proactive fault management has been explored previously in [12][13]. In the present work a new, simpler fault detection algorithm has been developed which takes advantage of the interrelationships between MIB variables through a simple combination scheme, generating time correlated node level alarms. An appropriate subset of MIB variables was chosen based on their relevance to the network traffic. The alarms were generated proactively, i.e., before the fault actually occured (refer section VIII). The algorithm has been successfully implemented on a real network at two different nodes suggesting that it is not specific to the properties of a given node. The problem of fault characterization was avoided by utilizing the statistical properties of the signal (specifically using the AR parameters). The simplicity of the algorithm and its adaptibility to topological changes in the network make it an attractive approach for online implementation. Since the processing is done locally, the method is expected to lend itself easily to a distributed implementation [10].

An adaptive GLR test was used to detect changes that occur in the traffic data collected from the network. The MIB variables represent a cross section of this traffic at the different layers of the protocol stack that defines the network. The changes detected in the signal (MIB variables) at the different protocol layers were then correlated so as to reflect the propagation of the change through the protocol stack. This correlated information was then used to declare an alarm at the node level. With very high probability the alarms obtained were found to be indicative of an impending fault. The methods developed are general to enable implementation

on any network environment.

II. Model

The network management system we developed is based on a network management protocol working in a client - server paradigm. The network manager is the client and the agent providing the data is the server. The network used was the Internet and the associated management protocol was SNMP [21]. The agents were different entitites on the network that are capable of collecting MIB variables. The SNMP provides an organized structure to these variables (MIB) as well as a mechanism for communication. However, these variables by themselves were not sufficient to detect faults. The variables had to be further processed by a management application. The node level alarm generation system consisted of three

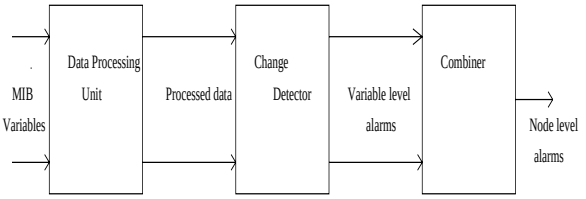

Fig. 1. **System model**

stages. These stages are depicted in Figure 1. The data processing unit divided the time series into piecewise stationary segments. The piecewise stationary segments of the data were modelled using an AR (Auto Regressive) process. The data from the processing unit was analysed by the change detector which determined the generalized likelihood ratio. A sequential hypothesis test based on the power of the residual signals in the time segments was performed to determine if a change had occurred. Each of these change points were then compared to the first two moments of the normal data. An alarm was raised at the variable level if the first two moments of the time series were significantly different from the normal data. The details of this approach are explained in Section IV. The change detector produced a series of alarms corresponding to changes observed in each of the individual MIB variables. These alarms were candidate points for fault occurrences.

In the next stage the variable level alarms were combined using the a priori information about the relationships between these MIB variables. This aspect is further discussed in Section IV. The time correlated alarms corresponding to the faults were obtained as the output of the combiner.

III. Choice of Variables

Choosing a subset of relevant MIB variables is an important first step towards developing a network node agent. Such a step is necessary to reduce the complexity of the problem and render it more tractable. This is feasible since a lot of the MIB variables are redundant in terms of fault detection.

The Management Information Base maintains 171 variables [19]. These variables fall into the following groups: System, Interfaces, Address Translation (at), Internet Protocol(ip), Internet Control Message Protocol ($icmp$), Transmission Control Protocol (tcp), User Datagram Protocol (udp), Exterior Gateway Protocol (egp), and Simple Network Management Protocol ($snmp$). Each group of variables describes a specific functionality of the network. Depending on the type of node monitored an appropriate group of variables was considered. The *if* group of variables and the *ip* group of variables describe the traffic characteristics at a particular interface and at the network level respectively. Since the *if* and *ip* variables sufficiently describe the functionality of the two nodes (router and gateway) under consideration we investigated only the *if* and *ip* groups.

