Improving Network Traffic Anomaly Detection for Cloud Computing Services

Ana Cristina Oliveira[*][†], Marco Spohn[†][‡], Reinaldo Gomes[†], Do Le Quoc[§] and Breno Jacinto Duarte[¶]
[*]Research Group on Convergent Networks (GPRC) - Federal Institute of Paraíba (IFPB)
Campina Grande, Paraíba, Brazil
[†]Systems and Computing Department (DSC) - Federal University of Campina Grande (UFCG)
Campina Grande, Paraíba, Brazil
[‡]Federal University of Fronteira Sul (UFFS)
Chapecó, Santa Catarina, Brazil
[§]Systems Engineering Group (SE Group) - Technical University of Dresden (TU Dresden)
Dresden, Germany
[¶]Research Group on Usable Security and Ubiquitous Communications - Federal Institute of Alagoas (IFAL)
Maceió, Alagoas, Brazil
Emails: ana.oliveira@ifpb.edu.br, {maspohn, reinaldo}@dsc.ufcg.edu.br,
do@se.inf.tu-dresden.de, brenojac@ifal.edu.br

Abstract—**Efficient network traffic anomaly detection is a widely studied problem on avoiding attacks and unwanted use of communication infrastructures. Existing techniques to detect, prevent or monitor these attacks are usually based on known thresholds, on the construction of profiles of normal traffic patterns, or on signature pattern matching of anomalous behavior (i.e., viruses and attacks). On the other hand, there are dynamic techniques that strive to predict the system's clutter degree; i.e., the system entropy, supposing that outliers translate to anomalies. We have developed and analyzed the accuracy of a network anomaly detector for Cloud Computing Systems based on the entropy of network traffic metrics. Although entropy-based solutions do not suppose hard knowledge of the system, the results point out to the need for more accurate adjustment of system parameters, taking into consideration the nature of the data, frequency of events, and the variation of metric values. To improve the results, unsupervised machine learning algorithms were added to the anomaly detection process.**

***Keywords*–Network traffic anomaly detection; Cloud Computing; Entropy; Machine learning.**

I. INTRODUCTION

Network traffic analysis in cloud environments is one of the most important tasks in cloud management to guarantee the quality of services, validate performance of new applications and services, build accurate network models and detect anomalies in the cloud. The network traffic produced by cloud computing systems reveals users' behavior regarding service utilization, once all services are accessed via the network. Traffic analysis and the recognition of all significant application flows are important tools for modeling service usage, building up patterns for identifying normal system operations [1].

Additionally, network communication between cloud provider and its customers affects significantly the performance of most cloud-based applications [2]. Analyzing network traffic will provide insights on the performance and behavior of application and services deployed in clouds. Therefore, it is necessary to develop network traffic measurement and analysis techniques to improve availability, performance and security in cloud computing environments.

On the other hand, managing and analyzing network traffic of large scale cloud systems is a challenging task. The techniques used to monitor and analyze traffic in conventional distributed systems differ from cloud computing systems. In conventional approaches, assumptions are made that network flows follow some patterns, which is acceptable for corporate applications, but cloud applications may have significant changes in traffic patterns [3].

The term **anomaly** is fairly generic and it covers attacks, unwanted traffic on the network due to misbehaving applications, packet loss, and undesired traffic injection from not allowed applications. In this context, one question is raised: can we actually have a monitoring system able to detect any sort of network anomaly without looking specifically for it? There are works that proposed the idea of identifying any type of anomaly by monitoring metrics' entropy [4][5][6]; i.e., analyzing the behavior of an application by monitoring the degree of concentration or dispersion of the target metrics' distribution.

In this paper, we focus on investigating techniques to detect anomalies within cloud computing network traffic. We adapted and implemented the Entropy-based Anomaly Testing (EbAT) methodology into the context of cloud computing network traffic monitoring [7]. We strived to identify if EbAT is suitable for monitoring cloud computing systems. Considering it is a scalable and lightweight technique, which is a non functional requirement for cloud computing systems, since they strongly depend on the network support in large scale.

We concluded that the EbAT technique by itself can be improved to monitor network traffic, especially for dynamic systems that may change the network load quickly. To improve the anomaly detection accuracy for cloud computing network traffic, we developed a new lightweight approach based on the EbAT method and unsupervised machine learning anomaly detection.

The contributions of this work are fourfold: (i) implementation and analysis of the EbAT technique applied to cloud computing network traffic; (ii) feasibility analysis of the entropy-based method to anomaly detection of cloud computing network traffic; (iii) implementation and analysis of unsupervised machine learning technique for anomaly detection; (iv) proposal and implementation of a novel lightweight method to improve network traffic anomaly detection for cloud computing systems.

