Detection of Network Faults and Performance Problems

Hassan Hajji, B. H. Far, and Jingde Cheng
Department of Information and Computer Science
Saitama University,
Saitama 338-8570, JAPAN
e-mail: hajji@aise.ics.saitama-u.ac.jp
far@enel.ucalgary.ca
cheng@ics.saitama-u.ac.jp

Abstract — Network normal operation baselining for automatic detection of anomalies is addressed. A model for network traffic is presented in which studied variables are modeled as a finite mixture model. Based on stochastic approximation of the maximum likelihood function, we propose a baseline of network normal operation as the asymptotic distribution of the difference between successive estimates of model parameters. The baseline multivariate random variable is shown to be stationary, with mean zero under normal operation. Performance problems are characterized by sudden jumps in the mean. Detection is formulated as an online change point problem, where the task is to process residuals and raise alarms as soon as anomalies occur. An analytical expression of false alarm rate allows us to choose the threshold, automatically. Extensive experimental results on a real network showed that the monitoring agent is able to detect even slight changes in the characteristics of the network, and adapt to traffic patterns, while maintaining a low alarm rate. Despite large fluctuations in network traffic, this work proves that tailoring traffic modeling to specific goals can be efficiently achieved.

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 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. In addition, network monitoring agents can be designed to interface network protocols, to tune their operations. For example, routing metrics can be adjusted based on management agents alarms.

A large amount of work has gone into developing mechanisms and protocols for collecting traffic statistics. Indeed, currently most of the work in the simple network management architecture focuses on defining detailed system and network traffic objects. Comparatively, little work is done to support user analysis of collected statistics. Most of the interpretation is left to the common-sense of network operators. Unless control mechanisms are driven by objective measures using well-tested network traffic models, the benefits and the results of network traffic analysis will remain biased by the common sense of human operators. On the other hand, existing network on fault and performance management assumed that the alarm generating mechanism is accurate, and network problems are given a priori [12, 11, 26]. 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 [10]. 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 generally accepted definition of what constitutes normal behavior [15]. The dynamics of the network normal operations need to be identified from routine operation data. Earlier work reported in [18] 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, [10] considered the model formed by segmenting time series obtained form Management Information Base (MIB) objects. Observations are declared abnormal if they do not fit an auto-regressive model of the traffic inside segments. In [22] 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 traffic volume is generally not sustained at a given level long enough to allow accurate estimation.

In this paper, we address the problem of faults and performance problems detection in local area networks. No knowledge about the problems to be detected is required. The emphasis is on fast detection – an important requirement for reducing potential impact of problems on network services users. We parameterize network traffic variables using finite Gaussian mixtures. Based on this parametric model, we propose a baseline of network normal operation as the asymptotic distribution of the difference between successive estimates of model parameters. This difference is 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 on a real network showed the effectiveness of our approach. A very low alarm rate and a high detection has been demonstrated.

This paper is arranged as follows: Section 2 introduces our proposed parametric model of the network traffic, and how network normal operation baseline is derived. Section 3 shows the characteristics of the baseline model, under abnormal condition, and introduces the formulation of the network problem detection. In Section 4, we present results of our experiments in a real network. We conclude in section 5.

2 Normal Operations Baselining

The goal of this section is to characterize network normal behavior. We first present a parametric model of traffic variables, and then show how this model can be used to build a baseline of normal operations.

A Traffic Variables Parametric Model

Our approach to network model parameterization is to view each variable as switching between different regimes, where each regime is a Gaussian distribution. This is a form of what referred to in the literature as *finite mixture model* [20]. The observations x_i are generated by one of the K Gaussian distribution, as shown in the following equations:

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

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

The errors ϵ_k are assumed to be Gaussian, with mean 0 and variance σ_k. The integer K denotes the number of regimes (components). Each regime k has a mixing probability, denoted by π_k. Strictly speaking, the errors ϵ_k should be truncated Gaussian, since negative, and extremely large values do not appear in the network traffic data sets. As it turns out, such issues do not affect the accuracy of the detection significantly.

The parametric finite mixture model has an attractive interpretation. It is useful in the analysis of data that are believed to come from a finite number of distinct subpopulations, indicated by a latent variable. In our case, the latent variable has the natural interpretation of time of the day, given the known fact that network traffic changes as a function of the time.