Within a particular MIB group there exists some redundancy. For example consider the variables interface Out Unicast packets $ifOU$, interface Out Non Unicast packets $ifONU$ and interface Out Octets $ifOO$. The $ifOO$ variable contains the same traffic information as that obtained using both $ifOU$ and $ifONU$. In order to simplify the problem, redundant variables were not incorporated. Some of the variables, by virtue of their standard definition [17], were not relevant towards the detection of a fault and therefore excluded. The relationships between the MIB variables of a particular group can be represented using a Case diagram [6]. The Case diagram for the *if* and *ip* variables are shown in figure 2. The arrows in the Case diagram show the direction of traffic flow from the lower to the upper network layers. From this diagram it is clear that the variables depicted in the figure by a bold line are not redundant and represent cross sections of the traffic at different points in the protocol stack. The variables chosen correspond to what are called filter counters [6]. A filter counter is a MIB variable that measures the level of traffic at the input and at the output of each layer. Using these principles for determining non-redundant variables, six MIB variables were selected for our algorithm. In the *if* layer, the variables $ifIO$ (In Octets) and $ifOO$ (Out Octets) were used to describe the characteristics of the traffic going into and out of that interface from the router. Similarly in the *ip* layer four variables were used. The variable $ipIR$ (In Receives), represents the total number of datagrams received from all interfaces of the router. $ipIDe$ (In Delivers), represents the number of datagrams correctly delivered to the higher layers as this node was their final destination. $ipFD$ (Forward Datagrams), represents the number of datagrams that were successfully forwarded to the next node. $ipOR$ (Out Requests), represents the number of datagrams passed on from the higher layers of the node to be forwarded by the ip layer. As described above the Case diagrams can be used to choose non-redundant MIB variables in any protocol layer.

The six MIB variables chosen are not strictly independent. However the relationships between these variables are not obvious and depend on parameters of the traffic such as source and destination of the packet, processing speed of the agent and the actual implementation of the protocol. Some of these dependencies have been incorporated at the combination stage .