The remainder of this paper is organized as follows. Related work is presented in Section II. The entropy-based and the machine learning anomaly detection methods are described in Section III. The novel approach is proposed in Section IV. The validity of the proposed technique is addressed in Section V. To conclude, Section VI contemplates final remarks.

II. Related Work

Wang [7] proposed a method for online generic anomaly detection based on the entropy of any sort of metric distribution or composition of metrics. Such technique is called EbAT. The results are achieved by establishing entropy time series resulting from visual spike detection, or wavelet analysis, instead of the observation of individual thresholds for the metrics. However, it assumes that some parameters are statistically estimated.

Wang *et al.* [5] conducted an experiment to demonstrate the feasibility and accuracy of the EbAT method in comparison with threshold-based anomaly detection procedures. They injected faulty operations at the application level, which were analyzed using CPU and memory metrics, and correlation between read and write operations at virtual disks.

Wang *et al.* [6] compared the EbAT technique and Gaussian model for anomaly detection of system metrics (e.g., CPU and memory). According to their study, the Gaussian model presented lower values for the *recall* metric. Notwithstanding, we believe those techniques may be applied together to support the anomaly detection decisions.

Benetazzo *et al.* [8] proposed the analysis of aggregate traffic by determining empirical rate-interval curves (RICs), which consist of dividing the flow measurements in quantiles, striving to delineate scaling properties and other metrological diagnostics. The RIC-based method characterizes network traffic without requiring a priori knowledge of the underlying flow model; however, the proposed method is quite costly.

Nychis *et al.* [9] analyzed the anomaly detection ability of different entropy-based metrics. They pointed out that the port and address distributions are strongly correlated to the detection capacity. In other words, both metrics provide similar results when detecting network traffic anomalies. In addition, the metrics have limited utility in detecting port scan attacks and flood attacks. The authors also found that behavioral metrics are less correlated with other metrics. However, their work was applied in a university network backbone; thus, the characteristics of the metrics would be different from a cloud computing environment.

Quan *et al.* [10] compared two entropy methods, network entropy and normalized relative network entropy (NRNE) to classify network behaviors. Two different probability distributions could share the same entropy value, even having discrepant probability vectors, and it is a problem regarding the technique. To avoid this problem, the authors employed the concept of relative entropy, or Kullback-Leibler (KL) deviation, which represents the difference between two probability distributions. The NRNE performed better; on the counterpart, it demands more input attributes to detect abnormal network behaviour.

Smith *et al.* [11] proposed an autonomic mechanism for detecting anomalies in cloud computing systems similar to EbAT. The authors defined a set of techniques involving data transformation to standardize the data format for analysis, an extraction phase to reduce data size, and unsupervised learning using clustering algorithms to detect which nodes are behaving in a different manner from the others (outliers). The anomalies are computed based on the system behavior.

III. Fundamentals

A. Entropy-Based Anomaly Detection

We may divide the EbAT [5][7] technique in three steps: (a) metric collection; (b) construction of entropy time series; and (c) processing of entropy time series. This technique is metric-independent, i.e., we may collect and analyze the most important metrics related to network traffic, or to application-level performance, for instance. Those steps are described in the following sections.

1) Construction of Entropy Time Series: The metrics being analyzed at a moment are placed into a look-back window of size n, where n means the number of samples for the metric (or metrics) that will take part of the analysis. That window slides as the new metrics are being produced.

Note that multiple types of metrics may be monitored and analysed altogether. Before constructing the entropy time series, the data is pre-processed. This pre-processing consists of normalizing and binning the data. Those phases are characterized as follows:

Data Normalization Phase: it consists of dividing all sample values in the current look-back window by the mean of all values of the same type that belong to the window in question.

$$s^{'}_{i,j} = \frac{s_{i,j}}{\frac{1}{n}\sum_{i=1}^{n} s_{i,j}} \tag{1}$$

Where:

- $i = \{1,..,n\}$ represents the index of the sample that will be binned in the look-back window;
- $j = \{1,..,k\}$ represents the index of the metric, and k is the number of metrics that are being monitored;
- $s_{i,j}$ is the i-th sample value of the window of the *j-th* metric;
- $s^{'}_{i,j}$ is the i-th normalized sample value of the window of the *j-th* metric.