Recent work on characterizing operation of network traffic has resulted in analytical models of many important network statistics. For instance, mixes of lognormal distributions have been found to model very well the call holding time in telephony [3, 4]. In [14], it was found that Telnet originator responder bytes, data transmitted in a given FTP connection, FTP session bytes can be modeled as a lognormal distribution. Lognormal distribution also fits well the distribution of message length in Public Access Mobile Radio (PAMR), a mixture of two lognormal distributions gives the best fit for transmission length [1]. Given the fact that if the random variable X is lognormal, then $log(X)$ follows a Normal distribution, the random variable whose realization are the logarithm of the original data can be modeled by the finite mixture model of Equation 2. This model provides, then, a good parametric characterization for many important traffic characteristics, with the advantage of accounting of non-stationary nature of this statistics, due to hourly changes in network traffic. Note that in general, finite Gaussian mixture models are general enough to approximate any continuous function with a finite number of discontinuities [24], providing a general first approximation to other network traffic variables.

B Normal Operation Baselining

Our approach to operation baselining starts by recognizing that any parametric model of network traffic is, at best, an approximation to the reality. Approximation errors and model mis-specification become very pronounced in any inference that use the estimated parameters, as if they are the "true" ones. Given that our ultimate goal is anomaly detection, formulated as a change detection in baseline model parameters, we claim that this task can be achieved without using these "true" parameters. The goal is to avoid solving the more general problem of parameter estimation, as an intermediate step to change detection.

Our approach to realize this idea is pictorially shown in Figure (1). Observation are passed through the learning algorithm to produce the point estimation θ^{n-1}. As new data points are sequentially added, the learning algorithm outputs a refined new estimate θ^n. The idea is to characterize network normal operations using the difference $(\theta^n - \theta^{n-1})$.

There are two major advantages of the residuals generated this way. First, the difference $(\theta^n - \theta^{n-1})$ does not depend on the "true" value θ_0 of the parameter θ. This is very important since, in practice, we do not know this "true" value, and the only available information is the value $\hat{\theta}$, estimated from the data. Approximating the true parameter θ_0 with $\hat{\theta}$, and studying the difference $(\theta^n - \hat{\theta})$ is possible, but our experiments showed that this approach is inefficient, as shown later. Second, the learning algorithm can be designed to adaptively track local changes in model parameters. It is unrealistic to assume that the model parameters will remain exactly the same over all the operating times of the network.

Intuitively, under normal conditions, the difference between the successive values θ^n and θ^{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, induced by a pattern not present in the training data. The learning algorithm will alter the parameter θ to its new value. The idea, then, is to generate the residuals based on the random variable $(\theta^n - \theta^{n-1})$. The mean value of this difference is a good indicator of the health of the network.

To realize this principle, two issues need to be addressed. First, the issue of how to design an adaptive learning algorithm. In addition, since detection is required to be online, the learning of Figure 1 should as fast as possible. Second, we have to work out the distribution of the difference $(\theta^n - \theta^{n-1})$. The remainder of this section addresses these two issues.

C Learning Algorithm and Residual Distribution

A well-known algorithm for parameter identification in finite mixture models is the Expectation Maximization (EM) algorithm [7]. The EM is, however, a batch-oriented algorithm. It requires the whole data to be available in memory, before a new refined estimates is produced. If we have to run this algorithm for each new observation, too much time and memory will be consumed. Worse yet, time and memory consumed keeps growing as monitoring goes on. We do not follow this approach, instead a stochastic approximation of the problem of maximizing the likelihood function, is used to turn the EM to an online algorithm.

Let the vectors $x_1, \ldots, x_n$ be a sequence of observations, whose joint probability distributions $f_x(\theta)$ depends on the unknown parameter θ. The goal is to derive an online algorithm for estimating the parameter θ. We define the recursive likelihood function $L^n(\theta)$ as follows:

$$L^n(\theta) = E_{\theta^n}\big(\log(f(y_n|x_1 \ldots x_{n-1})\big) + L^{n-1}(\theta) \quad (3)$$

Where $E_{\theta^n}(.)$ denotes the expectation with respect to the parameter θ_n, y is the latent, unobservable variable. This is basically the same recursive likelihood as in [23], except that the expectation is taken, conditionally on the whole set of observations $x_1 \ldots x_{n-1}$. It can be shown that the solution θ_n of the problem of maximizing $L^n(\theta)$ is given by:

$$\theta^n = \theta^{n-1} + I_c^{-1} S(x_n, \theta_n) \quad (4)$$

$$S(x_n, \theta_n) = D \log(f(x_n|y_1 \ldots y_{n-1})\big) \quad (5)$$

Where I_c denotes Fisher information matrix for the complete data, where the separation variable is known. Similar results are obtained in [25] by minimizing the Kullback-Liebler divergence, instead of maximizing the recursive likelihood. Working out the scores $S(x_n, \theta_n)$, and the I_c, leads to the following recursive formulas for updating the model parameters:

$$m_k^n = m_k^{n-1} + \frac{w_{kn}}{(\sum_{i=1}^n w_{ki})}(x_n - m_k^{n-1}) \quad (6)$$

$$\sigma_k^{2(n)} = \sigma_k^{2(n-1)} + \frac{w_{kn}}{\sum_{i=1}^n w_{ki}}((x_n - m_k^{n-1})^2 - \sigma_k^{2(n-1)}) \quad (7)$$

