World Applied Sciences Journal 13 (11): 2364-2371, 2011
ISSN 1818-4952

Change Detection Methods for Computer Network Problems

Mouhammd Al-Kasassbeh

Department of IT, University of Mutah, Karak, Jordan

Abstract: In previous research, the detection of network problems was performed using an appropriate subset of Management Information Base (MIB) variables. The changes in the behaviour of the MIB variables were detected using different change detection algorithms; this change gives a clear image about any problem that might occur in the network domain. This paper is concerned with the problem of change detection in network traffic using the most popular change detection online methods. A detailed review of the methodology of these techniques and their advantages and disadvantages are given in this research. Examples of interesting applications that use these methods include network fault management, security management and performance management.

Key words: Log ratio test . auto regressive . general likelihood ratio . cumulative sum

INTRODUCTION

The anomaly detection problem has important applications in many fields, such as; network faults detection, intrusion detection, stock data, digital signal processing, data mining. In [1-4], they performed to study the most common methods in Change Detection (CD). The most promising methods were selected based on their online applicability to capture the abrupt change on time series dataset; moreover these methods are evaluated to capture the change with offline datasets.

The potential use of MIB variables in fault management has been previously explored in schemes that used Bayesian belief networks [5]. However, there is no single MIB variable that is capable of capturing all manifestations of the network abnormality, because a single abnormality can be manifest itself differently under varying circumstances. Therefore, it is necessary to choose a subset of relevant MIB variables. This helps to reduce the complexity of the fault detection algorithms. The details of the MIB structure can be found in RFC 1155 [6]. According to [7], the MIB has 171 variables. These variables fall into 10 groups, depending on the type of the node and the management functions. For each node (server, a switch or a router), there are different groups of MIB variables to observe the statistical and status information. For example, in the case of a server, one can read most of the MIB variables, starting from the low-level, such as the interface group to an application layer group. In the case of a router, it is limited to the network layer Group (IP group) only.

It has been, experimentally, shown that changes in the statistics of traffic data can be generally used to detect faults [2, 8, 9]. Network problems produce some changes in the network traffic that requires methods more than second order statistics to capture and detect the faults [10].

In the following sections is a summary of the most change detection methods such as; Log Ratio Test (LRT), General likelihood Ratio, CUSUM with Bootstrap and others, with some examples using different datasets that show how the changed could be captured and detected.

LOG RATIO TEST (LRT) WITH FIRST ORDER AUTO REGRESSIVE AR (1)

This hypothesis testing is used to determine whether there is a change in the testing window compared to the learning window. The traffic series in these two windows is modeled using first order Auto Regressive AR (1) process, this model has been used to describe time series signals [11, 12].

Figure 1 shows, the Learning window L(t) and the testing window T(t) of length N_L and N_T respectively.

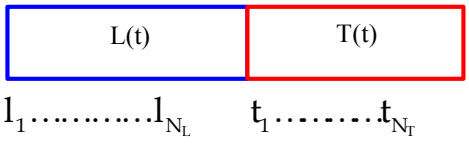


Fig. 1: Piecewise stationary windows

Corresponding Author: Dr. Mouhammd Al-Kasassbeh, Department of IT, University of Mutah, Karak, Jordan.

The Learning window L(t) can be presented as a set of data;

$$L(t) = \{l_1(t), l_2(t), \ldots\ldots\ldots\ldots, l_{N_L}(t)\}$$
$$\tilde{l}_i(t) = l_i(t) - \mu \quad (1)$$

where the μ is the mean of the learning window. We can model the $\tilde{l}_i(t)$ as first order auto regressive model with residual error;

$$\varepsilon_i(t) = \sum_{k=0}^{p} \alpha_k \tilde{l}_i(t-k) \quad (2)$$

where

$$\alpha_k = \{\alpha_1, \alpha_2, \ldots\ldots, \alpha_p\}$$

and $\alpha_0 = 1$ are Autoregressive weight parameters. Supposing each residual value in a stochastic process can be presented as a liner combination of independent random variable series with the same probability distribution, more often than not, these variables follow the normal distribution with expected value zero and variance σ^2, $N(0, \sigma^2)$ [13, 14].