Data Binning Phase: it takes all normalized values and inserts them into a bin, which represents an interval for the data. The equation that represents the binning is:

$$b_{i,j} = \begin{cases} m, & s'_{i,j} > r \\ \left\lfloor \dfrac{s'_{i,j}}{r/m} \right\rfloor, & s'_{i,j} \leqslant r \end{cases} \tag{2}$$

Where:

- $b_{i,j}$ is the corresponding bin index for $s'_{i,j}$;
- r is the range of the normalized data. An $[0, r]$ interval is defined to represent the most representative data;
- m is the greatest bin index. Since the first index is 0, than there are *m+1* bins.

The bin index, $b_{i,j}$, of the normalized sample value $s'_{i,j}$ will be the last bin index, *m*, if the sample value is greater than the range expected, r. The greatest sample values of the look-back window are placed into the m bin. The rest of the values are placed into the bins in the range $[0, r]$. To choose the adequate bin for those remaining values, we divide the normalized sample value by the ratio, which give us an idea of a fair placement of the values into m equal sized intervals. Finally, the bin index is the floor value of this division, as shown in (2).

Let C be the set of metrics being monitored. We may define $C = \{c_1, .., c_k\}, k = 1 \ .. \ |C|$. We may monitor any number of metrics. For each metric being monitored, there is one sample value, and one corresponding bin. Examples of metrics at application-level are CPU and memory, and at network-level we may consider delay, bandwidth, and jitter, for instance.

Event Creation Phase: after normalizing and binning the data, the next step deals with representing the metrics as events. Those events are named *measurement events*, or *m-events*. One event, e_i, is a vector that contains the bins of all the *j* metrics analyzed. One m-event is defined as: $e_i = \langle b_{i,1}, b_{i,2}, ..., b_{i,k} \rangle$.

Entropy Computation and Aggregation Phase: we will compute the entropy of the events. Then, we may define *E* as the set containing all events in the current look-back window as:

$$E = \{e_1, e_2, ..., e_v\}$$

Where:

- Let v be the number of distinct events in the look-back window, then $v \neq n$ if there is more than one equal event;
- The event e_a is equal to event e_b if $b_{a,j} = b_{b,j}; \forall j \in [1, k], \forall b_{a,j} \in e_a, \forall b_{b,j} \in e_b, k = |e_a| = |e_b|$.

We will compute the entropy of E, $H(E)$. Firstly, we will count the number of occurrences of event e_i and represent it by $n_i, \forall i = [1, v]$. In the sequence, the local entropy may be calculated as stated in (3), where ${}^{n_i}/_{n}$ is the probability of occurring the event e_i [4].

$$H(E) = -\sum_{i=1}^{v} \frac{n_i}{n} log \frac{n_i}{n} \tag{3}$$

2) Processing of Entropy Time Series: The processing of the entropy time series consists of applying one or more methods striving to find out anomalous patterns. Those methods may be a combination of spike detection, signal processing, and subspace analysis, for instance.

B. Machine Learning Anomaly Detection

The unsupervised machine learning technique for anomaly detection technique is based on fitting the data to a Gaussian Distribution. The values with very low probability are considered anomalies. The goal of this probability analysis is to find out a probability threshold that maximizes the detection accuracy. In this section we will describe how one can implement such a technique.

We start by collecting measurements of the features (or *metrics*, in the context of network performance) that we call *training set* (TS). The next step is to fit a Gaussian distribution on the TS, calculating the probability of every value (the pair of features).

Given a *training set* $x^{(1)}, ..., x^{(m)}$ (where $x^{(i)} \in R_n$), let us estimate the Gaussian distribution for each of the features x_i. For each metric $i = 1, ..., n$, we need to find the parameters μ_i and σ_i^2 that fit the data in the i-th dimension $x_i^1, ..., x_i^m$ (i.e., the samples collected for metric i). The Gaussian distribution is given by (4), where μ is the mean and σ^2 is the variance. We estimate the parameters μ_i and σ_i^2 of the i-th metric by using (5) and (6), respectively [12].

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4}$$

$$\mu_i = \frac{1}{m} \sum_{j=1}^{m} x_i^{(j)} \tag{5}$$

$$\sigma_i^2 = \frac{1}{m} \sum_{j=1}^{m} (x_i^{(j)} - \mu_i)^2 \tag{6}$$

After calculating the mean and variance, we fit the data to the Gaussian model. Then, we observe which values have a very high probability according to the Gaussian distribution, and which have a very low probability. The low probability samples are anomalies. We predict which metric samples are anomalies by defining a *threshold*. We choose the threshold, ϵ, that maximizes the accuracy on a *cross validation set* [12].