Where

$$w_{ki} = \frac{\pi_k^{n-1} f_{ik}}{\sum_{k=1}^K \pi_k^{n-1} f_{ik}} \quad (8)$$

$$f_{ik} = \frac{1}{\sqrt{2\pi\sigma_k^{2(n-1)}}} \exp\big(\frac{-(x_i - m_k^{n-1})^2}{2\sigma_k^{2(n-1)}}\big) \quad (9)$$

$$k = 1 \ldots K \quad (10)$$

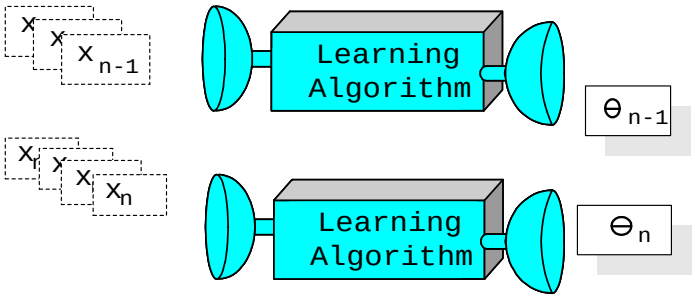


Figure 1: Normal operation baselining based on repeated identification of model parameters

Now let us verify our design goal. First, note that parameter updating is fast enough, to be implemented in real-time. Sequentially acquired data points are merged with existing processed data, and do not require the re-computation of all collected data. Time and memory requirements are, hence, kept minimal. To allow the learning algorithm to track slight change in model parameters, we introduce an exponential forgetting factor $0 < \zeta \leq 1$, that reduces the effect of old observations. Evaluating the sum $\sum_{i=1}^{n} w_{ki}$ is then replaced by $\sum_{i=1}^{n} \zeta^i w_{ki}$. In the sequel, we shall be interested only in changes in the K-dimensional mean $m = (m_1, \ldots, m_k)$.

As stated earlier, approximating θ_0 with $\hat{\theta}$, and studying the difference $(\theta^n - \hat{\theta})$ is possible, but our experiments showed that this approach is inefficient. Figure 2-a compares both differences for a duration of one hour under the same network conditions. Results are shown only for one of the second component, in the mixture model of the number of broadcast packets traffic variable. It can be concluded that the $(m_k^n - \hat{m}_k)$ is not symmetric around zero, while the difference $(m_k^n - m_k^{n-1})$is both symmetric and very close to zero under normal conditions.

For the distribution of the residuals, we showed empirically [9] that the K-variate residuals e_n given by:

$$e_n = (m^n - m^{n-1})^T \Lambda^{-1} (m^n - m^{n-1}) \qquad (11)$$

$$\Lambda = \text{ diag } \left(\frac{\sqrt{w_{kn}} \hat{\sigma}_k^{n-1}}{\sum_{i=1}^{n} w_{ki}} \right) \qquad (12)$$

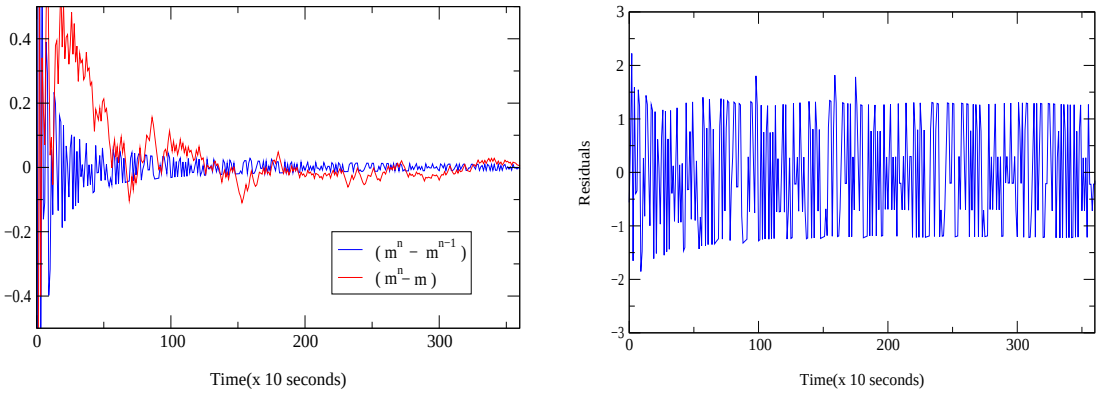
are approximately Normal, with mean zero under network normal conditions. Note that e_n given in Equation (11) is simply the difference $(m^n - m^{n-1})$, scaled such that its variance-covariance matrix becomes Identity. Figure (2)-b shows the residuals e_{2n}, corresponding to the mixture model of the number of broadcast packets traffic variable. It can be seen that the residuals e_{2n} are stable, and their mean is very close to 0.

