Vol. 44 No. 2 IPSJ Journal Feb. 2003

Regular Paper

Integrated Network Operation Baselining and Adaptive Detection of Faults and Performance Problems

Hassan Hajji[†] and Behrouz H. Far[††]

Detection of network problems is a crucial step in automating network management. It has a direct impact on the accuracy of fault, performance and security management functions. In this paper, we propose an adaptive technique for network fault and performance detection. Network operations normal baseline is parameterized as a finite mixture model, and model parameters are estimated from data using the Expectation Maximization algorithm. We propose a novel method for online residual generation, based on repeated identification of model parameters. The resulting multivariate residuals are shown to be approximately Normal with the mean 0 and unit variance under normal conditions, and the mean jumps suddenly to a different value, once anomaly occurs. To achieve fast detection, we formulate the problem of detection as a sequential disruption problem, where the task is to detect departures from the baseline distribution as soon as the disruption happens. An analytical expression of false alarm rate allows us to choose the threshold, automatically. Experimental results on a real network showed that the monitoring agent is able to detect even slight changes in the characteristics of the network, while maintaining a low false alarm rate.

1. Introduction

Networks and distributed processing systems have become an important substrate of modern information technology. The rapid growth of these systems throughout the workplace has given rise to a discontinuity in expertise of human operators to manage them. There is a need for automating the management functions to reduce network operations and management cost.

Detection of network problems is a crucial step in automating network management. It has a direct impact on the accuracy of fault, performance and security management functions. From a control viewpoint, well designed fault and performance problems detection algorithms enhance the network control capability, by providing timely indication of network incipient problems. The possibility of early detection of performance degradation can alleviate the constant fire-fighting of network managers. Early warnings from the monitoring agent can trigger preventive actions, and serious and expensive outages can be avoided.

Most of the existing research assumed that the alarm generating mechanism is accurate, and network problems are given a priori [6),7),25)]. Current practice in network management rely on user-defined thresholds for detection. Alarms are generated when some variable of interest crosses a predefined threshold. Generally, the predefined value of the threshold is no more than an estimation of the normal range within which the measured feature is believed to operate. Not only there is little objective insights on how to choose these thresholds, but also there is a risk of missing subtle changes in the network state [5)]. In addition, the complexity and size of current network systems makes them vulnerable to novel faults and performance degradation patterns.

The main difficulty of network anomaly detection is the lack of a general definition of what constitutes normal behavior [8)]. The dynamics of the network normal operations need to be identified from routine operation data. To this end,[11)] characterizes the normal behavior by different templates, obtained by taking the standard deviations of observations (typically Ethernet load and packets count), at different operating times. An observation is declared abnormal if it exceeds the upper bound of the envelope. Given the bursty nature of network traffic, the standard deviation estimates are likely to be distorted, making subtle changes in the network state go undetected. To mitigate the effect of the non-stationary nature of network traffic [5)], considered the model formed

† IBM Research, Tokyo Research Laboratory, Japan
†† Department of Electrical and Computer Engineering, University of Calgary, Canada

This research is done while authors at Department of Information and Computer Science, Saitama Univeristy, Japan by segmenting time series obtained form SNMP MIB objects. Observations are declared abnormal if they do not fit an auto-regressive model of the traffic inside segments. In[20] the observation are declared abnormal after a statistical test with the mean of 24-hour period sample. In these approaches, the assumption of piece-wise constancy of the traffic is questionable, since the traffic burst is not sustained long enough to allow accurate estimation, and the combining scheme for correlating different alarms from the objects is ad-hoc.

In this paper, we address the problem of performance problems detection in IP-Networks, where the knowledge about the problems to be detected is not required. The emphasis is on fast detection with minimal human supervision – an important requirement for reducing potential impact of problems on network services users. We propose a model of the network operations in terms of MIB objects dependencies, and we show that the parametric characterization of this dependency can be described as a finite mixture of simple regression models. Model parameters are identified from routine operation data, using the expectation maximization algorithm.

A new method for residual generation, based on successive parameter identification, is introduced. The residuals are shown to be approximately multivariate Normal, with mean zero under normal operations, and sudden jumps in this mean are characteristics of abnormal conditions. The detection problem is formulated as a *change point problem*. A real-time online change detection algorithm is designed to processes, sequentially, the residuals and raise an alarm as soon as the anomaly occurs. We motivate this formulation through a real problem scenario that occurred in Saitama university network. The proposed approach requires neither the set of faults and performance degradation nor the thresholds to be supplied by the user. Experimental results showed the effectiveness of the method on real data. A very low alarm rate and a high detection capability has been demonstrated.

This paper is arranged as follows: Section 2 introduces our proposed model of the network normal behavior, and the learning algorithm for parameter identification from routine operation data. Section 3 introduces our proposed approach for residual generation, and the formulation of the network problem detection. In Section 4, we present results of our experiments in a real network. We conclude in section 6.

Table 1 Object clusters for performance problem detection.

Dependent Variable (Y)	**Independent Variable (X)**
ifInNUcastPkts	
ipInReceives	ifInPkts for all interfaces
ipInDelivers	ipInReceives + ifInPkts (loopback interface)
ipForwDatagrams	ipInReceives - ipInDelivers
ifOutPkts for all interfaces	ipForwDatagrams + ipOutRequests
ifOutNUcastPkts	

2. Normal Operations Baselining

MIB objects are designed to reflect the activity of the network at each protocol layer entity. The goal of this section is to characterize network normal behavior using these objects, and to identify network model parameters from routine operation data.