Let the cross validation set be $CV = \{(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})\}$, where the label $y = 1$ corresponds to an anomalous metric sample, and $y = 0$ corresponds to a normal sample. For each cross validation element, we computed $p(x_{cv}^i)$, which is the mass probability of that element according to the Gaussian distribution. Let the vector of all of these probabilities be $P = \left\langle p(x_{cv}^{(1)}), \ldots, p(x_{cv}^{(m_{cv})}) \right\rangle$.

We define the threshold probability, ϵ, by selecting it at a range from the minimum and maximum values for $p(x_{cv}^i) \in P$. The ϵ value that maximizes the accuracy will be chosen as an anomaly indicator to the detection process [12].

IV. Efficient Entropy-based and Machine Learning-based Anomaly Detection

We argue that the anomaly detection based on the Gaussian model may help to set up the input parameters of the EbAT. We may use the EbAT to label the cross validation set adaptively. On the other hand, as time passes by, both techniques will work in synergy.

It is difficult to configure the input parameters of the EbAT technique, and to validate the identified alarms. On the other hand, from the machine learning perspective, it is difficult to label the samples without prior knowledge.

The proposed technique has 5 phases: (i) traffic capture; (ii) threshold estimation with machine learning; (iii) cloud services' monitoring; (iv) entropy estimation; (v) anomaly detection. We have summarized the processes of both techniques, described in Sections III-A and III-B, and how they interact in those phases in Figure 1. The idea is that the alarms generated by the EbAT will feed the subprocess of labeling the anomalous traffic packets, and that the threshold obtained by the machine learning-based technique will also become a proof when testing if the service metrics contains or not anomalous packets.

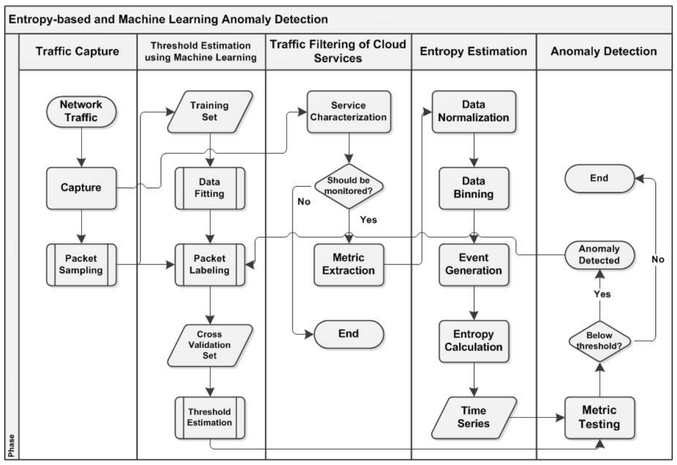


Figure 1. Proposed anomaly detection method.

V. Validation

A. Methodology

We are going to validate the accuracy of the detection system using the *F-measure* metric, also named *accuracy*, or *F1-score*. It represents the harmonic mean of the *precision* and *recall* metrics [13]. Those metrics are described along this section.

The **precision** metric is the ratio of the anomalies correctly detected and the total number of anomalies detected, either correct or wrong, shown in (7). Then, it characterizes the percentage of correctly detected anomalies; i.e., if a prediction algorithm has precision of 90%, then we understand that 90% of alerts are correct, and, thus, 10% of them are false positives, or *false alarm rate* (FAR, *1-Precision*).

$$Precision = \frac{\#\ of\ successful\ detections}{\#\ of\ total\ alarms} \tag{7}$$

The **recall** metric, in turn, represents the ratio of anomalies correctly detected and the actual number of anomalies, as shown in (8). For example, if the recall metric is 55%, it denotes that 55% of the abnormalities were detected; consequently, there were 45% of missing alerts.

$$Recall = \frac{\#\ of\ successful\ detections}{\#\ of\ total\ anomalies} \tag{8}$$

The analysis of those two metrics better expresses the degree of quality of the anomaly detector. How can we interpret the precision and recall metrics? When the same weights are assigned to the two metrics, one obtains the value of **F-measure** by (9). The larger the value of F-measure, the higher the quality predictor.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9}$$

B. Experiment Description

To analyze the behavior of the system, we adopted the model 2^3-factorial experimental design, which makes up a total of 8 different treatments. For each treatment, we have analyzed 3 factors (n, m, and r), each one varying at two levels, according to Table I. We have analyzed how the three factors and their interactions influenced the overall accuracy.

TABLE I. ANALYSED TREATMENTS.