To summarize results of this section, we showed how the learning algorithm transforms the raw data, to stationary multi-variate residuals e_n. The residuals e_n have the desirable property of being Normal with mean zero and variance Identity matrix, under normal network operations. The mean of the random variable e_n serves as the *baseline for normal operation*. The next section shows the behavior of these residuals under abnormal conditions, and how we formulate and solve the detection problem.

3 Anomaly Detection

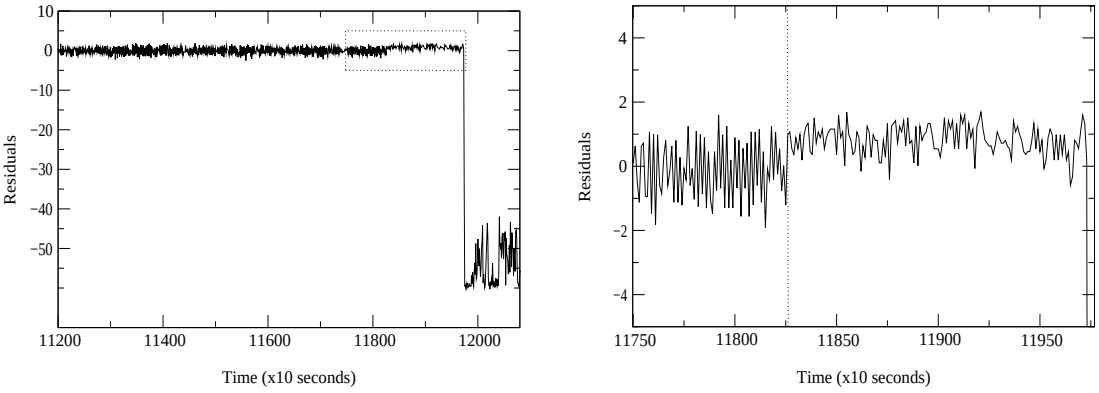
Anomaly detection is determining the discrepancy between the normal behavior and the predicted behavior. Figure 3 shows the behavior of the residuals generated by the model under a real abnormal condition that affected Saitama university network, due to badly formatted packets. As shown in Figure 3-a, this abnormal condition causes a sudden jump in the mean of the residuals. Figure 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 Figure 3 before it became serious. The problem could have been avoided, or at least addressed immediately after its occurrence. In general, however, not all problems presents 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 zero, and Identity variance-covariance matrix (Section B). At some unknown time point c, a change happens in the model, and the new generated residuals shift to a new distribution, with a different mean, denoted by θ_1. 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



(a) (b)

Figure 2: Comparison of the drifts $(m_2^n - m_2^{n-1})$ and the $(m_2^n - \hat{m}_2)$



(a) (b)

Figure 3: Behavior of the residuals under abnormal network conditions

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 sufficient 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) [2] 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 probability distribution 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 [2]. That is, for the unknown parameter θ_1 of the post-change distribution $P_{\theta_1}(e_i)$, and the change point c are estimated from data, using the maximum likelihood estimator. The resulting decision function is given by:

$$R_n = \sup_{1 \le c \le n} \sup_{\theta_1} \ln \frac{P(E_c^n|\theta_1, c)}{P(E_c^n|\theta_0)} \tag{13}$$

$$T_n = \inf \{n : R_n > \lambda\} \tag{14}$$

In our case, where pre-change and post-change distributions are Normal, the maximization problem of Equation (13) can be worked out explicitly. It has a simple form, given by:

$$S_0 = (0, \ldots, 0)^T \qquad S_n = \sum_{i=1}^{n} e_i \tag{15}$$

$$T_n = \inf \{n : \max_{0 \le c < n} \frac{\|S_n - S_c\|}{\sqrt{n-c}} > \lambda\} \tag{16}$$

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.

A 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 [21]:

$$E_\infty(T) \sim \frac{\Gamma(K/2)2^{K/2}\exp(\lambda^2/2)}{\lambda^K \int_0^\lambda xv^2(x)dx} \text{ as } \lambda \to \infty \qquad (17)$$

$$v(x) = 2x^2 \exp\left(-2\sum_1^\infty n^{-1}\Phi(\frac{-xn^{1/2}}{2})\right),\ x > 0 \qquad (18)$$

Where Φ denotes the Normal distribution function. For calculation, see [21] for an approximation of $v(x)$. Not surprisingly, Equation (17) 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 Equation (17).