After modeling the traffic series of the adjacent windows, two white noise series will be generated. These noise series follow the $N(0, \sigma^2)$ distribution. Herein we calculate the probability of a change happening between the adjacent windows using Equation 3;

$$l = \left(\frac{1}{\sqrt{2\pi\sigma_L^2}}\right)^{N_L} \left(\frac{1}{\sqrt{2\pi\sigma_T^2}}\right)^{N_T} \exp\left(\frac{-N_L\sigma_L^2}{2\sigma_L^2}\right)\exp\left(\frac{-N_T\sigma_T^2}{2\sigma_T^2}\right) \quad (3)$$

where N_T and N_R denote the number of variables in each window. σ_L^2, σ_T^2 are the variance of the learning and testing window respectively.

The other probability we have is no change and it is performed as follows;

$$l_0 = \left(\frac{1}{\sqrt{2\pi\sigma_P^2}}\right)^{N_L+N_T} \exp\left(\frac{-(N_L+N_T)\sigma_P^2}{2\sigma_P^2}\right) \quad (4)$$

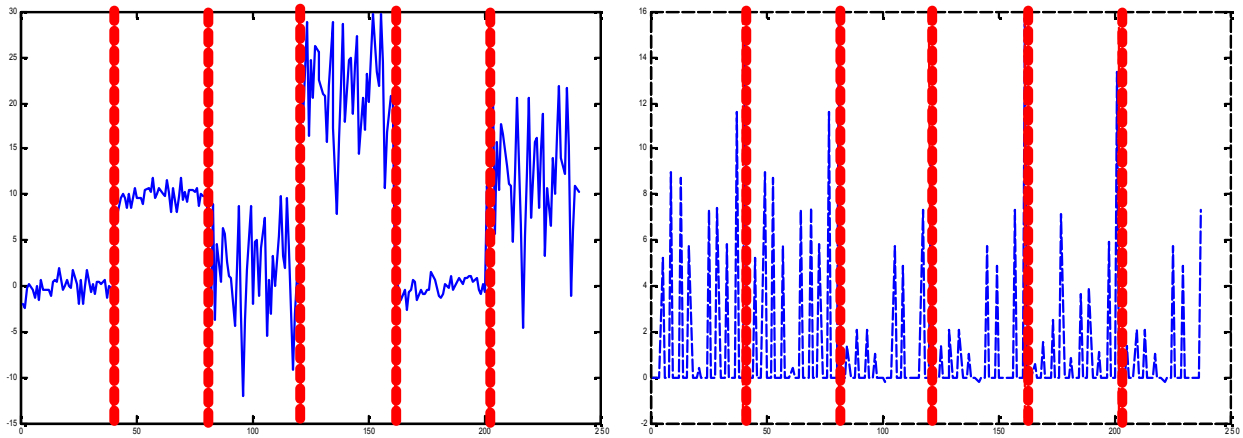
where σ_P^2 is the variance of the combination of the Learning and testing window segments.
Finally, we calculate the log ratio test by;

$$n = \log\frac{l}{l_o} \quad (5)$$

$$n = \log\frac{\left(\frac{1}{\sqrt{2\pi\sigma_L^2}}\right)^{N_L}\left(\frac{1}{\sqrt{2\pi\sigma_T^2}}\right)^{N_T} \exp\left(\frac{-N_L\sigma_L^2}{2\sigma_L^2}\right)\exp\left(\frac{-N_T\sigma_T^2}{2\sigma_T^2}\right)}{\left(\frac{1}{\sqrt{2\pi\sigma_P^2}}\right)^{N_L+N_T} \exp\left(\frac{-(N_L+N_T)\sigma_P^2}{2\sigma_P^2}\right)} \quad (1)$$

After simplifying Equation 6, we can get the following equation;

$$n = (N_L + N_T)\log\sigma_p - N_L\log\sigma_L - N_T\log\sigma_T \quad (2)$$



(a) (b)

Fig. 2: a) Test dataset b) The output of the log ratio test LRT algorithm of testdata

The value of n is directly proportional with the change of the traffic series. Following this, there will be some examples to test this method in real dataset. Two datasets have been used to test this method. The first dataset is created by a random dataset and it has been changed at different points which already known, the mean and the variance have been changed, the red lines in Fig. 2a identify the change points.