Parameter	# Scenario							
	1	2	3	4	5	6	7	8
n	5	5	5	5	10	10	10	10
m	6	6	7	7	6	6	7	7
r	5	10	5	10	5	10	5	10

1) DoS Attack to Cloud Computing Services: We performed a VM-to-VM attack using the open-source tool Hping3 [14] within a cloud system running one application called Nutch, which digs up the Web searching for pages. The tool Hping3 allows us to generate arbitrary packets to flood a victim host. We set Hping3 on three VMs in a cloud to generate TCP SYN packets of the Hadoop application targeted to attack Hadoop ports on two victim VMs. The VMs are part of the same cloud, including one master node and one slave node, as shown in Figure 2. The timestamp of the attacks are detailed in Table II [15].

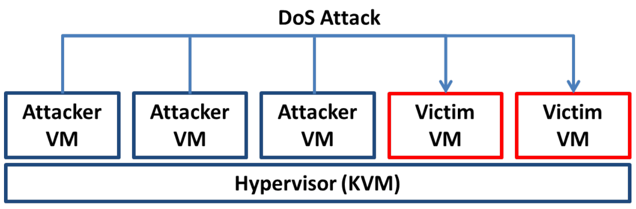


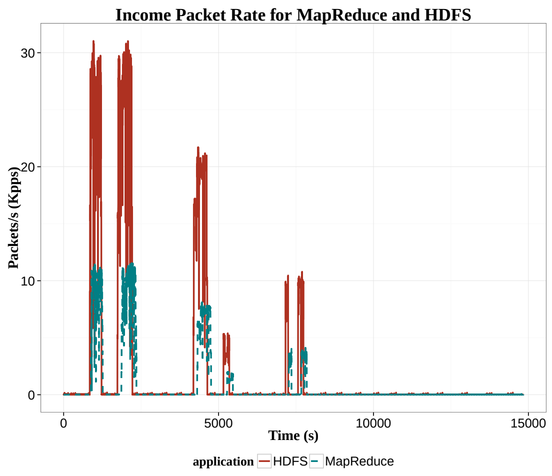
Figure 2. Nodes that are part of the DoS attack [15].

Then, we have measured the following metrics: (i) number of packets generated by the Hadoop Distributed File System (HDFS); (ii) number of packets generated by the MapReduce

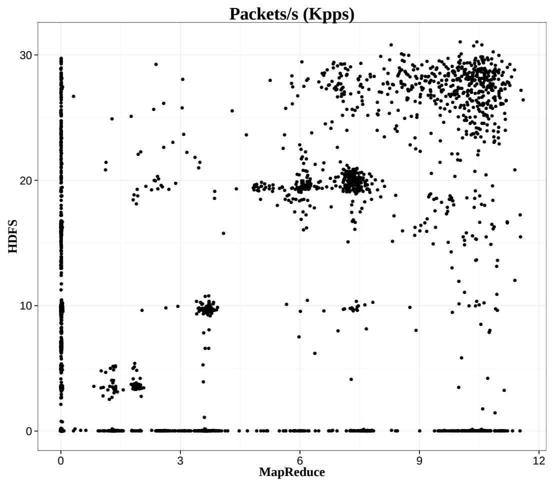
TABLE II. TCP SYN FLOOD ATTACK LIST [15].

	Start Time	Finish Time	Attackers	Victims
1	18:23:32	18:29:10	VM3, VM4, VM5	VM1, VM2
2	18:40:14	18:47:58	VM3, VM4, VM5	VM1, VM2
3	19:20:35	19:28:15	VM4, VM5	VM1
4	19:36:29	19:39:35	VM3	VM1, VM2
5	20:09:37	20:11:06	VM4	VM1
6	20:16:36	20:19:08	VM4	VM1

application. The packet rate of those two applications is shown in Figure 3(a), and the scatterplot of the Map Reduce versus HDFS application packets is shown in Figure 3(b). We may observe that the packet rate series shown have correlation, however they are not synchronized.



(a) time series

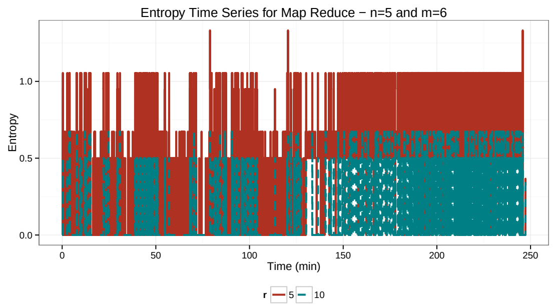


(b) scatterplot

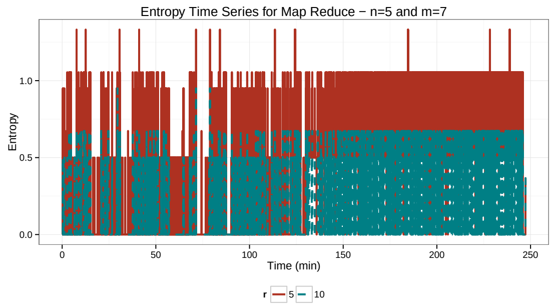
Figure 3. MapReduce and HDFS packet rate.