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.

A Data Collection

The implementation of the monitoring software consists of two modules: statistics collection and monitoring modules. Statistic collection module interfaces the network for protocols operation statistics. The monitoring module monitors management objects, for online anomaly detection.

Statistics collection is implemented as a Remote Monitoring (RMON) agent, running as user-level process. This solution is particularly appealing in the sense that dependency on the operating system kernel is minimally reduced to the interface to access the data link layer. This way, we have full control of all aspects of network statistics, as opposed to depending on whether operating system kernel keeps track of traffic statistics. Our earlier implementation experience of an SNMP agent [9], revealed that some kernels do not have entries for all management objects, as defined in MIB-II

The network monitoring module operates on top of RMON. It accesses raw measurements through RMON management information base. All monitoring computation is done locally. In contrast to the Network Management Station (NMS) based polling of network statistics, the bandwidth and computation time consumed during transfer and processing of raw measurement is kept minimal. Distributed management organizational models [8, 17], addressed these issues, but failed to address the critical issue of how to use collected statistics. In this sense, our approach complements distributed management by providing the details of monitoring tasks to be carried by the distributed agents. Currently our monitoring agent software runs on Linux operating system.

B Experimental Setup

To illustrate each of the capabilities of our proposed monitoring approach, Table 1 lists the variables studied. The first three variables are modeled using a finite Gaussian mixture model. The last variable data is modeled using a finite mixture of lognormal distribution. For each of these variables, Table 2 shows its experimental configuration. The number of components is determined in an ad-hoc manner, based on observed fluctuation of traffic in one week training data. We are now studying how to choose the number of components automatically form the data. It is, also, assumed that the training data is "pure". That is, no anomaly occurred during its collection.

C Detection Accuracy

Figure (4)-a shows how the decision function reacts to excessive broadcasts, created by injecting additional two broadcast packets, every second. As shown in the figure, the decision function shows a sharp increase, crossing the threshold after a delay of 16 minutes approximately. Figure(4)-b shows how the test statistic reacts to a sustained rate of TCP passive opens, created by injecting 10 additional packets every second. It can be shown that the problem is detected, with a delay of 17 minutes, approximately.

Our RMON implementation allows us to study per-protocol statistics. Here we focus on Address Resolution Protocol (ARP), as an illustrative example, given both the lack of counters for ARP packet operations, and the range of problems that manifest themselves as changes in the statistic characteristics of this protocol traffic. Figure (4)-c shows results of monitoring ARP operation for 24 hours, and then perturbing network operations by injecting an additional two ARP request packets per seconds. It can be seen clearly that the anomaly is detected. The delay to detection is 16 minutes, approximately.

Variable	Definition
etherStatsBroadcastPkts	The total number of good packets received that were directed to the broadcast address.
etherStatsUnknownProts	The total number of Ethernet packets with an unknown protocol type
arpStatsPkts	The total number of good Address Resolution Protocol (ARP) packets on the segment
tcpPassiveOpens	The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state

Table 1: Definition of MIB variables used in experimental results

Variable	Number of Components	Forgetting Factor	Horizon Length
etherStatsBroadcastPkts	4	0.8	180
etherStatsUnknownProts	3	0.8	180
arpStatsPkts	3	0.8	180
tcpPassiveOpens	3	0.7	180

Table 2: Configuration of each of the variables studied

Figure (4)-d shows how the agent reacts to excessive unknown protocols. This variable is obtained by simply counting packets that do not conform to Ethernet packet format. Most of this traffic is caused by packets with protocol type less than 1500. In this case, it took 50 seconds for this anomaly to be detected.

In summary, we note that in all cases tested the detection is accurate, even with slight changes in network traffic. Recall that we are not modeling particular faults, and performance degradation patterns. The agent contrasts the baseline normal behavior with observed traffic, making it possible to detect novel network problems. It follows that, in principle, our approach can be easily deployed across different networks. It is also important to notice that in all the above cases, analyzing the captured packets after an alarm is raised, immediately reveals the problem. This is to be contrasted with methods as in [26], that take alarms as given, and yet have to match faults signatures with pre-stored patterns. These approach is both knowledge-extensive, and do not have the capability to learn new, unseen faults.

D Adaptability to Normal Traffic Fluctuations

Network traffic exhibits clear diurnal patterns. Night hours and less busy days of the week show a decrease in network traffic. Traffic picks up again during working days. The purpose of this section is to show that the monitoring agent learns these patterns, and does not take them for anomalies.