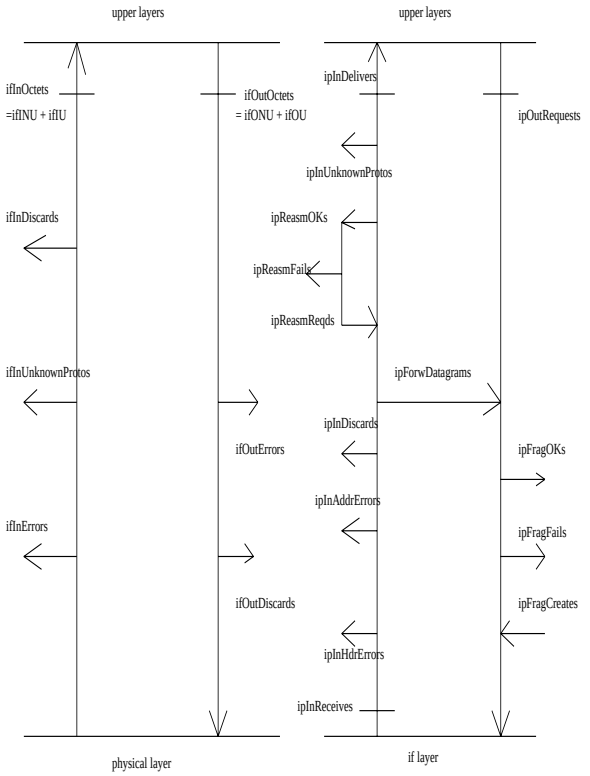

Fig. 2. **Case diagram for the *if* and *ip* variables**

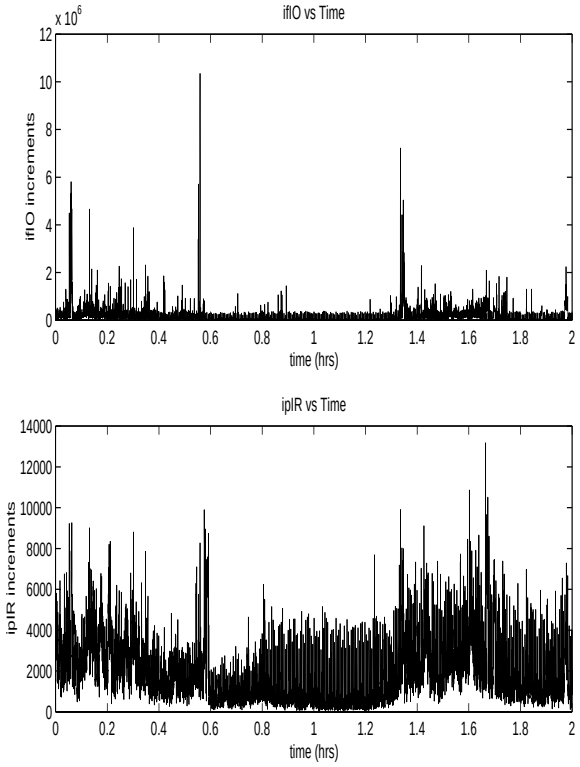

Fig. 3. **Representative trace of *if* and *ip* variables**

IV. Alarm Generation

Once the appropriate set of MIB varaibles were chosen, variable level alarms were obtained using a change detection algorithm. The detection algorithm employed at the variable level was based on the premise that the statistical properties of the MIB variables change in response to fault conditions. It has been experimentally shown that changes in the statistics of traffic data can be used to detect faults [8][18][16]. The detection algorithm was implemented independently on each MIB variable. The increments in the MIB counters were obtained every 15 seconds and the data thus generated constituted a time series. Representative time series data from both the *ip* and the *if* layers are shown in Figure 3. These two data traces reveal each variables' behavior over a two hour period. Note that the data exhibits a high degree of nonstationarity. Piecewise stationary Auto-Regressive models have been used to successfully describe such nonstationary stochastic time series signals [4]. Thus the MIB data was divided into 2.5 minute (10 time lags) piecewise stationary windows. The 2.5 minute window was chosen to avoid dependencies between two adjacent time segments (refer Section V). Within a time window of size N (N=10), the MIB data was linearly modelled using a first-order AR process. Using these piecewise stationary windows a sequential hypothesis test was performed using the GLR (Generalized Likelihood Ratio) test [3] [7]. Non overlapping windows were used in order to obtain less correlated residuals. The detection algorithm consists of two steps: (1) detecting changes within the time series and (2) determining whether a change corresponds to an abnormal situation.

A. Derivation of the Generalized Likelihood Ratio Test

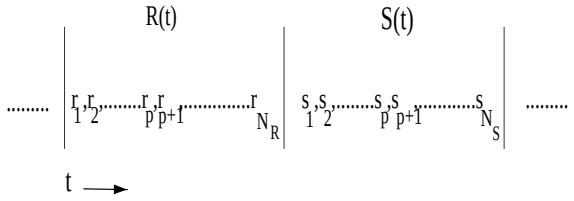

Fig. 4. **Piecewise stationary segments**

To detect changes that occur within a time series we use the Generalized Likelihood Ratio (GLR) test. For a given MIB variable consider two adjacent time windows R(t) and S(t) of lengths N_R and N_S respectively as in figure 4 ($N_R = N_S = 10$). Let us first consider R(t):

$$R(t) = \{r_1(t), r_2(t), ..., r_{N_R}(t)\} \quad (1)$$

Here we can express any $r_i(t)$ as $\tilde{r}_i(t)$, where $\tilde{r}_i(t) = r_i(t) - \mu$ and μ is the mean of the segment R(t). Now $\tilde{r}_i(t)$ can be modelled as an AR order p process (p=1) with a residual error ϵ_i.

$$\epsilon_i(t) = \sum_{k=0}^{p} \alpha_k \tilde{r}_i(t-k)$$

where $\alpha_R = \{\alpha_1, \alpha_2, ..., \alpha_p\}$ are the AR parameters and $\epsilon_i(t)$ is assumed to be white noise. From Equation(2) and assuming

that each sample of the residual error $\epsilon_i(t)$ is drawn from an $N(0, \sigma_R^2)$ distribution [3][15], the joint likelihood of the residual time series was obtained as

$$p(\epsilon_{p+1}, ..., \epsilon_{N_R}/\alpha_1, ..., \alpha_p) = \left(\frac{1}{\sqrt{2\pi\sigma_R^2}}\right)^{\acute{N_R}} exp\left(\frac{-\acute{N_R}\hat{\sigma}_R^2}{2\sigma_R^2}\right), \quad (2)$$

where σ_R^2 is the variance of the residual in segment R(t), and $\acute{N_R} = N_R - p$, and $\hat{\sigma}_R^2$ is the covariance estimate of σ_R^2 [7]. We can obtain a similar expression for the segment S(t). Now the joint likelihood l of the residual errors in the two segments R(t) and S(t) is given as,

$$l = \left(\frac{1}{\sqrt{2\pi\sigma_R^2}}\right)^{\acute{N_R}} \left(\frac{1}{\sqrt{2\pi\sigma_S^2}}\right)^{\acute{N_S}} exp\left(\frac{-\acute{N_R}\hat{\sigma}_R^2}{2\sigma_R^2}\right) exp\left(\frac{-\acute{N_S}\hat{\sigma}_S^2}{2\sigma_S^2}\right), \quad (3)$$

where σ_S^2 is the variance of the residual in the segment S_t and $\acute{N_S} = N_S - p$. The expression for l is a sufficient statistic and is used to perform a binary hypothesis test. The two hypotheses are H_0, implying that no change is observed between segments, and H_1, implying that a change is observed. Under the hypothesis H_0 we have,

$$\alpha_R = \alpha_S$$
$$\sigma_R^2 = \sigma_S^2 = \sigma_P^2$$

where σ_P^2 is the pooled variance. Under hypothesis H_1 , we have,

$$\alpha_R \neq \alpha_S$$
$$\sigma_R^2 \neq \sigma_S^2.$$

Using the conditions in Equations(4 and 4) we obtained the likelihood ratio as,

$$\lambda = \sigma_P^{-(\acute{N_R}+\acute{N_S})} \sigma_R^{\acute{N_R}} \sigma_S^{\acute{N_S}} * exp\left(\frac{-\hat{\sigma}_P^2(\acute{N_R}+\acute{N_S})}{2\sigma_P^2} + \frac{1}{2}\left[\frac{\acute{N_R}\hat{\sigma}_R^2}{\sigma_R^2} + \frac{\acute{N_S}\hat{\sigma}_S^2}{\sigma_S^2}\right]\right). \quad (4)$$