2.1 Network Model

Our proposed model is to define the network normal behavior model in terms of the relationships between selected MIB objects. That is, instead of studying individual objects, the normal behavior of the network is defined as the parametric characterization of clusters of dependent variables in the interface and network layers. The node's view of the of the network behavior is, then, the aggregation of these models. **Table 1** shows object dependencies at the network and interface level, as can be extracted from the functional requirement of TCP/IP protocol stack, or case diagrams[19].

This approach has three main benefits. First, examining the relationship between dependent variables provides a robust method for detecting network anomalies: it allows the model to interpret individual variable values in conjunction with dependent variable values. For example, the number of interface errors alone is not a clear-cut indication of network problems unless studied in relation to the total amount of inbound traffic[9]. Second, the statistical characteristics of individual variables is nonstationary, and better characterization of normal behavior can be obtained by examining explicitly variables dependency. Segmentation of individual variables, as proposed in Refs. 5), 20), overlooks the effect of variables on each other. Clearly, the segmentation of the variable *ipInReceives*, for instance, could be better done if all incoming packets *ifInPkts* are taken into account. Third, the specific nature of the MIB-II[12] shows that structural change in the dependency between these variables is a symptom of an abnormal behavior. For example, if the relation between *ifInPkts* and *ipInReceives* changes, it is because of errors caused by lack of buffer space, line noise or unknown protocols packets and so on. In this sense, selected objects form a sufficient subset that needs to be continuously monitored. In addition, it is guaranteed that these variables statistics are available across most operating systems kernels. Currently, there is little support for detailed interface error statistics.

Most of operating systems share the same dependency as in Table 1, except for some systems that put loop-back packets directly in the IP input queue, rather than requiring network interface cards to read their own transmissions. We can account for this case by adding the loop-back packets to the *ipInReceives* as shown in Table 1.

2.2 Objects Dependency Parametric Model

To be able to identify the network operation parameters from operation data, we have to define a parametric model for network operations. Our approach to network model parameterization is to view each cluster (X, Y) as switching between different regimes, where each regime is a simple linear regression model. This is a form of what referred to in the literature as *switching regression*[13),15)]. The observations (x_i, y_i) are generated by one of the K linear regression, as shown in the following equations:

$$y_i = b'_k x_i + \epsilon_{ki} \qquad k = 1, \ldots, K \tag{1}$$

$$p(y_i|x_i) = \sum_{i=1}^{K} \frac{\pi_k}{\sqrt{2\pi}\sigma_k} \exp \frac{-(y_i - b'_k x_i)^2}{2\sigma_k^2} \tag{2}$$

For the case of single variables in Table 1, such as *IfInNUcastPkts*, we have:

$$y_i = m_k + \epsilon_{ki} \qquad k = 1, \ldots, K \tag{3}$$

$$p(y_i) = \sum_{i=1}^{K} \frac{\pi_k}{\sqrt{2\pi}\sigma_k} \exp \frac{-(y_i - m_k)^2}{2\sigma_k^2} \tag{4}$$

The errors ϵ_k are assumed to be Gaussian, with mean 0 and variance σ_k. The column vector b_k is made of the slope and the intercept for the regime k. Abusing the notation slightly, the integer K denotes the number of regimes, for both the mixture of regression and Normal distributions. Each regime k has a mixing probability, denoted by π_k.

Finite mixture models for network operation baselining captures both clusters that may be described *parsimoniously* by linear relationships, and more generally non-linear dynamics of any given cluster. In fact, finite Gaussian mixture models are general enough to approximate any continuous function with a finite number of discontinuities, under appropriate regularity conditions[22)]. For any cluster of variables (X, Y) that are linearly related, or even locally linear with slowly time-varying parameters, an adaptive algorithm with suitably chosen *forgetting factor* can track the model parameters. In general, however, breaks in the linear relationship are normal, and the idea then is to explicitly accommodate these breaks in the network model. The resulting model is, then, a mixture of regimes, where each regime describe a given mode of network operations.

The network normal behavior is then characterized by the parameters of the finite mixture model. In the next subsection, we show how these parameters are identified.

2.3 Learning Model Parameters

Identifying the network normal operations from routine operation data amounts to estimate the parameter θ of the switching regression. The vector θ consists of the vectors $b_1, \ldots, b_K$, the variances $\sigma_1, \ldots, \sigma_K$ and the mixing probabilities $\pi_1, \ldots, \pi_K$. Given a training set of N independent and identically distributed data points (x_i, y_i), the Maximum Likelihood (ML) estimator is the vector $\hat{\theta}$ that maximizes the likelihood function $L(\theta)$, given by:

$$\hat{\theta} = \arg\max_{\theta} L(\theta) \tag{5}$$

$$L(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} \pi_k f_{ik} \tag{6}$$

$$f_{ik} = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(\frac{-(y_i - b'_k x_i)^2}{2\sigma_k^2}\right) \tag{7}$$

For the case of mixture of Normal distributions, f_{ik} is given by:

$$f_{ik} = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(\frac{-(y_i - m_k)^2}{2\sigma_k^2}\right) \tag{8}$$

If we note (x, y, z) the complete sample information, and given the values of the separation variable z, that assigns each observation to its corresponding regime, maximum likelihood estimation is straightforward. In the absence of this complete information explicit maximization of the likelihood function is not tractable. However, viewing the separation variable z as missing, the estimation can be formally identified with the Expectation Maximization (EM) algorithm [4]. The EM algorithm [2),16] estimate the unknown parameter θ by iteratively maximizing the expected log likelihood $Q(\theta|\theta')$. Given an initial parameter θ^0, the following two steps are iterated until convergence:

E-Step: Given a current approximation θ^m, determine the expected log likelihood $Q(\theta|\theta^m)$:

$$Q(\theta|\theta^m) = \sum_{i=1}^{N}\sum_{k=1}^{K} w_{ki}\log(f_{ik}) \qquad (9)$$

$$w_{ki} = \frac{\pi_k f_{ik}}{\sum_{k=1}^{K}\pi_k f_{ik}} \qquad (10)$$

M-Step: The new refined estimate θ^{m+1} is the solution of the maximization:

$$\theta^{m+1} = \arg\max_{\theta\in\Theta} Q(\theta|\theta^m) \qquad (11)$$

The EM algorithm has a number of appealing properties. It is numerically stable, in the sense that the incomplete data likelihood is increased at each iteration [2]. Also, since the M-Step relies on the complete data, the maximization in this step is analytically solvable. In fact, in our case the M-Step is simply the weighted least squares, where the weights are w_{ki}, obtained by calculating the posterior probability of each observation with respect to each of the regimes, as shown in Eq. (10). Similar reasoning can be done for the case of finite Normal distributions mixtures.

3. Online Network Problems Detection

The previous section showed how the dynamics of the networks are identified. In this section we turn to discuss how the residuals are generated, and our formulation of the problem of anomaly detection.

3.1 Residual Generation

The residual generation method we propose is based on the algorithm for parameter estimation. After convergence of the learning algorithm (Sec. 2.3), we keep updating the slopes b_k and the means m_k of each regime, assuming that only one iteration of the EM algorithm is used for each new observation (x_n, y_n). It can be shown, that:

$$b_k^n = b_k^{n-1} + \frac{w_{kn}x_n(y_n - b_k^{n-1}x_n)}{\sum_{i=1}^{n} w_{ki}x_i^2} \qquad (12)$$

$$b_k^0 = \hat{b}_k \qquad (13)$$

For the case of mixture of Normal distributions, the mean m_k of each regime k is updated as follows:

$$m_k^n = m_k^n + \frac{w_{kn}(y_n - m_k^{n-1})}{\sum_{i=1}^{n} w_{ki}} \qquad (14)$$

$$m_k^0 = \hat{m}_k \qquad (15)$$

Where b_k^n and m_k^n are the estimate of the slope parameter b_k and mean m_k at iteration n, respectively. X_n is a $n \times 1$ vector made of the regressor x_i, and X_n' is its transpose vector. W_{kn} is the $n\times n$ diagonal matrix, made of the weights w_{ki}. The initial parameters b^0 and μ^0 are the estimates obtained using the batch EM (Sec. 2.3). It is worth nothing to note that Eq. (12) is very similar to the recursive parameter estimation proposed in [21], where parameters are updated online for each new observation. In the sequel, we are interested only in the changes of the parameters b_k, and m_k of the network model.

Under normal conditions, the difference between the successive values b^n and b^{n-1} is expected to fluctuate around zero. This difference should not drift constantly in a fixed direction. On the other hand, if this difference drifts systematically over long duration, then the new observations are generated by a different model, and the recursion in Eq. (12) will alter the parameter b to its new value. The idea, then, is to generate the residuals based on the K-variate random variable $(b^n - b^{n-1})$. The mean value of this difference is a good indicator of the health of the network.

There is two major advantages of the residuals generated this way. First, successive identification of the parameters allows the model to adaptively track local changes in its parameters. It is unrealistic to assume that the model parameters will remain exactly the same over all the operating times of the network. Second, the difference $(b^n - b^{n-1})$ does not depend on the "true" value of the parameter b. This is very important since, in practice, we do not know this "true" value, and the only available information is the value $\hat{b}$, estimated from the data. Approximating b with $\hat{b}$, and studying the difference $(b^n - \hat{b})$ is possible, but our experiments showed that this approach is inefficient.

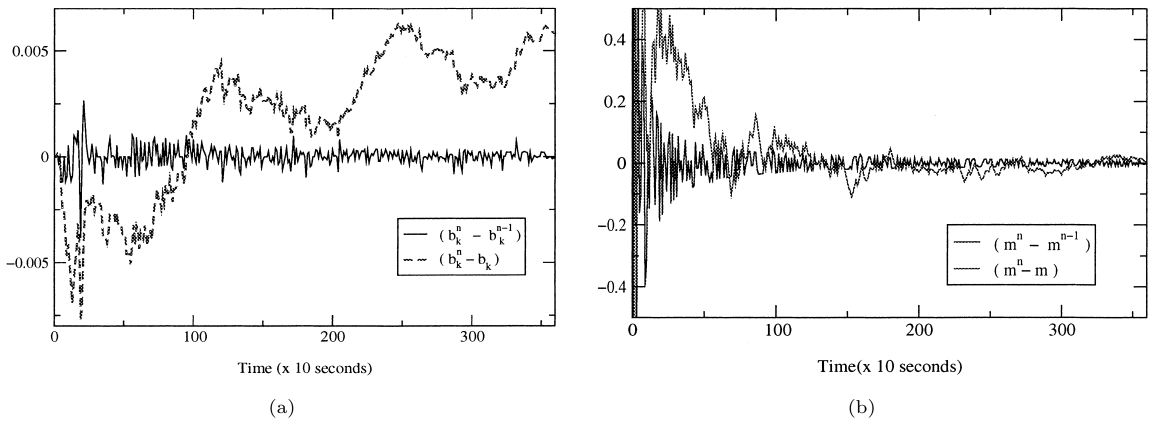

Fig. 1 Behavior of the drifts of two typical clusters of objects, (a) drifts $(b_k^n - b_k^{n-1})$ and $(b_k^n - \hat{b}_k)$ under the same network conditions, for the cluster *(IfInPkts, ipInReceives)*, (b) drifts $(m_k^n - m_k^{n-1})$ and $(m_k^n - \hat{m}_k)$ under the same network conditions, for the variable *ifOutNUcastPkts.*