Figure 5-(a) shows an increase in ARP volume, that is part of normal operation of the network. ARP packets count increases, as transition is made from night hours to day hours. Figure 5-(b) shows that the decision function remains within the normal range, for this pattern. Similarly, Figure 5-(c) shows a decrease in ARP traffic volume, as network becomes less busy, in late evening hours. Figure 5-(d) plots the reaction of the decision function, around the same time this transition took place. Here also, the decision remains within its normal range. What actually happens, is that ARP traffic model is a mixture, with three components each modeling a given level of ARP traffic. Depending on the time of the day (latent variable), observations are assigned to the corresponding component, making it possible to adapt to these traffic fluctuation. One can conclude, then, that if the training data is large enough to contain all the possible regimes of operation, the monitoring can adapt to these patterns, and will not be taken for anomalies.

E Alarm Rate

Ideally, we would like to estimate the false alarm rate, given that we know for sure 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 3, and Table 4 show the average alarm rate per hour, evaluated after the agent is set to run for one week, then one month, re-

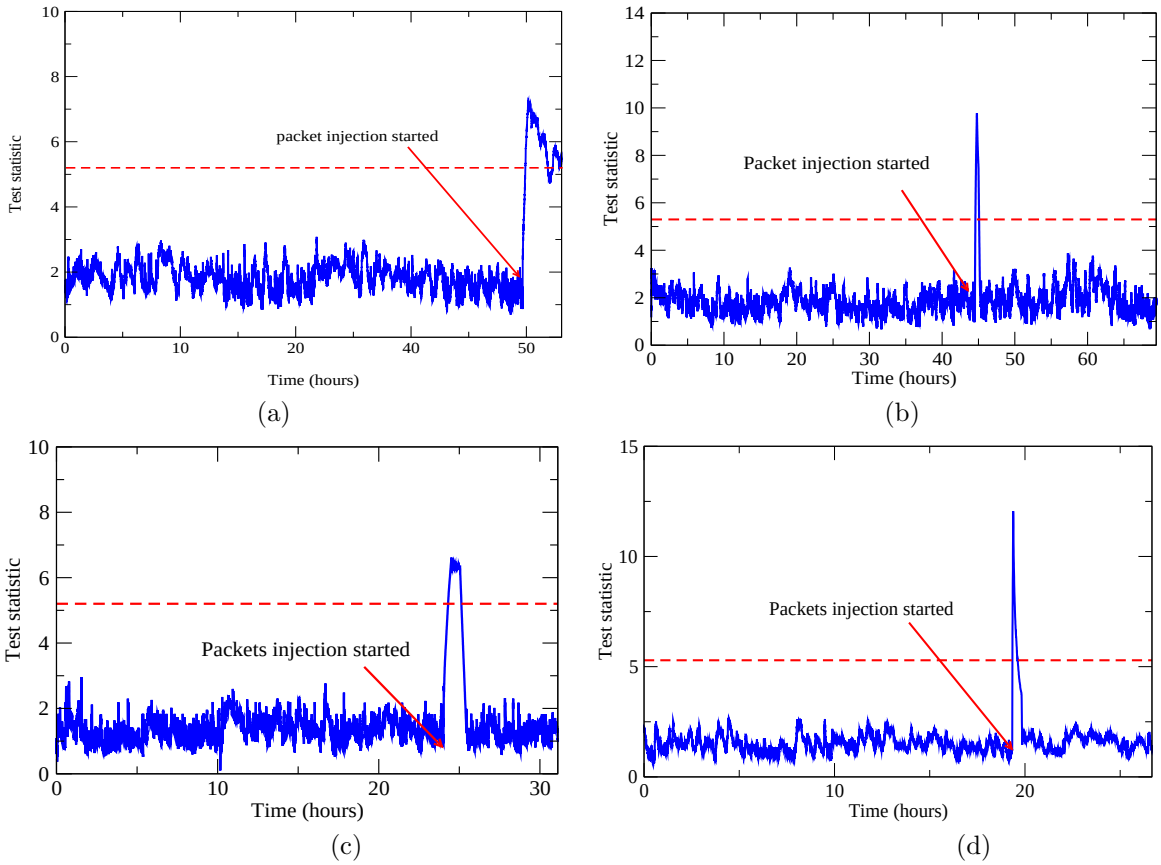


Figure 4: Behavior of the test statistic corresponding to (a) excessive broadcasts (b) sustained rate of TCP passive opens (c) excessive ARP packets (d) excessive unknown packets

spectively.