As we can see in Fig. 2b, which is the output of the LRT algorithm of TestData, this method efficiently captures the main change points. However, it is complicated when it is calculating the AR parameters. In other words, the process time is high if we compare it with other methods in the same field.

Moreover, this method has been tested on real data, which is the network traffic from live network. Figure 3a depicts one of the MIB variables, that reflect the network traffic. The results of this method are given in Fig. 3b. The dashed vertical lines show the changed that occurred on the traffic.

As illustrated in Fig. 3b, the Log ratio method does not capture all changes in the dataset which obtained from the MIB variables. So we can't fully rely on this method to capture all the changes in the IP traffic. In the next section, we will go through very similar method but more efficient called general likelihood ratio test.

GENERAL LIKELIHOOD RATIO TEST GLR WITH FIRST ORDER AUTO REGRESSIVE AR (1)

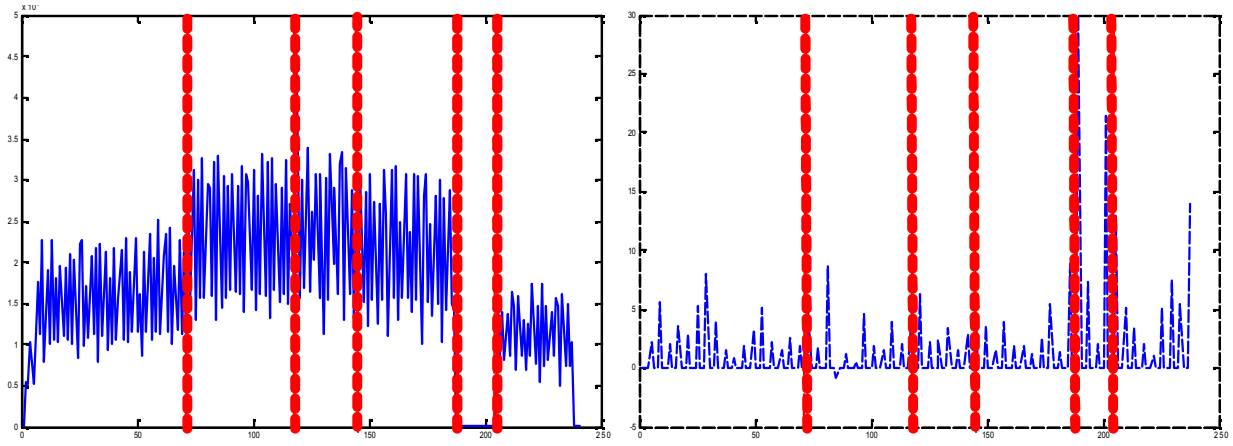
This method is very similar to the LRT method since It uses similar method of piecewise windows same autoregressive AR (1) model and the same hypotheses test. However, to obtain the value of the generalized likelihood ratio test n between 0 and 1, the n is defined as:

$$n = \frac{1}{1+l_o} \tag{3}$$

$$n = \frac{\left(\frac{1}{\sqrt{2\pi\sigma_L^2}}\right)^{N_L}\left(\frac{1}{\sqrt{2\pi\sigma_T^2}}\right)^{N_T}\exp\left(\frac{-N_L\sigma_L^2}{2\sigma_L^2}\right)\exp\left(\frac{-N_T\sigma_T^2}{2\sigma_T^2}\right)}{\left(\frac{1}{\sqrt{2\pi\sigma_L^2}}\right)^{N_L}\left(\frac{1}{\sqrt{2\pi\sigma_T^2}}\right)^{N_T}\exp\left(\frac{-N_L\sigma_L^2}{2\sigma_L^2}\right)\exp\left(\frac{-N_T\sigma_T^2}{2\sigma_T^2}\right)+\left(\frac{1}{\sqrt{2\pi\sigma_P^2}}\right)^{N_L+N_T}\exp\left(\frac{-(N_L+N_T)\sigma_P^2}{2\sigma_P^2}\right)} \tag{4}$$

after simplifying Equation 9, we get;