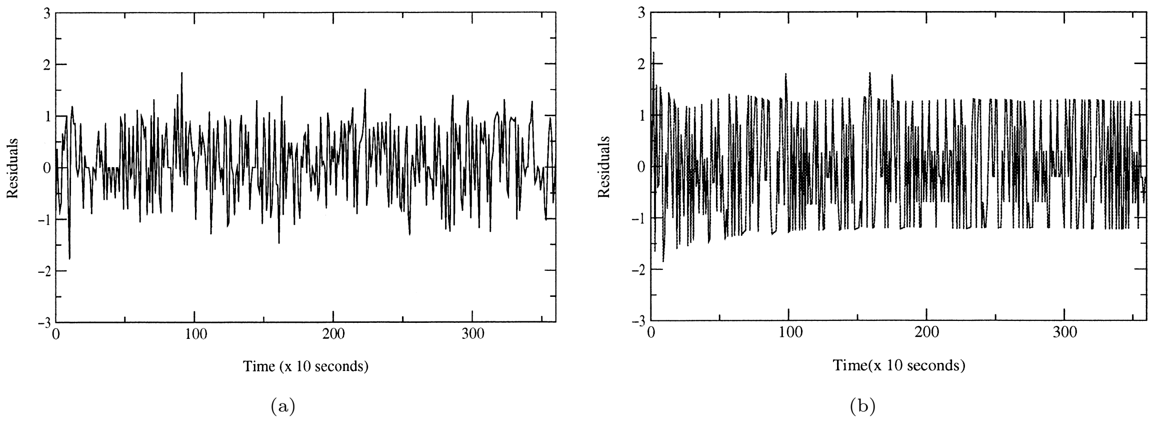

Fig. 2 Residuals plots under the same network conditions as in Fig. 1: (a) Univariate residuals e_{kn} for regime k of the switching regression, as given by Eq. (16), (b) e_{kn} for finite Normal mixtures, as given by Eq. (18).

Figure 1 compares both differences for a duration of one hour under the same network conditions. Results are shown only for one of the two regimes of the cluster, denoted by k. It is clear that the difference $(b_k^n - \hat{b}_k)$ is not symmetric around zero, while the difference $(b_k^n - b_k^{n-1})$ is both symmetric and very close to zero under normal conditions. We showed empirically[3] that the K-variate residuals e_n given by:

$$e_n = (b^n - b^{n-1})^T \Lambda^{-1} (b^n - b^{n-1}) \quad (16)$$

$$\Lambda = \text{diag}\left(\frac{\sqrt{w_{kn}} x_n \hat{\sigma}_k}{X_n^T W_{kn} X_n}\right) \quad (17)$$

For the case of mixture of Normal distributions, the residuals e_n are given by:

$$e_n = (m^n - m^{n-1})^T \Lambda^{-1} (m^n - m^{n-1}) \quad (18)$$

$$\Lambda = \text{diag}\left(\frac{\sqrt{w_{kn}} \hat{\sigma}_k}{\sum_{i=1}^{n} w_{ki}}\right) \quad (19)$$

are approximately Normal, with mean zero under network normal conditions. Note that e_n given in Eq. (16) (respectively Eq. (18)) is simply the difference $(b^n - b^{n-1})$ (respectively $(m^n - m^{n-1})$), scaled such that its variance-covariance matrix becomes Identity. **Figure 2** shows the behavior of the residuals e_{kn} under the same network conditions as in Fig. 1. It can be seen that these residuals are stable, and their mean is very close to 0.

In summary, network operations are characterized by the distribution of the residuals e_n.

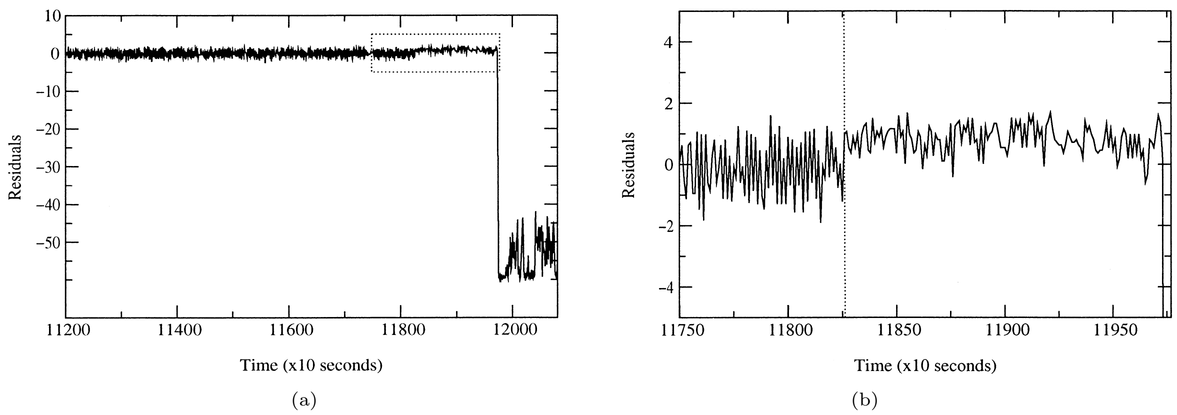

Fig. 3 Residuals generated by one of the two regimes of *(ifInPkts, IpIn-Receives)*, under abnormal conditions: (a) structural break variables relationship, (b) slight change in the mean of the residuals just before the break.