Furthermore on using the maximum likelihood estimates for the variance terms, we get the log likelihood ratio to be,

$$-\ln\lambda = \acute{N_R}(\ln\hat{\sigma}_P - \ln\hat{\sigma}_R) + \acute{N_S}(\ln\hat{\sigma}_P - \ln\hat{\sigma}_S). \quad (5)$$

The log likelihood ratio $-\ln\lambda$ is compared with an optimally chosen threshold h (described in Section VI). Segment boundaries where the threshold was exceeded were considered to be change points. That is,

$$-\ln\lambda \quad > \quad h \Longrightarrow H_1$$
$$\leq \quad h \Longrightarrow H_0.$$

$\ln\lambda$ can also be interpreted as a measure of the information loss incurred by accepting the hypothesis H_0 [5].

B. Generation of Alarms at Variable Level

Once a change is detected a second hypothesis test was conducted to determine whether a specific change point detected by the first hypothesis test corresponds to an abnormal situation. In the second hypothesis test the mean μ_P and variance σ_P of the pooled segment (R(t) and S(t)) were compared with the normal mean μ_0 and variance σ_0 using a likelihood ratio and a second threshold η. The normal mean and variance were computed from the normal data over a twenty four hour period. The two hypotheses considered were H_0 with a distribution of $N(\mu_0, \sigma_0^2)$ implying that the change was normal and H_1 with a distribution of $N(\mu_P, \sigma_P^2)$ implying an abnormal change had occurred. Abnormal changes were noted as variable level alarms. Our algorithm depends on the following parameters: (1) the order p of the AR process, (2) the threshold values h, and η, and (3) the normal mean μ_0 and variance σ_0. The choice of these parameters are discussed in Section VI.

C. Combination

The variable level alarms obtained were combined using a duration filter. The duration filter [8] was implemented on the premise that a change observed in a particular variable would propagate into another variable that was higher up in the protocol stack. For example, in the case of the *ifIO* variable, the flow of traffic is towards the *ipIR* variable. Using the relationships from the Case diagram representation shown in Figure 2, the possible transitions between the chosen variables were determined (see Figure 5). The duration filter

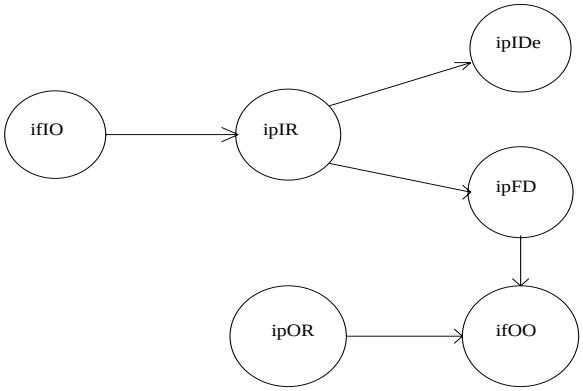

Fig. 5. **Transitions between MIB variables**

was designed to detect all five transition types.

V. Study of the Statistics of the Residual Error ϵ_t

The network traffic has been shown to exhibit long range dependence [23]. Therefore, it is necessary to explore the time lagged properties of the residuals of the piecewise stationary segments. The correlation function of a typical residual signal is shown in Figure 6. The correlogram was obtained over 25 time lags (approx. 12mins). Each time lag corresponds to 15 seconds. Note that there is no significant correlations after 10 lags (2.5mins). In our algorithm the hypothesis test is

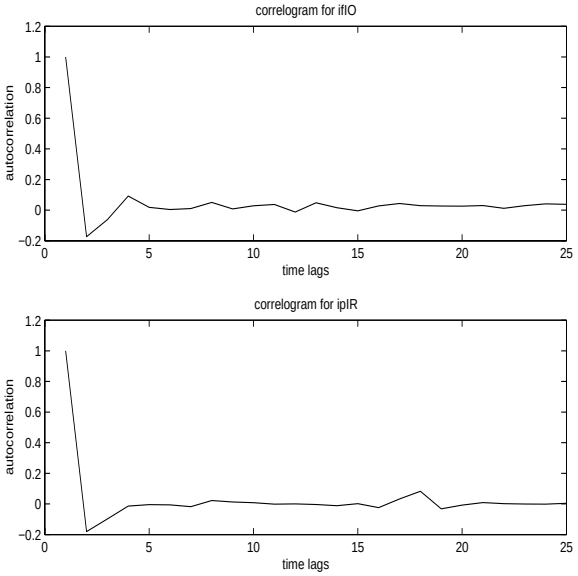

Fig. 6. **Auto correlation of residuals**

performed over two adjacent non-overlapping time windows of length 10 lags. Thus we were able to obtain uncorrelated residuals between the windows. However, note that the residuals may be more correlated if sampling was done at higher frequencies. This effect is due to the long range dependence in the signal.