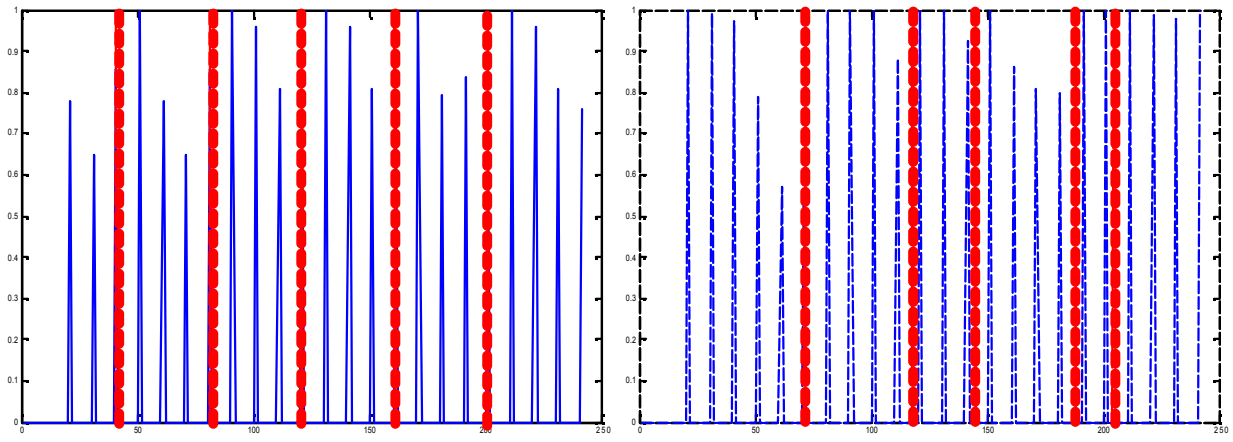
$$n = \frac{\sigma_L^{-N_L}\sigma_T^{-N_T}}{\sigma_L^{-N_L}\sigma_T^{-N_T}+\sigma_P^{-(N_L+N_T)}} \tag{5}$$



(a) (b)

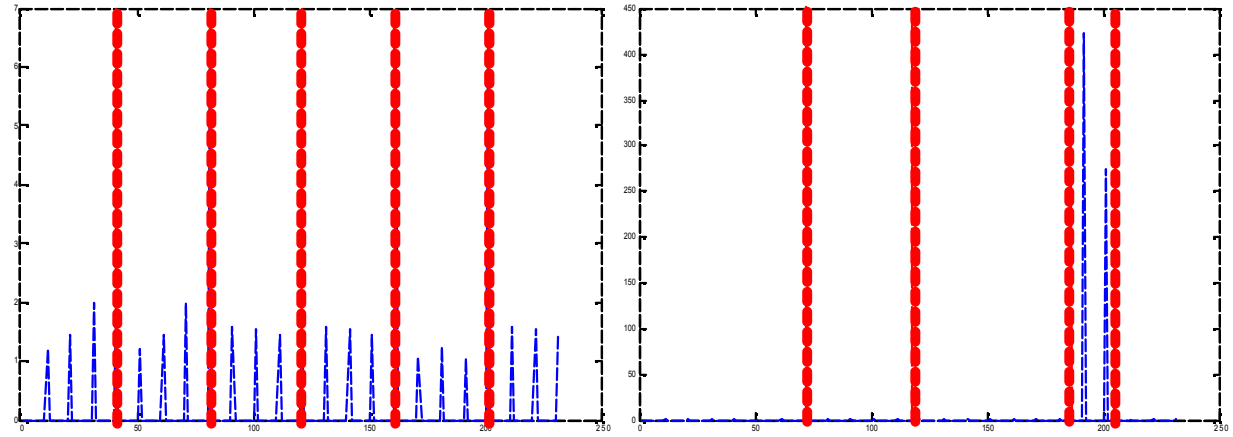
Fig. 3: a) One of our MIB variables (ifInUcastPkts) b) The output of the log ratio test LRT algorithm of ifInUcastPkts

Herein, the change detector is very sensitive; hence, the false alarm rate is very high. Figure 4 Illustrates the output of this method with the same input data that used in the previously.