We showed that, under normal operations, the residuals e_n are approximately Normal with mean zero and variance Identity matrix. The next section shows the behavior of the residuals under abnormal conditions, and how we formulate and solve the detection problem.

3.2 Anomaly Detection

Anomaly detection is determining the discrepancy between the normal behavior and the predicted behavior. **Figure 3** shows the behavior of the residuals generated by the model under a real abnormal condition that affected Saitama university network, due to badly formatted packets. As shown in Fig. 3-a, this abnormal condition causes a sudden jump in the mean of the residuals. Fig. 3-b shows the behavior of the residuals just before the sudden jump in the mean. Interestingly, we notice that the sudden jump is preceded by a slight change in the mean of residuals. If the detection approach is designed to be sensitive to slight changes in the operating characteristics of the network, we could have predicted the problem of Fig. 3 at least 19 minutes before it became serious. The problem could have been avoided, or at least addressed immediately after its occurrence. In general, however, we do not expect all problems to present signs to allow their prediction. In this case, we require our detection method to raise alarm as soon as change in the mean occurs.

Consider the residuals E_c^n obtained by observing sequentially the residuals e_i from time point c to n. Under the normal operations of the network, the sample of e_n follows a K-variate Normal distribution with mean 0 and Identity covariance matrix (Sec. 3.1). At some unknown time point c, a change happens in the model, and the new generated residuals shift to a new distribution. The goal is to find a *decision function* and a *stopping rule* that detects this change and raise an alarm as soon as possible, under a controlled false alarm rate. This formulation is known in sequential analysis literature as the *disruption problem.* The main difference with classical hypothesis testing is that the sample size is a function of the observations made so far (i.e., not fixed a priori), and the distribution of the residuals is known, when the process being monitored, is in control. The goal is to achieve fast detection of change, by using no more than the necessary sample size to decide whether an alarm is to be raised or not.

It is well-known that for known probability distribution after change, Page-Lorden cumulative sum (CUSUM) 10),14) test is optimal, in the sense that it minimizes the delay to detection, among all tests with a given false alarm rate. However, in the present case of network anomaly detection, we do not have a priori knowledge about the distribution probability after change P_{θ_1}, and the change point c. The common extension of Page-Lorden CUSUM test consists of estimating the post-change distribution mean, and the change point from the data. This approach is known as the Generalized Likelihood Ratio (GLR) test 1),23). That is, the unknown parameter θ_1 of the distribution of $P_{\theta_1}(e_i)$ after change, and the change point c are estimated from data, using the maximum likelihood estimator. The resulting decision function is given by:

$$R_n = \sup_{1 \le c \le n} \sup_{\theta_1} \ln \frac{P(E_c^n|\theta_1, c)}{P(E_c^n|\theta_0)} \quad (20)$$

$$T_n = \inf\{n : R_n > \lambda\} \quad (21)$$

In our case, where pre-change and post-change distributions are Normal, the maximization problem of Eq. (20) can be worked out explicitly. It has a simple form, given by:

$$S_0 = (0, \ldots, 0)^T \qquad S_n = \sum_{i=1}^{n} e_i \quad (22)$$

$$T_n = \inf\{n : \max_{0 \le c < n} \frac{\| S_n - S_c \|}{\sqrt{n-c}} > \lambda\} \quad (23)$$

The equation assumes that after change, the distribution of the residuals is still Normal, but with different mean. For the abnormal case, it is hard to obtain an unbiased fit of the post-change distribution $P_{\theta_1}(e_i)$. Fortunately such accurate estimation is not crucial. What is needed is that, when an anomaly occurs, the closest Normal distribution, obtained by maximum likelihood estimation, has a mean significantly different from zero.

3.3 Tuning the Threshold λ

So far we have introduced the decision function and the stopping rule used for online detection of network faults and performance degradation. The remainder of our problem set-up concerns the choice of the design threshold λ.

It can be shown that the expectation of the stopping rule, under no change denoted by $E_\infty(T)$, is given by[17]:

$$E_\infty(T) \sim \frac{\Gamma(K/2) 2^{K/2} \exp(\lambda^2/2)}{\lambda^K \int_0^\lambda x v^2(x) dx} \text{as} \lambda \to \infty \quad (24)$$

$$v(x) = 2x^2 \exp\left(-2\sum_1^\infty n^{-1} \Phi\left(\frac{-x n^{1/2}}{2}\right)\right), \quad x > 0 \quad (25)$$

where Φ denotes the normal distribution function. For calculation, see[17),18)] for an approximation of $v(x)$. Not surprisingly, Eq. (24) turns out to be the mean time between false alarms. It follows that, given a desired false alarm rate, we can *recover* the design threshold λ, by solving Eq. (24).

4. Evaluation and Results

The network monitoring algorithms described earlier has been implemented in a real networks. This section discusses how the data is collected, and the results that validate the agent capabilities.

4.1 Experimental Setting

The network traffic is monitored using the standard MIB-II information base. To ensure reliable data sets, the agent is implemented to fetch the network statistics directly from the OS kernel. The agent is written in C++, and currently runs on both Solaris 2.6 and AIX 4.1. To validate the agent results, we implemented a program to access the underlying data-link layer for fault injection. The program is written using the Data Link Provider Interface (DLPI), on Solaris 2.6 operating system. The goal is not to simulate all possible faults, but rather to prove that alarms generated by the agent are, effectively, due to abnormal network conditions.

4.2 Implementation Aspects

To allow our model to track time varying parameters, we introduce an exponential forgetting factor $0 < \zeta \le 1$, that reduces the effect of old observations, much in the same way as proposed in[24)]. Evaluating the sum $\sum_{i=1}^{n} w_{ki} x_i^2$ (Eq. (12), Sec. 3.1) is then replaced by $\sum_{i=1}^{n} \zeta^i w_{ki} x_i^2$. Similar modification is done for finite Normal mixture parameter updating (Eq. (14)).