Furthermore, the assumption that the residuals are Gaussian was studied using the marginal distributions of the residuals. The quantile distribution of the residuals of two of the MIB variable are plotted against the quantiles of a standard Normal distribution in Figures 7 and 8. From this figure it is clear that the distribution can be approximated by a Gaussian form. However, the 'S' shape of the curve shows that the residuals have a longer tail than the standard Normal. As seen from the figure the *ip* variables can be better approximated as Gaussian random variables than the *if* variables, since *ip* traffic can be regarded as a *sum* of the traffic from the different interfaces. However, using the Normal approximation for the residual error distribution is reasonable as we are only concerned with the first two moments and the magnitude of ϵ_t is moderate.

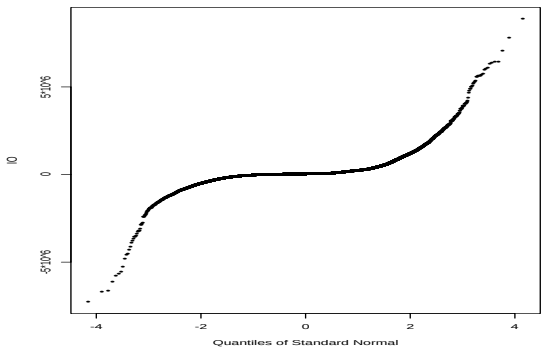

Fig. 7. **Quantile - quantile plot of ifIO residuals**

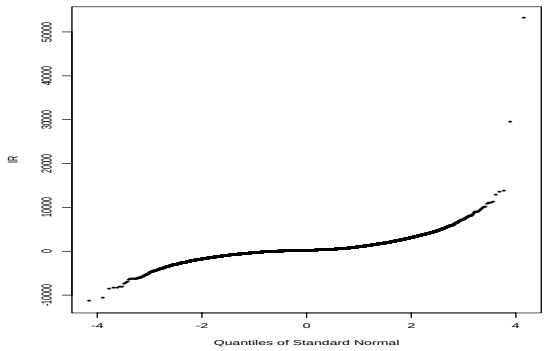

Fig. 8. **Quantile - quantile plot of ipIR residuals**

VI. Implementation Issues

The implementation of our algorithm depends on the choice of the following parameters.

AR order (p): Adequate representation of the signal and parsimonious modelling [4] were competing requirements; hence a trade off between these two issues was necessary. The accuracy of the model was measured in terms of Akaike's final prediction error (FPE) criterion [1]. The order corresponding to a minimum prediction error was the one that best modelled the signal. However due to singularity issues there was a constraint on the order p [3], expressed as:

$$0 \leq p \leq 0.1N \tag{6}$$

where N is the length of the sample window. Furthermore, if we wish to detect a change segment of length Q or longer, then the sample window size has an upper limit [3],

$$N \leq 0.7Q \tag{7}$$

In our algorithm we set Q=15 time lags (3.75 min, approx duration of faults, see table II) and N=10 time lags (2.5 mins). Using these values for N and Q and subject to the constraint Equations (6) and (7) we found the only appropriate order for p that minimizes the FPE for each of the MIB variables to be 1.

Threshold h and η: The threshold h was optimised experimentally on data sets 3 and 6. The thresholds were set to ensure that maximum number of faults were detected at very low false alarm rate. The same values were used on all the other data sets, both as h in the first stage and as η in the second stage. The optimised values (h=(5 to 20)) worked well on the other data sets (refer Section VIII). Since the *ip* variables represent traffic at a lower resolution (in units of datagrams), and the *if* variables at a higher resolution (in units of octets), the thresholds used for the *ip* variables were half that of the *if* variables.

Normal mean and variance: The normal values of mean μ_0 and variance σ_0 used in stage 2 of the detection algorithm were computed over a 24 hour period for each variable. These values were obtained from data sets 3 and 6 and they worked well for the remaining data sets (refer Section VIII).