(a) (b)

Fig. 4: a) The output of GLR algorithm of TestData b) The output of the GLR algorithm of ifInUcastPkts



(a) (b)

Fig. 5: a) The output of standard deviation comparing method of TestData. b) The output of standard deviation comparing method of ifInUcastPkts

In Fig. 4a and 4b are the results from GLR method, Fig. 4a is the output of the testdata that we created as we can see it captured all the change points. And in the Fig. 4b, it is the results of real IP traffic data, this method captured most of the changes as we can see in the vertical dashed lines.

The LRT and GLR are very good to capture most of the changes in the data, but the process time comparatively is longer.

STANDARD DEVIATION COMPARISON

Standard deviation is one of the change detection methods, it shows the change between the two different windows, the Learning window L(t) and the testing window T(t). It checks the value of the variance and the standard deviation and compares them to see the difference. The change n is defined as:

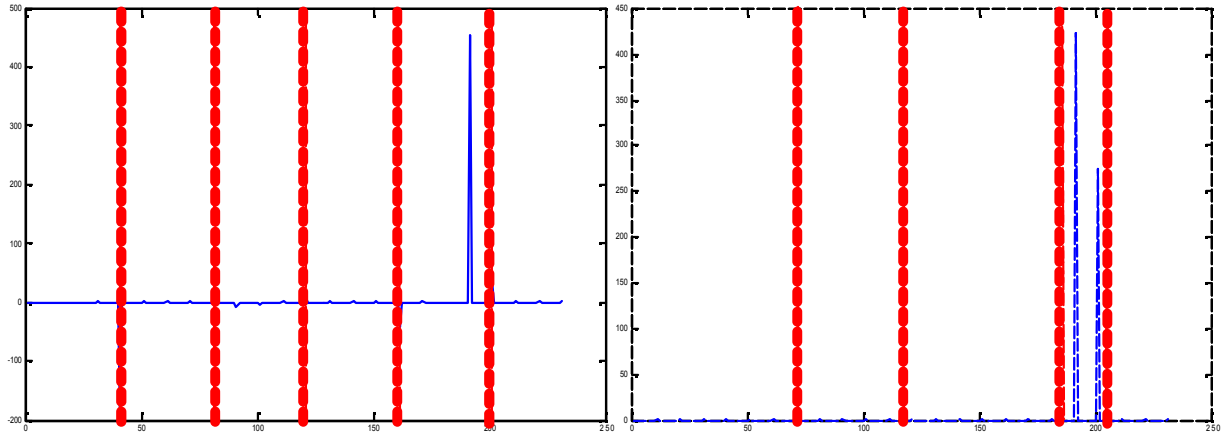
$$n = \frac{\sigma_{MAX}}{\sigma_{MIN}} \quad (6)$$

Figure 5 depicts the outcomes of the standard deviation method of the TestData and the ifInUcastPkts (one of the main MIB's variables). This method gave a good result in the TestData as shown in Fig. 5a, however, it missed many change points in the ifInUcastPkts datasets as shown in Fig. 5b.