Unfortunately, Eq. (23) can not be written recursively. Consequently, the number of residuals to be inspected can grow large. To circumvent this difficulty, we use a moving horizon of fixed length, where the starting point of the horizon moves one step forward as new observation are made available to inspection. For the mean false alarm rate (Sec. 3.3), it is fixed to 8640, which corresponds to 24-hours period, given that samples are collected every 10 seconds. Finally, for the number of components K, it was found empirically that at most 4 regimes are enough to describe the data satisfactorily. Work is underway to infer the number of components automatically from data.

4.3 Detection Capability

The agent detection capability is first illustrated using the network problem, introduced earlier in Section 3.2. The problem occurred in Saitama university, on November 24th, 2000. It showed up as a streaming network interface card card sending excessively badly formatted

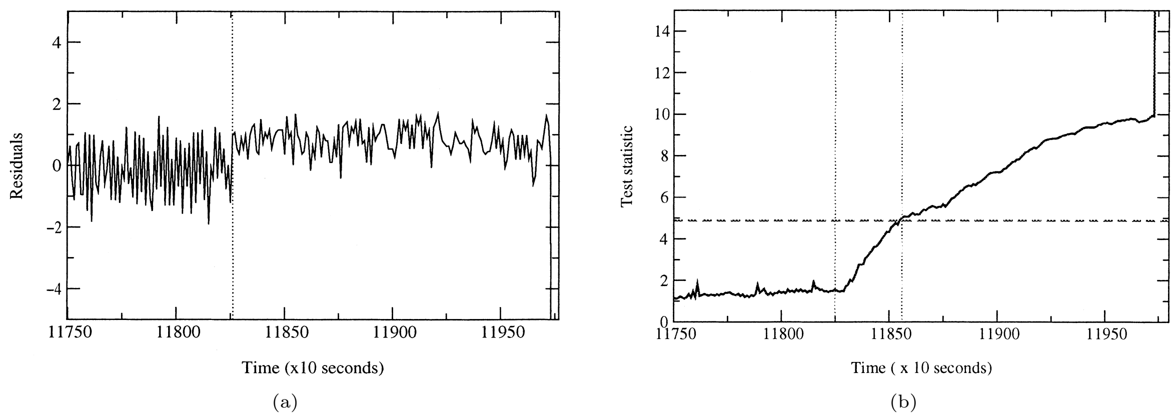

Fig. 4 Behavior of the test static for the problem that occurred in Saitama university network.

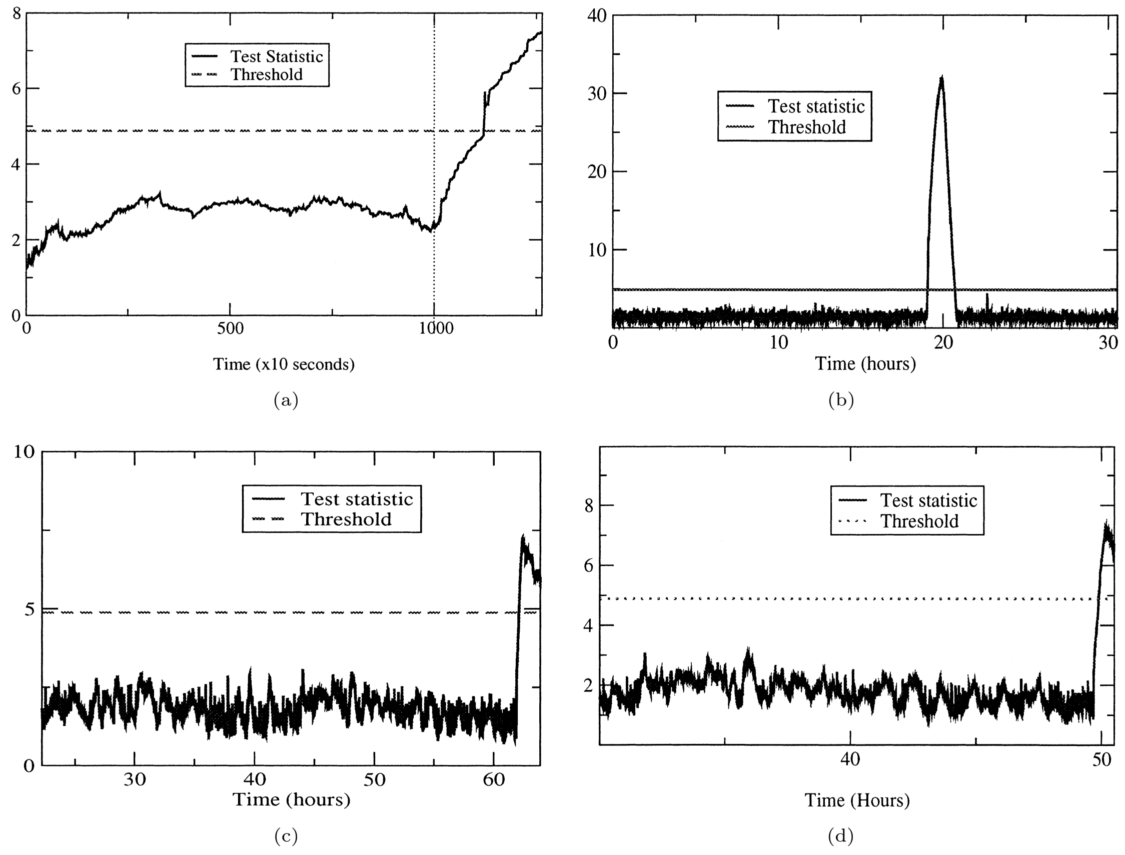

Fig. 5 Behavior of the test statistic corresponding to excessive ARP packets, excessive inbound broadcasts, excessive outbound broadcasts, and IP packet loss problems: (a) IP packet discards, (b) Outbound broadcast packets, (c) Inbound broadcast packets, (d) Inbound ARP packets.