VII. Experimental Network

The algorithm developed in this work is sufficiently general to allow implementation on any type of network. As an example, in this work we show the application to the Internet. The experiments were conducted on the Local Area Network (LAN) of the Computer Science (CS) Department at Rensselaer Polytechnic Institute. The network topology is as shown in Figure 9. The CS network forms one subnet of the main campus network. The network implements the IEEE 802.3 standard. Within the CS network there are seven smaller subnets and two routers. All of the subnets use some form of CSMA (Carrier Sense Multiple Access) for transmission. The routers implement a version of the Dijkstra's algorithm [22]. One router (shown as Node 1 in Figure 9) is used for internal routing and the other serves mainly as a gateway (shown as Node 2) to the campus backbone. The external router or

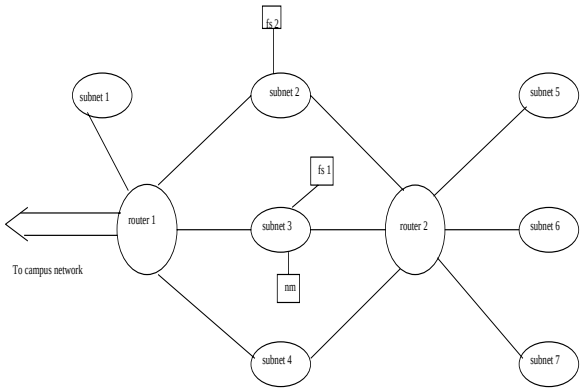

Fig. 9. **Configuration of the monitored network**

gateway also provides some limited amount of internal routing. The internal router has 6 interfaces and the external router has 5 interfaces with the CS subnets. The back bone is an FDDI (Fiber Distributed Data Interface) ring. There are 133 host addresses and 2 file servers. The majority of the host machines are SPARC stations and there are also a few Ultra SPARCs and PCs. The network spans 2 buildings. It is a well designed network in the sense that there are no recurrent problems and the utilization is within the limitations of the bandwidth. The majority of the traffic is file transfers or web traffic which involves the workstations accessing the file servers. The speed of the network is 10Mb/s. The external gateway and the internal router are SNMP agents. The Management Information Base on these agents were polled (using a PERL script) every 15 seconds to obtain the measurement variables. Data was collected from the interface of the subnet 2 with both the routers and at the routers themselves. Queries to the MIB were sent out from a machine (denoted as nm in Figure 9) on subnet 3. As there were no dedicated machines for data collection purposes, the issues of storage space and the impact of polling on the network traffic dictated the choice of the polling frequency [2].

There were no network management measures in place on this network. Each machine on the network ran the UNIX syslog function. This function generated messages that related to problems associated with the network, applications,

TABLE I

Description of fault Data Sets: fault location and time

Data set no	No of faults in data set	Fault location (subnet)	Time and/ duration of faults
1	1	2	4.19 - 4.21pm
2	1	2	11.10 - 11.17am
3	3	2	8.23 - 8.26pm
		2	1.23 - 1.25am
		2	9.48 - 9.52am
4	1	2	6.33 - 6.36am
5	1	2	9.36 - 9.41pm
6	2	3	11.17 - 11.21pm
			11.28 - 11.30pm
		3	3.22 - 3.26pm

TABLE II

Number of machines affected by the faults

Data set no	No of machines affected on subnet 2	No of machines affected on subnet 3	No of machines affected outside subnets 2 and 3
1	8	6	18
2	12	8	16
3	5	6	8
	2	0	0
	9	6	9
4	7	4	2
5	8	5	5
6	1	5	1
	1	8	2

or the specific machine itself. These syslog messages were used to identify the network problems. One of the most common network problems was NFS server not responding. The syslog messages only reported that the file server was not responding, but was unable to identify the cause of the problem. Possible reasons for this problem are of unavailibility of network path or that the server was down. Although not all problems could be associated with syslog messages, those problems which were identified by syslog messages were accurately correlated with fault incidents. A description of the data sets used is provided in the table I and II.

VIII. Results and Discussions

The alarm generation system was implemented on both nodes (Routers 1 and 2 in fig 9). The results obtained at the different stages of the algorithm are described below.

A. Results at the Variable Level

The average number of alarms generated per hour on each variable was computed over all the data sets of Node 1. The results of the variable level alarm generation are summarised

TABLE III
SUMMARY OF RESULTS FOR ALARMS GENERATED AT VARIABLE LEVEL FOR NODE 1

MIB variable	avg no of alarms per hour	no: of faults detected
ifIO	0.91	4/9
ifOO	1.14	5/9
ipIR	3.97	7/9
ipFD	3.98	7/9
ipIDe	2.08	7/9
ipOR	1.19	9/9

in Table III. Representative outputs of stage 1 are shown in Figure 10. All of the variables showed the ability to detect