MEAN COMPARING

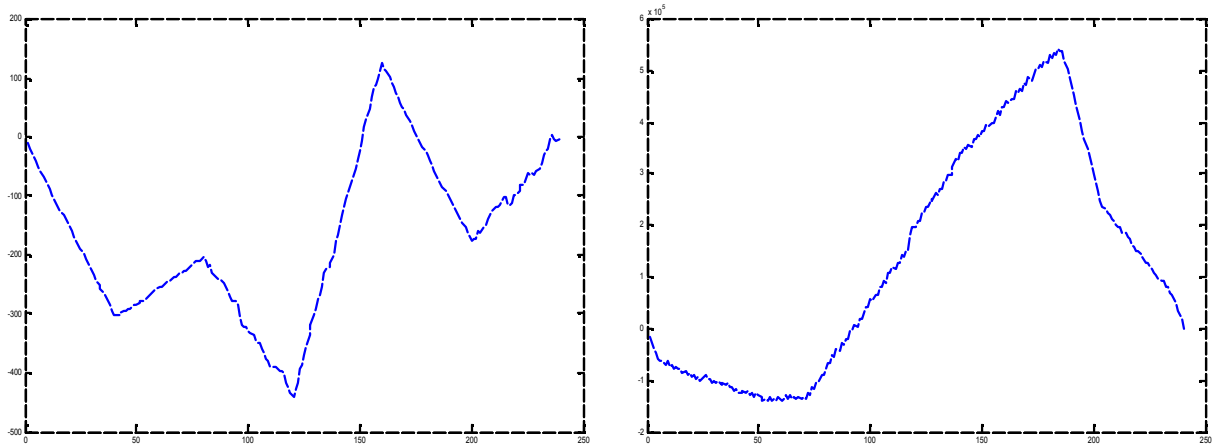
This method is the simplest method to capture the changes between the adjacent windows, once the dataset burst, the average of that dataset will change and increase.

The change n is defined as;



(a) (b)

Fig. 6: a) The output of mean comparing method of TestData. b) The output of mean comparing method of ifInUcastPkts



(a) (b)

Fig. 7: a) CUSUM chart of TestData b) CUSUM chart of ifInUcastPkts

$$n = \frac{\text{Average}_{MAX}}{\text{Average}_{MIN}} \quad (7)$$

as we can see in the Fig. 6, this method captures the change in the mean of the TestData datasets, but it is not powerful for the ifInUcastPkts datasets, since it is non-stationary data and it is hard to trace the changes by using simple method like the mean comparing method.

CUMULATIVE SUM WITH BOOTSTRAP

One of the change point detections used by [15] is cumulative sum (CUSUM) with Bootstrap which, simply, is a combination of CUSUM and bootstrapping to detect changes in the sequences. This method starts its calculation with the CUSUM as below;

Let x_1, x_2, x_3, ……, x_n represent a sequence, from this sequence the CUSUM is S_1, S_2, S_3, ……, S_n are collected as follow;

First step, we calculate the mean;

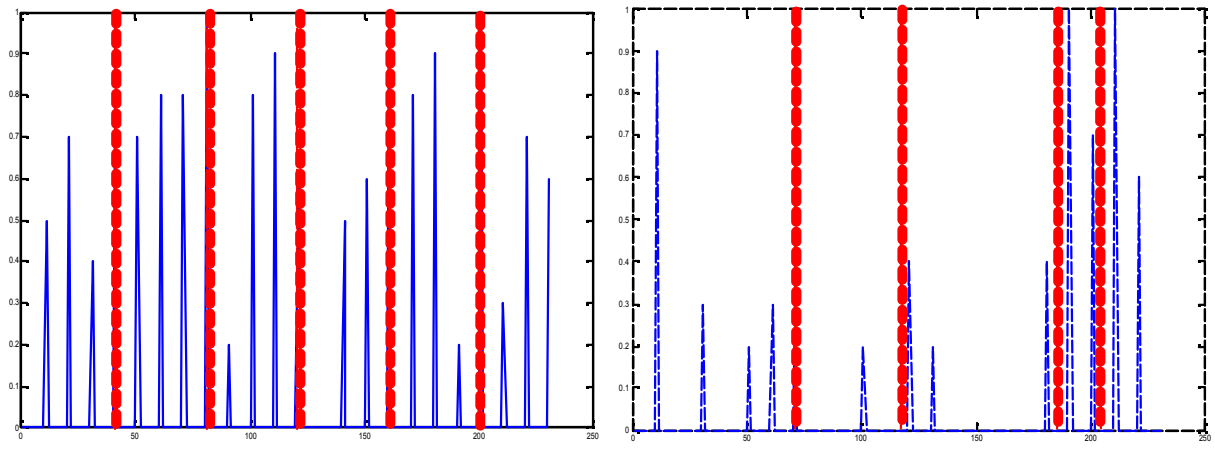
$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_n}{n} \quad (8)$$

Initializing the CUSUM with zero $S_1 = 0$.