We have calculated the entropy values for the pair of metrics measured in Figure 4. We may observe that the results are visually difficult to interpret, and establish the right parameters to propose the anomalous values.

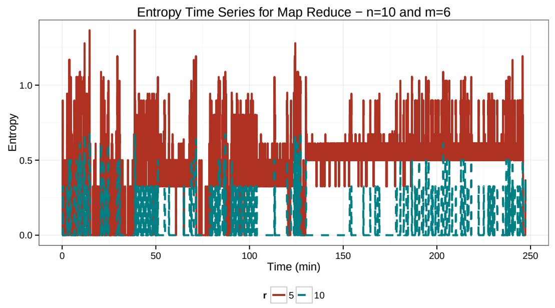
Table III summarizes the results presented in Figure 4.



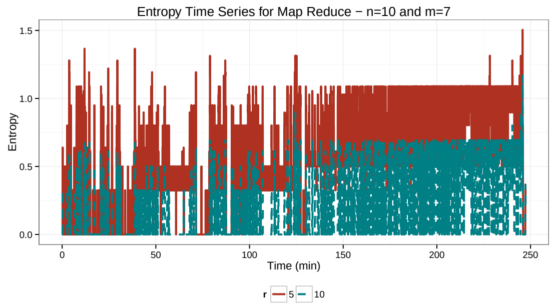
(a) $n = 5$; $m = 6$



(b) $n = 5$; $m = 7$



(c) $n = 10$; $m = 6$



(d) $n = 10$; $m = 7$

Figure 4. Entropy time series with: $n = 5$ a, b); $n = 10$ (c, d).

We observe that the scenario 3 has the best results, i.e., the precision is 100 %. Although, there are others that provide good accuracy results as well, e.g., over 90 %.

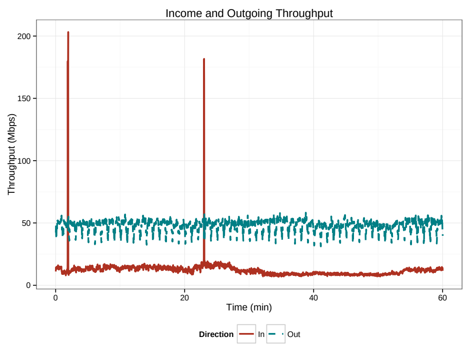
2) DoS Attack to a Network Provider Backbone: We have validated our work using a trace from the *Pohang University*

TABLE III. SUMMARY OF THE RESULTS OF THE ANOMALY DETECTION METRICS OF THE CLOUD DOS.

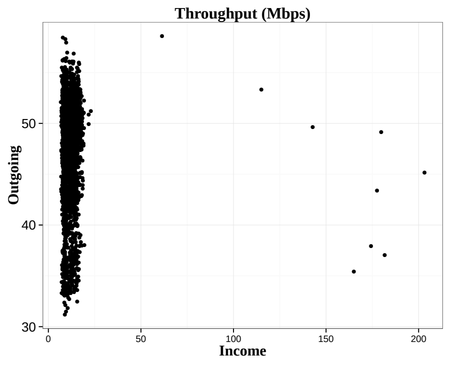
#Scenario	TP	FP	FN	Recall	Precision	F1
1	9967	711	1077	0.902481	0.933414	0.917687
2	543	0	10501	0.0491670	1	0.093726
3	**11044**	**0**	**0**	**1**	**1**	**1**
4	2181	0	8863	0.1974828	1	0.329830
5	10397	1749	647	0.941416	0.856002	0.896680
6	153	9	10891	0.013854	0.944444	0.027308
7	10638	1773	406	0.963238	0.857143	0.907099
8	2402	155	8642	0.217494	0.939382	0.353209

of Science and Technology (POSTECH) that contains traffic of a famous DDoS attack to government and commercial websites in South Korea in July 7th, 2009. Those attacks were probably launched by a special cyber warfare unit belonging to North Korean Army. During the attack, many computers in POSTECHs network campus were zombies. We have analysed one hour of network capture that contains the packets from the attack [16].

We measured and analysed two features of the network traffic: (i) the income throughput, and (ii) the outgoing throughput. The throughput series are depicted in Figure 5(a), and the scatterplot is in Figure 5(b). For this trace, we have applied the Gaussian model on a cross validation set to identify the threshold probability of having an anomalous sample. Then, we have used this value to predict which packets took part of the DoS attack.