packets. **Figure 4** shows the behavior of the residuals and test statistic as detected by the cluster (*ifInPkts, ipInReceives*). As shown in the figure, it takes approximately 30 samples (5 minutes) to detect the slight change in the residuals. In this particular case, the threshold is crossed 19 minutes before the sudden disruption. It could have been possible to address the problem, *proactively*, before it became serious, or at least draw the attention of network operators earlier, before the impact of the problem is felt by all network users.

Detection capability of the agent is tested with other problems. The obtained results are shown in **Fig. 5**. For IP packet loss and excessive outbound broadcasts, appropriately assembled packets are sent every two seconds. For the remaining problems, assembled packets are injected every one second. It can be seen that the agent could detect all of these problems, with reasonably short delay.

We note that, apart from reasons given in Section 2.1 for defining network model as the parametric characterization of dependent objects, there is no MIB counter for ARP operations. Also, for the problem of packet loss induced by assembling Ethernet packets with non-existent protocol type, some kernels do not have any entry for this type of errors, even if this object is part of the standard MIB-II information base. For IP operations, one can easily check that *IpInDelivers* and *IpInReceives* can be sometimes very large, without noticing any change in IP related errors. This is true even when we take into consideration the fact that loopback packets are to be added to *IpInReceives*. With current under-instrumented networks, our approach to network modeling and performance problems detection can provide useful insights about incipient problems.

In conclusion, we showed that our proposed approach is very efficient. In all the results reported here, we actually stress tested the agent detection capability. The total amount of all injected packets is smoothed over the whole test duration. In practice, network problems are expected to induce large magnitude changes in the model residuals. The detection is expected to even work better.

4.4 Alarm Rate

Ideally, we would like to estimate the false alarm rate, given that the network is operating normally. Unfortunately, it is difficult to gain perfect knowledge about all the subtle changes in the network behavior. Instead, **Table 2** shows the average alarm rate per hour, for data collected during a period of 24-hours.

We note that ifOutNUcastPkts is very stable, recording a zero alarm rate per 24-hours. The same comment is also true for ifInNUcastPkts. This raises questions about whether the switching regression model adds any accuracy to the model. It is probably enough to model all the variable by a finite mixture model. We plan to investigate this question in detail in the future.

Table 2 Average number of alarms per hour for each of the clusters of network model.

Clusters	**Average alarm rate per hour**
ifInNUcastPkts	0.04
ipInReceives, *ifInPkts*	0.50
ipInReceives, *ipInDelivers*	0.08
ipForwDatagrams, *ipInReceive - ipInDelivers*	0.04 ☆
ifOutPkts, ipForwDatagrams + *ipOutRequests*	0.16
ifOutNUcastPkts	0.00

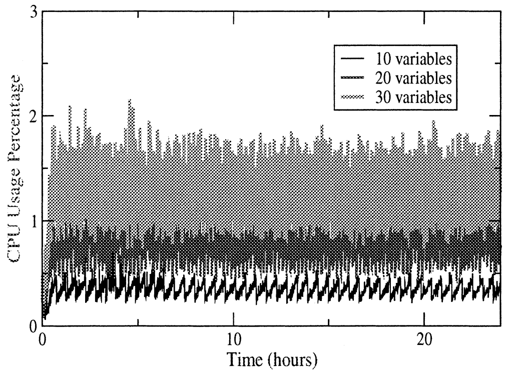

Fig. 6 CPU time percentage consumed by the monitoring process.

5. CPU Time

The final aspect we investigate in our proposed approach is the amount of CPU time consumed by the monitoring agent. In this experiment, another process is set to sample the agent's CPU time usage, every 10 seconds. The agent runs on SunOS Release 5.8, with SPARC CPU (333 MHz). **Figure 6** shows the CPU time consumed by the agent, for monitoring simultaneously 10 variables, 20 variables, and 30 variables.

It can be clearly seen that the monitoring consumes a very small amount of the CPU time. The obtained results shows that for monitoring 10 variables simultaneously, the process consumes an average of 0.34 CPU time. The average is calculated by sampling the CPU usage every 10 seconds, for an overall duration of 24

☆ Results are from an internal router of Saitama University network.

hours. Scaling this figure to 20 variables, an average of 0.76 CPU usage has been recorded. Finally with 30 variables 1.27 have been achieved. It is safe, then, to claim that our approach can scale well with large number of variables.

Overall results shows a highly efficient network monitoring scheme. The methods for both monitoring are generic to be readily tailored to specific scenarios. It is, however, important to note that convergence of learning algorithm for parameter identification needs very large data sets. It is recommended that a sample of at least two week to one month of data points is to be used for normal operations baselining.

6. Conclusion

In this paper, we developed an online technique for fast detection of performance problems in IP-Networks. We proposed a model of the network operations in terms of MIB objects dependencies, and we showed that the parametric characterization of this dependency is amenable to a finite mixture model. Model parameters are identified from routine operation data, using the expectation maximization algorithm. A new method for residual generation, based on successive parameter identification, is introduced. The residuals are shown to be approximately Normal, with mean zero under normal operations, and sudden jumps in this mean are characteristics of abnormal conditions. A real-time online change detection algorithm is designed to process the residuals sequentially, and raise an alarm as soon as the anomaly occurs. The proposed approach requires neither the set of faults and performance degradation nor the thresholds to be supplied by the user. Experimental results showed the effectiveness of the method on real data. A low false alarm rate and a high detection capability has been demonstrated.