The duration of the testing is long enough to conclude that our monitoring technique adapts to different traffic patterns. The results show a very low alarm rate, yet a high detection accuracy, as evidenced by results of section C. In addition, it should be noted that most of the alarms generated by the variables *tcpPassiveOpens* and *etherStatsBroadcasts* in Table 4 are caused by one particular anomalous activity. Around this problem, 35 alarms generated by the variable *tcpPassiveOpens*, out of 43 total alarms raised, during one month duration of the experiment. These alarms are generated around one anomaly that manifested itself as an almost fixed rate of passive opens for 14 hours. In the remaining 29 days, only 8 alarms are generated. The same also applies to broadcast packets. Broadcast packets generated 12 alarms, 11 of them are caused by a failure of the file server in the neighboring subnet. Only one alarm is generated for the remaining 29 days.

5 Conclusion

In this paper, we developed an online technique for real-time detection of anomalies in IP-Networks. We showed that the parametric characterization of studied variables 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 processes, sequentially, the residuals 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 accuracy has been demonstrated. The key innovation that allowed efficient detection of network problems was to avoid solving the more general problem of accurate parameter estimation of traffic model, as in intermediate step for change detection.

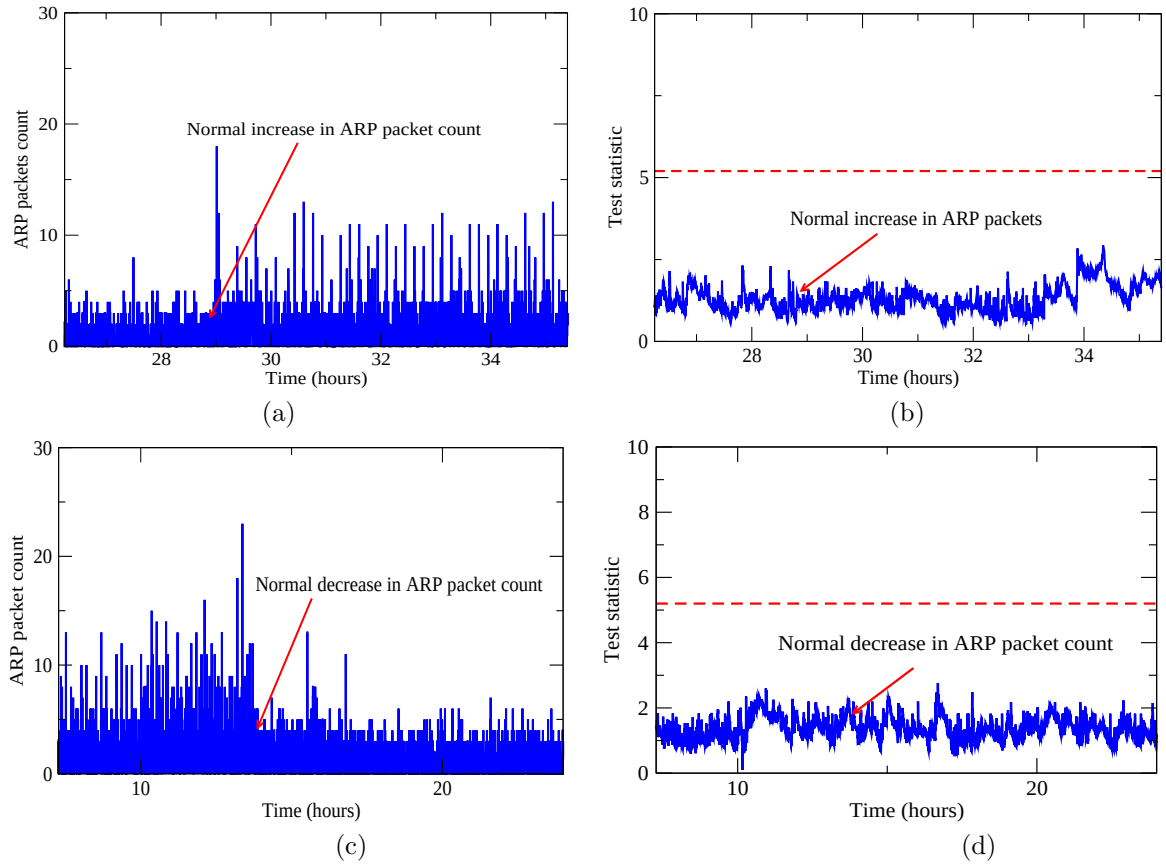


Figure 5: Behavior of the test statistic corresponding to excessive ARP packets, excessive inbound broadcasts, excessive outbound broadcasts, and IP packet loss problems: (a)IP packet discards (b) Outbound broadcast packets (c)Inbound broadcast packets (d) Inbound ARP packets

References

[1] F. Barceló, and J. Jordán. "Channel Holding Time Distribution in Public Telephony System (PAMR and PCS)". *IEEE Transaction on Vehicular Technology*, Vol. 49, No. 5, pp: 1615-1625,September 2000.