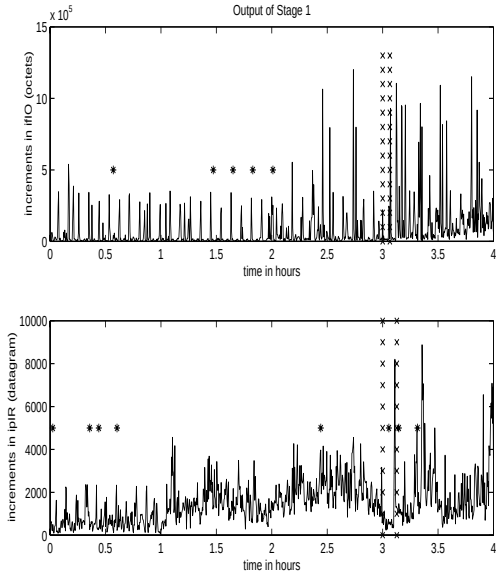

Fig. 10. **Alarms at the variable level of node 1**
" * " alarms, " x " down time period

changes associated with fault scenarios. However, not all the variables were capable of detecting all the faults. The *ip* level variables detect more faults than the *if* variables, suggesting that at the router level the *ip* variables alone are sufficient for fault detection. But in order for the router to detect problems which occur locally within a subnet, it is necessary to know the total throughput to and from that interface (represented by the *ifIO* amd *ifOO*).

B. Results at the Node Level

A combination filter was used to reduce the node level alarms as compared to the variable level alarms. Using the duration filter we were able to reduce the average number of alarms per hour to 1.4. The results after the combination stage on both Nodes 1 and 2 are shown in Tables IV and V. The performance of the algorithm is discussed in terms of the probability of detection P_D and the probability of false alarm P_F [8]. The values for the P_D and the P_F are estimated using

TABLE IV
SUMMARY OF FAULTS DETECTED BY NODE 1
(* DETECTED AFTER THE FAULT)

data set no:	P_D	recent proactive alarm in min	P_F	avg no: of alarms per hour
1	1	40	0.0037	0.89
2	1	8.7*	0.0038	0.91
3	1	31.4 59.8 59.4	0.0071	1.74
4	1	18.4	0.0060	1.45
5	0		0.0065	1.55
6	0.5	31	0.0076	1.83

Equations (8) and (9).

$$P_D = \frac{totalnumberofcorrectmatches}{totalnumberofknownfaults} \tag{8}$$

$$P_F = \frac{totalnumberoffalsealarms}{totalnumberofdatasamples} \tag{9}$$

All alarms generated within one hour before or 15 minutes after the fault occurred were considered to be true alarms. By recent proactive alarm, we refer to the alarm time that was closest to the actual fault. However, in some cases the alarms were more frequent. Note that seven of the nine faults were detected. In six out of seven faults detected, the alarms were generated before the fault actually occurred. Despite the lack of information from the *if* variables of subnet 3 (data set 6) our algorithm was able to detect one of the two faults on that subnet. Since these faults were reported by only 2 or 3 machines outside subnet 3, we did not expect a significant effect on the *ip* variables. The detection of the first fault was attributed to its repeated occurrence.

C. Generalization of the Algorithm to a Second Node

The ability of the algorithm to generalize to a second node was also studied. The results obtained for the second node are summarized in Table V. We were unable to obtain data from this node for data sets 2 and 5. For node 2, six out of the seven faults were detected. The one fault that went undetected was the second fault on data set 6 which occurred on subnet 3 (see above for explanation). Our algorithm was capable of detecting faults that occurred at different times of the day. Representative outputs at the node level of the algorithm are shown in Figure 11.

IX. PERFORMANCE OF THE ALGORITHM

The performance of the algorithm was studied under different sets of thresholds, and the corresponding values for the probability of detection and probability of false alarm are shown in table VI. Thresholds for each of the MIB variables were set to be five above and five below the optimal values. It was observed that as the thresholds were increased the probability of detection increased along with the false alarm rate.

TABLE V

Summary of faults detected by node 2

data set no:	P_D	recent proactive proactive alarm in min	P_F	avg no: of alarms per hour
1	1	43.8	0.0093	2.23
3	1	38.2 22.3 11.2	0.0075	1.86
4	1	8.2	0.0106	2.57
6	0.5	21.6	0.0040	0.96

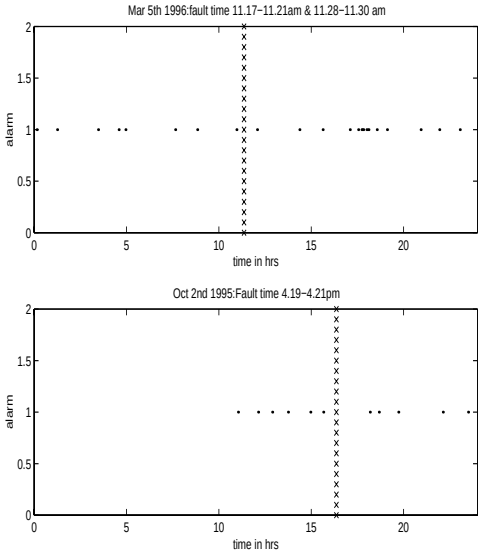

Fig. 11. **Output at node level of node 1**

The algorithm is capable of detecting changes that persist for lengths greater than 15 time lags (3.75 mins). To be able to increase the sensitivity of the algorithm to track even more abrupt changes the length of the sample window N (=10) has to be further reduced. However this would require an AR order $p < 1$. This problem can be circumvented using a higher polling rate.