References

1) Basseville, M. and Nikiforov, I.V.: *Detection of Abrupt Changes: Theory and Application*, Prentice-Hall (1993).
2) Dempster, A., Laird, N. and Rubin, D.: Maximum Likelihood from Incomplete Data via the EM Algorithm, *J.R. Statist. Soc. B*, Vol.39, pp.1–38 (1977).
3) Hajji, H. and Far, B.H.: Continuous Network Monitoring for Fast Detection of Performance Problems, *Proc. 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems* (2001).
4) Hartley, M.J.: Comment on "Estimating mixtures of Normal Distributions and Switching Regressions", *J. Am. Stat. Assoc.*, Vol.73, pp.738–741 (1978).
5) Hood, C. and Ji, C.: Proactive Network Fault Detection, *IEEE Trans. Reliability*, Vol.46, No.3, pp.333–341 (1997).
6) Jakobson, G. and Weissman, M.D.: Alarm Correlation, *IEEE Network*, pp.52–59 (1993).
7) Katzela, I. and Schwarz, M.: Schemes For Fault Identification is Communication Networks, *IEEE/ACM Trans. Networking*, Vol.3, pp.753–764 (1995).
8) LaBarre, L.: Management by Exception: OSI event generation, reporting, and logging, *Proc. Second International Symposium on Integrated Network Management* (1991).
9) Leinwand, A. and Conroy, K.F.: Network management, a practical perspective, 2nd Edition, Addison-Wesley (1996).
10) Lorden, G.: Procedures for reacting to a change in distribution, *Annals of Mathematical Statistics*, Vol.42, pp.1897–1908 (1971).
11) Maxion, R.A. and Feather, F.E.: A Case Study of Ethernet Anomalies in a Distributed Computing Environments, *IEEE Transactions on Reliability*, Vol.39, No.4, pp.433–443 (1990).
12) McCloghrie, K. and Rose, M.: Management Information Base for Network Management of TCP/IP-based internets: MIB-II, RFC 1213 (1991).
13) McLachlan, G.J. and Basford, K.E.: *Mixture Models: Inference and Application to Clustering*, New York, Dekker (1988).
14) Page, E.S.: Continuous Inspection Schemes, *Biometrika*, Vol.41, pp.100–115 (1954).
15) Quandt, R.E.: A New Approach to Estimating Switching Regressions, *J. Am. Stat. Assoc.*, Vol.67, No.338, pp.306–310 (1972).
16) Redner, R.A. and Walker, H.F.: Mixture Densities, Maximum Likelihood and the EM Algorithm, *SIAM Review*, Vol.26, No.2, pp.195–239 (1984).
17) Seigmund, D. and Venkatraman, E.S.: Using the Generalized Likelihood Ratio Statistic for Sequential Detection of a Change Points, *The Annals of Statistics*, Vol.23, No.1, pp.255–271 (1995).
18) Siegmund, D.: *Sequential Analysis: Tests and Confidence Intervals*, Springer, New York (1985).
19) Stalling, W.: *The SNMPv1, SNMPv2 and RMON*, Addison-Wesley, Reading, MA, USA (1999).
20) Thottan, M. and Ji, C.: Statistical Detection of Enterprise Network problems, *Journal of*

Network and Systems Management (1999).

21) Titterington, D.M.: Recursive Parameter Estimation using Incomplete, *Journal of Royal Statistics Society*, Serie B, Vol.46, No.2, pp.257–267 (1984).

22) Vlassis, N.A.: A Kurtosis-Based Dynamic Approach to Gaussian Mixture Modeling, *IEEE Transactions On Systems, Man, And Cybernetics, Part A*, Vol.29, No.4, pp.393–399 (1999).

23) Willsky, A.S. and Jones, H.L.: A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems, *IEEE Transactions on Automatic Control*, Short Paper, pp.108–112 (1976).

24) Weinstein, E., Feder, M. and Oppenheim, A.V.: Sequential algorithms for parameter estimation based on Kullback-Leibler information measure, *IEEE Trans. Acous., Speech, Signal Processing,* Vol.38, No.9, pp.1652–1654 (1990).

25) Yemini, S., Kliger, S., Mozes, E., Yemini, Y. and Ohsie, D.: High speed and robust event correlation, *IEEE Communication Magazine*, pp.82–90 (1996).

(Received August 7, 2001)
(Accepted November 5, 2002)

Hassan Hajji received his BS.c degree from Mohamed Premier University, Morocco in 1995, MSc. and Ph.D degrees from Saitama University, Japan in 1999 and 2002, respectively. He is currently with IBM research, Tokyo Research Laboratory, Japan. His current research interests include network and system management, traffic modeling and analysis, and autonomic computing. Dr. Hassan Hajji is a member of IEEE computer society and IPSJ.

Behrouz Homayoun Far received BSc. and MSc. degrees in Electrical Engineering in 1993 and 1986, respectively, from Tehran University, Iran. He has received his Ph.D degree from Chiba University, Japan in 1990. He is currently an Associate Professor at the Department of Electrical and Computer Engineering, University of Calgary, Canada, where he is the coordinator of the Intelligent Systems Group at the University of Calgary. The research fields of his interests are automatic programming, software quality management and distributed AI. Dr. Far is a member of the ACM, IEEE computer society, JSAI, and IPSJ.