Calculating the other CUSUMs using the equation below,

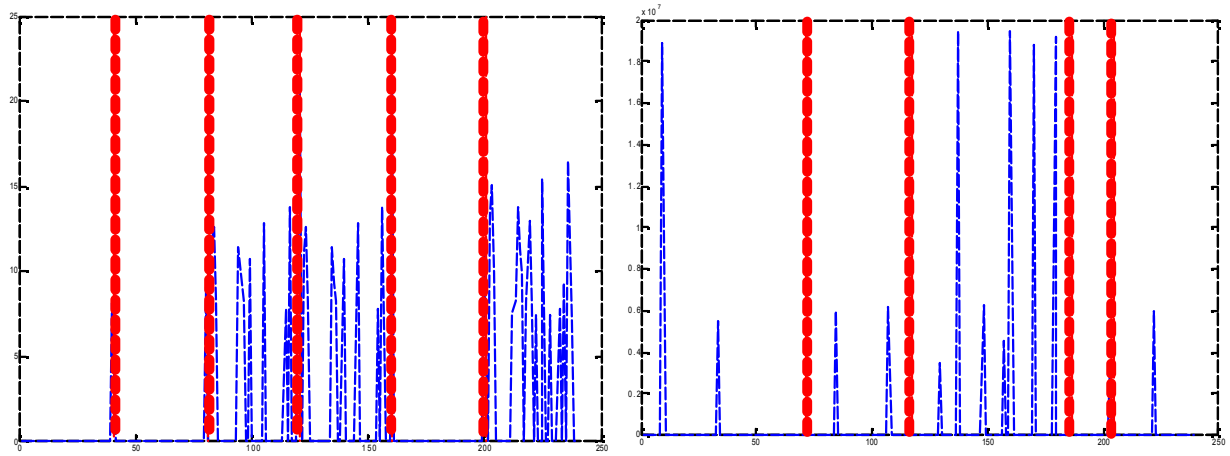
$$S_i = S_{i-1} + (X_i - \bar{X}) \quad (14)$$

the CUSUM series are plotted in Fig. 7a and 7b of our two datasets used in this research.



(a) (b)

Fig. 8: a) The CUSUM with bootstrap method of TestData. b) The CUSUM with bootstrap method of ifInUcastPkts



(a) (b)

Fig. 9: a) The differencing method of TestData. b) The differencing method of ifInUcastPkts

In the Fig. 7, the sudden change in direction of the CUSUM means a sudden change in the average. Now, however this raises a problem how to make sure that the change has been taken place in these points. In order to overcome this problem, [15] suggested using a confidence level by performing a bootstrap analysis; before using this method the magnitude of the change is required. This can be calculated by taking the the difference between the Minimum and the Maximum of the CUSUMs as follow;

$$S_{diff} = S_{Max} - S_{Min} \quad (14)$$

as soon as we calculate the magnitude of the change, the bootstrap can be performed as follows;

- Generate new samples from the original one by randomly reordering them, for example; $x'_1, x'_2, x'_3, \ldots\ldots\ldots, x'_n$;
- Calculate the Bootstrap CUSUM, $S'_1, S'_2, S'_3, \ldots\ldots\ldots, S'_n$
- As done for the original data, we calculate the magnitude of the change by calculating the;

$$S'_{diff} = S'_{Max} - S'_{Min} \quad (16)$$

- Finally, determining whether the bootstrap difference S'_{diff} is less than the original difference S_{diff} or not.

A bootstrap analysis, mainly, performs a large number of bootstraps, as well as the counting of the number of bootstraps for which S'_{diff} is less than S_{diff}. The confidence level that a change occurred as a percentage is calculated as follows:

$$\text{ConfidenceLevel} = \frac{X}{N} 100\% \quad (15)$$

where N is the number of bootstrap samples performed and X is the number of bootstraps for which $S0_{diff} < S_{diff}$.