(a) time series



(b) scatterplot

Figure 5. Income and outgoing throughput.

We have summarized the mean results obtained for the 8 different treatments in Table IV. We may realize that there are two treatments that contributes to the best accuracy results, which are the first and third scenario (in bold font). In this sense, we restricted our scope, and may also choose setup parameters from one of them both that lead this particular system to provide the best predictions about the presence of anomalies in the traffic. In our case, we selected the third scenario, which parameters are $n = 5$, $m = 7$, and $r = 5$. At those configurations, the F-measure reached 100 %.

TABLE IV. SUMMARY OF THE RESULTS OF THE ANOMALY DETECTION METRICS OF THE NETWORK PROVIDER.

#Scenario	TP	FP	FN	Recall	Precision	F1
1	**13**	**0**	**0**	**1**	**1**	**1**
2	0	0	13	0	NA	NA
3	**13**	**0**	**0**	**1**	**1**	**1**
4	0	0	13	0	NA	NA
5	12	9	1	0.923077	0.571429	0.705882
6	5	2	8	0.384615	0.714286	0.5
7	10	8	3	0.769231	0.555556	0.645161
8	0	0	13	0	NA	NA

We found out that the accuracy of the entropy-based anomaly detection mechanism itself directly depends on a further analysis of the traffic. Those assumptions, however, are too strong for a detection system and constitute barriers to the implementation of such a technique.

VI. Final Remarks

Cloud computing systems have peculiar characteristics, such as aggregation of many different services, which makes it difficult to classify applications either by techniques based on packet payload signature matching, or probabilistic methods for the identification of traffic patterns, and profile of traffic normal behaviors. Those characteristics bring up challenges regarding online traffic monitoring and analysis, which should be done at wire speed while digging a high volume of data.

Choosing one optimal traffic anomaly detection technique is a complex task, because in order to have good results, we may have to know several characteristics of the traffic that are not known in practice. Another challenge is to develop high performance network packet sampling, and speeding up data processing, since the volume of traffic that traverses the cloud service provider is of the order of gigabits per second.

The obtained results show that when applying the EbAT itself, there is still the need to better analyze and comprehend the traffic patterns, finding out the normal behavior of the monitored systems, based on predictions using historical data, or feedback from experts on the business and network traffic, or by making new assumptions regarding the traffic. In this sense, we argue that the accuracy results may be improved by the aid of probability models, such as anomaly detection using the Gaussian model.

As a conclusion, we found out that it was still necessary to investigate new solutions to the cloud computing network anomaly detection problem. As future work, we intend to improve the anomaly detection mechanism by developing and analyzing new techniques to increase the detection accuracy, and to propose a parallel model for capturing and processing the network packets at wire speed.

REFERENCES

[1] A. C. Oliveira, H. Chagas, M. Spohn, R. Gomes, and B. J. Duarte, “Efficient network service level agreement monitoring for cloud computing systems (to appear),” in Computers and Communications (ISCC), 2014 IEEE Symposium on, June 2014.

[2] S. Shetty, “Auditing and analysis of network traffic in cloud environment,” in Proceedings of the 2013 IEEE Ninth World Congress on Services, ser. SERVICES ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 260–267. [Online]. Available: http://dx.doi.org/10.1109/SERVICES.2013.42

[3] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” Journal of Internet Services and Applications, vol. 1, no. 1, Apr. 2010, pp. 7–18. [Online]. Available: http://www.springerlink.com/index/10.1007/s13174-010-0007-6

[4] C. Wang, “Ebat: online methods for detecting utility cloud anomalies,” in Proceedings of the 6th Middleware Doctoral Symposium, ser. MDS ’09. New York, NY, USA: ACM, 2009, pp. 4:1–4:6. [Online]. Available: http://doi.acm.org/10.1145/1659753.1659757

[5] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan, “Online detection of utility cloud anomalies using metric distributions,” in 2010 IEEE Network Operations and Management Symposium - NOMS 2010. Ieee, 2010, pp. 96–103. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5488443

[6] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, “Statistical techniques for online anomaly detection in data centers,” in Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on, May 2011, pp. 385–392.

[7] S. C. Wang, K. Q. Yan, and S. S. Wang, “Achieving High Efficient Agreement with Malicious Faulty Nodes on a Cloud Computing Environment,” Industrial Engineering, 2009, pp. 3–8.

[8] L. Benetazzo, G. Giorgi, and C. Narduzzi, “On the analysis of communication and computer networks by traffic flow measurements,” Instrumentation and Measurement, IEEE Transactions on, vol. 56, no. 4, Aug 2007, pp. 1157–1164.