[2] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application.*

[3] V. A. Bolotin, Telephone circuit holding time distributions, in The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks (Proc. 14th ITC). Amsterdam, the Netherlands: Elsevier, 1994, vol. 1a, pp. 125134.

[4] V. A. Bolotin Modeling call holding time distributions for CCS network design and performance analysis, IEEE J. Select. Areas Commun., vol. 12, pp. 433438, Apr. 1994.

[5] V. A. Bolotin, Y. Levi, and D. Liu, Characterizing data connection and messages by mixtures of distributions on logarithmic scale, in Teletraffic Engineering in a Competitive World (Proc. 16th ITC). Amsterdam, Prentice-Hall, 1993.

[6] J. Case, M. Fedor, M. Schoffstall and J. Davi. A Simple Network Management Protocol, RFC 1157, 1990.

[7] A. Dempster, N. Laird and D.Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Statist. Soc. B*, Vol. 39, pp:1-38, 1977.

[8] G. Goldzsmidt, Y. Yemini. Distributed Management by Delegation. *15th International Conference on Distributed Computing Systems*, IEEE Computer, 1995.

[9] Hassan Hajji, B. H. Far. Continuous Network Monitoring for Fast Detection of Performance Problems. *Proceedings of 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, 2001.

[10] S. C. Hood and C. Ji. Proactive Network Fault Detection. *Proceedings of the INFOCOM'97*, pp:1147-1155, 1997.

[11] G. Jakobson and M. D. Weissman. Alarm Correlation. *IEEE Network*, pp:52-59, 1993.

[12] I. Katzela and M. Schwarz. Schemes For Fault Identification is Communication Networks. *IEEE/ACM Transactions on Networking*, Vol. 3, pp:753-764, 1995.

[13] F. Kastenholz. Definitions of Managed Objects for the Ethernet-like Interface Types, RFC 1643, 1994.

[14] V. Paxson. Empirically-Derived Analytic Models of Wide-Area TCP Connections. *IEEE/ACM Transactions on Networking*, Vol. 2. No. 4, August 1994.

[15] L. LaBarre. Management by Exception: OSI event generation, reporting, and logging. *Proceedings of Second International Symposium on Integrated Network Management*,1991.

[16] A. Leinwand, K. Fang Conroy. Network management, a practical perspective. 2nd Edition. Addison-Wesley, 1996.

[17] J.P. Martin-Flatin, S. Znaty and J.P. Hubaux. A Survey of Distributed Enterprise Network and Systems Management Paradigms. *Journal of Network and Systems Management*, Vol.7, No.1, pp:9-26, 1999.

Variable	Number of Alarms	Average Alarm Rate per Hour
etherStatsBroadcastPkts	12	0.077
etherStatsUnknownProts	9	0.053
arpStatsPkts	2	0.011
tcpInSyn	2	0.011

Table 3: Average alarm rate per hour for a duration of one week (168 hours)

Variable	Number of Alarms	Average Alarm Rate per Hour
etherStatsBroadcastPkts	13	0.018
etherStatsUnknownProts	89	0.123
arpStatsPkts	3	0.004
tcpInSyn	43	0.059

Table 4: Average alarm rate per hour for a duration of one month (720 hours)

[18] R. A. Maxion and F. E. Feather. A Case Study of Ethernet Anomalies in a Distributed Computing Environments. *IEEE Transactions on Reliability*, Vol. 39, No. 4, pp:433-443, 1990.

[19] K. McCloghrie, M. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II, RFC 1213, 1991.

[20] G. J. McLachlan, and K. E. Basford. *Mixture Models: Inference and Application to Clustering.* New York: Dekker, 1988.

[21] D. Seigmund and E. S. Venkatraman. 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.

[22] M. Thottan and C. Ji. Proactive Anomaly Detection Using Distributed Intelligent Agents. *IEEE International Workshop on Systems Management*, 1998.

[23] D. M. Titterington. Recursive Parameter Estimation using Incomplete. *Journal of Royal Statistics Society*, Serie B, Vol. 46, No. 2, pp:257-267, 1984.

[24] B.D. Ripley. *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

[25] E. Weinstein, M. Feder and A. V. Oppenheim. 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.

[26] S. Yemini, S. Kliger, E. Mozes, Y. Yemini, D. Ohsie. High speed and robust event correlation. *IEEE communication Magazine*, pp 82-90, 1996.