There was a relatively large number of alarms in the early part of the day. However by using the time of day as an additional feature these alarms can be further reduced. Furthermore, once a fault occurs the system requires time to return to normal. These points were also detected as change points, although they do not necessarily correlate to a fault. Alarms that are generated at these times can be avoided by allowing a renewal time immediately after a faullt has been detected.

TABLE VI

Optimization of thresholds

threshold	P_D	P_F
lower	1	0.0134
optimal	0.75	0.0058
higher	0.305	0.0017

X. Conclusions

This work shows that statistical methods complemented with a good understanding of network protocols can be used to address the problem of network fault detection. The advantages of this method are its independence from specific fault descriptions and ease of implementation. In addition our fault detection scheme is capable of categorising network problems based on the type of transition that detects it. Future efforts will be directed towards diagnosis of faults using this information.

The algorithm is also an improvement over simple thresholding methods since it takes into account the relationships between the measurement variables that reflect the properties of the network. Implementation on a second node suggests the feasibility of our algorithm to larger heterogeneous networks. Efforts are currently under way to increase the specificity of the algorithm through more sophisticated combination schemes and thereby reduce the false alarm rate and remove the need for a heuristic duration filter.

Our algorithm has been able to detect network problems on a fully functional network with complex traffic patterns. Since the network that was considered can be related to any enterprise network both in complexity and in terms topology, our algorithm should find application in these networks.

References

[1] H. Akaike. A new look at statistical model identification. *IEEE trans. on Automatic Control*, 19(6):716–723, Dec 1974.

[2] T.K. Apostolopoulos and V.C. Daskalou. Temporal network management model, concepts and implementation issues. *Computer Communications*, 20:694–708, 1997.

[3] U. Appel and A.V. Brandt. Adaptive sequential segmentation of piecewise stationary time series. *Information Sciences*, 29:27–56, 1983.

[4] Box and Jenkins. *Time Series Analysis, Forecasting and Control.* Holden Day Series, 1976.

[5] A.V. Brandt. An entropy distance measure for segmentation and clustering of time series with application to eeg signals. In *Sixth International Conference on Pattern Recognition Munich*, 1982.

[6] J.D. Case and C. Partridge. Case diagrams: A first approach to diagrammed management information bases. *Computer Communication Review*, 19:13–16, Jan 1989.

[7] P.V. Desouza. Statistical tests and distance measures for lpc coefficients. *IEEE trans on Acoustics,Speech and Signal Processing*, 25(6), Dec 1977.

[8] F. Feather and R. Maxion. Fault detection in an ethernet network using anomaly signature matching. In *ACM SIGCOMM*, volume 23, Sept 1993.

[9] Kormann Franceschi and Westphall. Performance evaluation for proactive network management. In *Proc. IEEE ICC*, 1996.

[10] G. Goldszmidt and Y. Yemini. Distributed management by delegation. In *15th International Conference on Distributed Computing*, 1995.

[11] J. Hellerstein. Personal communications.

[12] C.S. Hood and C. Ji. Proactive network fault detection. *Proceedings of INFOCOM, Kobe, Japan*, 1997. Also available from http://neuron.ecse.rpi.edu/.

[13] C.S. Hood and C. Ji. Intelligent processing agents for network fault detection. *IEEE Internet Computing*, 1998.

[14] I. Katzela and M. Schwarz. Schemes for fault identification in communication networks. *IEEE/ACM Trans.Networking*, 3:753–764, 1995.

[15] H.B. Mann and A. Wald. On the statistical treatment of linear stochastic difference equations. *Econometrica*, 11, 1943.

[16] R. Maxion. A case study of ethernet anomalies in a distributed computing environment. *IEEE transactions on Reliability*, 39(4), Oct 1990.

[17] K. McCloghrie and M. Rose. Management information base for network management of tcp/ip-based internets: Mib 2. *RFC1213*, 1991.

[18] R.E. Moore. Problem detection, isolation and notification in systems network architecture. In *Proc. IEEE INFOCOM*, pages 377–381, 1986.

[19] M.T. Rose. *The Simple Book*. Prentice Hall Series in Innovative Technology, 1991.

[20] I. Rouvellou and G.W. Hart. Automatic alarm correlation for fault identification. In *Proc. IEEE INFOCOM*, pages 553–561, 1995.

[21] W. Stallings. *SNMP, SNMPv2, and CMIP The practical guide to Network Management Standards*. Addison-Wesley Publishing Company, 5 edition, 1994.

[22] A. S. Tanenbaum. *Computer Networks*. Prentice hall, Inc., 3 edition, 1995.

[23] W. Willinger W.E. Leland, M.S. Taqqu and D.V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans.Networking*, 2(1), 1994.