[9] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, “An empirical evaluation of entropy-based traffic anomaly detection,” in Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement, ser. IMC ’08. New York, NY, USA: ACM, 2008, pp. 151–156. [Online]. Available: http://doi.acm.org/10.1145/1452520.1452539

[10] Q. Quan, C. Hong-Yi, and Z. Rui, “Entropy based method for network anomaly detection,” in Dependable Computing, 2009. PRDC ’09. 15th IEEE Pacific Rim International Symposium on, Nov 2009, pp. 189–191.

[11] D. Smith, Q. Guan, and S. Fu, “An Anomaly Detection Framework for Autonomic Management of Compute Cloud Systems,” 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, Jul. 2010, pp. 376–381. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5615245

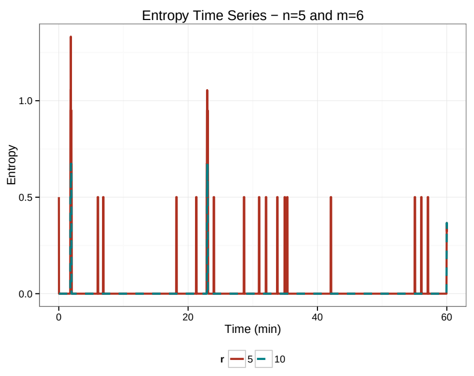
[12] A. Ng, “Machine learning: Anomaly detection,” Lecture Notes, Coursera Course, Standford University, September 2014. [Online]. Available: https://www.coursera.org/course/ml

[13] F. Salfner, M. Lenk, and M. Malek, “A survey of online failure prediction methods,” ACM Computing Surveys, vol. 42, no. 3, Mar. 2010, pp. 1–42. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1670679.1670680

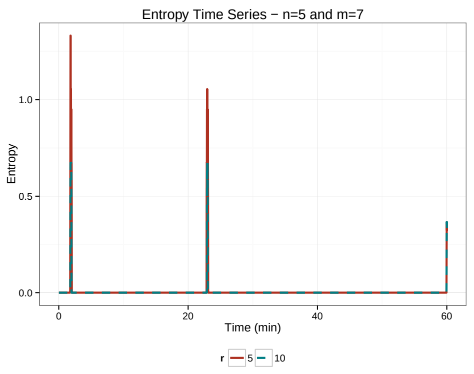
[14] Hping, “Hping 3,” September 2014. [Online]. Available: http://www.hping.org/hping3.html

[15] D. L. Quoc, L. Yazdanov, and C. Fetzer, “Dolen: User-side multi-cloud application monitoring,” in Future Internet of Things and Cloud. IEEE, 2014.

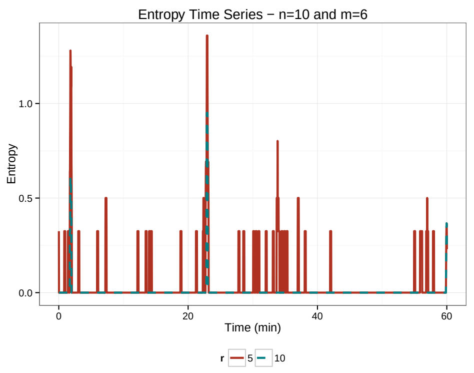
[16] D. L. Quoc, T. Jeong, H. E. Roman, and J. W.-K. Hong, “Traffic dispersion graph based anomaly detection,” in Proceedings of the Second Symposium on Information and Communication Technology, ser. SoICT ’11. New York, NY, USA: ACM, 2011, pp. 36–41. [Online]. Available: http://doi.acm.org/10.1145/2069216.2069227



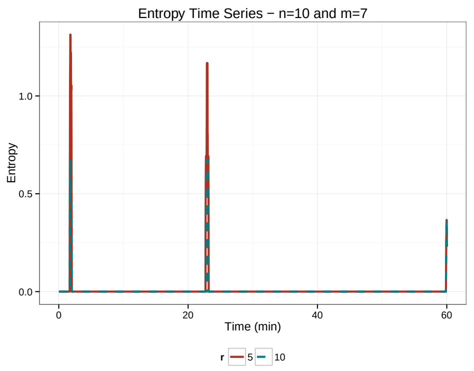
(a) $n = 5$; $m = 6$



(b) $n = 5$; $m = 7$



(c) $n = 10$; $m = 6$



(d) $n = 10$; $m = 7$

Figure 6. Entropy time series with: $n = 5$ (a, b); $n = 10$ (c, d).