As we can see in the figures below, this method captures the change in the main change point of the TestData datasets however, it is not capable for the ifInUcastPkts datasets.

DIFFERENCING METHOD

Differencing method is a flexible method for change detection. The difference method is performed as follow;

$$\Delta_i = |x_i - x_{i-n}| \quad (9)$$

where Δ_i is the difference, X_i is the current value and X_{i-n} is the lagged value, the change will be obtained when the value of Δ_i>=thershold. Figure 9 shows the results of this method in the TestData datasets and the ifInUcastPkts datasets. In the first datasets, this method capture the main change points in the mean, but it has vague information about the change in the variance. In the second datasets, this method gives a very low performance to detect the changes in the sequence, because the type of our data is non-stationary, it is too complicated to trace the changes by this simple method.

CONCLUSIONS

In this work, the most popular change detection methods have been examined in details. It has been shown that there is no perfect method that can capture the changes in the network traffic. Each method has its own style and can give good results that depend on the input data. Whatever missing in any one of them, the other methods can find and complete it. In reality, it is hard to combine these methods together, as the process delay will be very high. For online detection in online systems, the delay is a crucial issue, the best system should have the minimum delay in order to give good results in short time. In the future work, more studies will be conducted to find hybrid method to find adaptive detection method that deals with any type of traffic with less complexity and high performance.

REFERENCES

1. Chandola, V., A. Banerjee and V. Kumar, 2009. Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41: 15. DOI: 10.1145/1541880.1541882
2. Kai, H., Q. Zhengwei and L. Bo, Network Anomaly Detection Based on Statistical Approach and Time Series Analysis. in Proc. AINA Workshops, 2009, pp.205-211. DOI: 10.1109/WAINA.2009.58
3. Thottan, M., G. Liu and C. Ji, 2010. Anomaly detection approaches for communication networks. Algorithms for Next Generation Networks, pp: 239-261. DOI: 10.1007/978-1-84882-765-3_11
4. Yu, Y., J. Wang, X. Zhan and J. Song, 2009. Novel anomaly detection approach for telecommunication network proactive performance monitoring. Frontiers of Electrical and Electronic Engineering in China, 4: 307-312. DOI: 10.1007/s11460-009-0051-9
5. Hood, C.S. and C. Ji, 1997. Proactive network-fault detection. Telecommunications, 46: 333-341. DOI: 10.1109/24.664004
6. Leinwand, A. and K.F. Conroy, 1995. Network management: A practical perspective: Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA. ISBN:0-201-60999-1
7. Rose, M.T., 1991. The Simple Book: An Introduction to Management of TCP/IP-based Internets: Prentice Hall. ISBN: 0138126119
8. Fassois, S.D. and J.S. Sakellariou, 2007. Time-series methods for fault detection and identification in vibrating structures. Philosophical Transactions A, 365: 411. DOI:10.1098/rsta.2006.1929
9. Li, X., F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone and A. Lakhina, 2006. Detection and identification of network anomalies using sketch subspaces. DOI: 10.1145/1177080.1177099
10. Hood, C.S. and C. Ji, 1997. Proactive network-fault detection telecommunications. IEEE Transactions on reliability, 46: 333-341. DOI: 10.1109/24.664004
11. Box, G., G.M. Jenkins and G. Reinsel, 2006. Time Series Analysis: Forecasting and Control. Beijing: Posts & Telecom Press. DOI: 10.1109/TAC.1972.1099963
12. Brockwell, P.J. and R.A. Davis, 2009. Time series: Theory and methods: Springer Verlag. ISBN 978-0-387-97429-3
13. Appel, U. and A.V. Brandt, 1983. Adaptive sequential segmentation of piecewise stationary time series. Information Sciences, 29: 27-56. DOI: 10.1016/0020-0255(83)90008-7

14. Mann, H.B. and A. Wald, 1943. On the Statistical Treatment of Linear Stochastic Difference Equations. Econometrica, 11: 173-220. DOI: 10.1214/08-EJS290

15. Taylor, W.A., Change-point analysis: A powerful new tool for detecting changes. Preprint, available as http://www.variation.com/cpa/tech/changepoint.html.