2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11

Forecasting the Distribution of Network Traffic for Anomaly Detection

Christian Callegari, Stefano Giordano, Michele Pagano, and Teresa Pepe
Dept. of Information Engineering, University of Pisa, ITALY
E-mail: {firstname.lastname}@iet.unipi.it

***Abstract*—The increasing number of network attacks causes growing problems for network operators and users. Thus, detecting anomalous traffic is of primary interest in IP networks management.**

In this paper we present a novel method for network anomaly detection, based on the idea of discovering Heavy Change (HC) in the distribution of the Heavy Hitters in the network traffic. To assess the validity of the proposed method, we have performed an extensive experimental evaluation phase, during which our system performance have been compared to a more "classical" HC-based approach. The performance analysis, presented in this paper, demonstrates the effectiveness of the proposed method.

***Index Terms*—Anomaly Detection, Sketch, Heavy Hitter, Heavy Change**

I. Introduction

Uncovering anomalies in large ISPs and enterprise networks is challenging because of the wide variety of such anomalies. They can come from activity with malicious intentions (e.g., scanning, DDoS, prefix hijacking), or from misconfigurations and failures of network components (e.g., link failures, routing problems, outages in measurement equipment).

In the literature, the problem of detecting anomalies in the network traffic has often been seen as equivalent to the problem of detecting heavy changes (HCs) in some traffic descriptors. In this context a wide variety of approaches has been proposed.

Nevertheless most of them analyzes the single traffic flows, resulting to be unscalable and thus not applicable in modern backbone networks.

For this reason, in this work we have decided to analyze traffic aggregates, so as to obtain a more scalable system, and in more detail we have designed our system to work on the top of probabilistic structures, namely the sketches, that allow us to obtain a scalable real-time system (that analyzes the traffic flows after having randomly aggregated them), while simultaneously improving the detection rate of "classical" systems [4].

Given this *substrate*, our method is based on a statistical analysis of the distribution of the Heavy Hitters (HHs) in the network traffic. The idea behind this approach is that the distribution of the *big* flows should change between normal and attacks period, especially in the case of DoS/DDoS attacks, bot-nets, network scans, and so on.

Hence, in this paper we present a novel method for network anomaly detection, based on the idea of discovering HC in the distribution of the HHs in the network traffic. In more detail, we have explored the combined use of several forecasting algorithms and HC-based methods, applied to the HH evolution in time. To assess the validity of the proposed method, we have performed an extensive experimental evaluation phase, during which our system performance have been compared to a more "classical" HC-based approach.

The remainder of this paper is organized as follows: Section II presents some relevant related works. Then Section III describes the theoretical background useful for fully understanding the proposed technique, while Section IV provides a detailed description of the implemented system. Then Section V presents the experimental results and finally Section VI concludes the paper with some final remarks.

II. Related Works

In the area of data mining, there has been an extensive research of algorithms for HH detection.

The general problem of finding HHs in data streams has been studied extensively in [5], [15], [21], and [7]. The algorithms presented in these papers maintain summary structures that allow element frequencies to be estimated, within a prespecified error bound; the algorithms differ in whether they provide deterministic or probabilistic guarantees on the error bound, and whether they operate over insert-only streams or streams where elements can be inserted and also deleted.

The connection of these algorithms to the computer networks field is first made in [12], [22], [10].

In [12], Estan et al. initiate a new direction in traffic measurement by recommending concentrating on large flows only, i.e., flows whose volume is above a certain threshold. The authors also propose two algorithms for detecting large flows: Sample and Hold algorithm and Multistage Filters algorithm. Theoretical results show that the errors of these two algorithms are inversely proportional to the memory available. Furthermore, in [12], the authors note that network measurement problems bear a lot of similarities to measurement problems in other research areas such as data mining, and thus initiate the application of data streaming techniques to network anomaly detection.

In [22], the authors propose the Sticky Sampling algorithm and the Lossy Counting algorithm to compute approximate frequency counts of elements in a data stream. The proposed algorithms give guaranteed error bounds and require small memory. Thus, these algorithms also provide powerful tools for solving the HH detection problem in high-speed networks.

978-0-7695-4600-1/11 $26.00 © 2011 IEEE
DOI 10.1109/TrustCom.2011.25

In [10], Cormode et al. introduce the Count-Min Sketch method to HH and hierarchical HH detection. Sketches are probabilistic summary data structures based on random projections (the sketches will be detailed in next Section). The authors note that it is an open problem to develop extremely simple and practical sketches for data streaming applications.

In all these works, the application of such techniques to network anomaly detection is not explicitly discussed, only in [28] the authors propose a method based on the detection of the anomalies in the time series that contribute to the HHH. The main drawback of this work is the heavy computational complexity, due to the need of: at first estimating the HHH, then reconstructing the time series that contribute to the HHH (that is not always possible) and after that performing the real "detection" of anomalies.

On the other hand, in the context of anomaly detection, HC detection has been extensively studied as an important *component* of statistics based IDS.

Thus, several works have faced the problem, by proposing, among the others, techniques based on the use of "general" algorithms for detecting changes in data streams [16] and [11], neural networks [13], Markov models [27], and clustering algorithms [26]. Unfortunately most of the existing HC detection techniques can typically handle a relatively small number of time series. Given today's traffic volume, directly applying existing techniques on a per-flow basis cannot scale up to the needs of such massive data streams.

In this context, sketches have shown great potential. In [24] the authors first apply sketch to the HC detection problem. The input data are summarized using k-ary sketches and then different time series forecast models are implemented on top of the aggregate. The forecast errors are used to identify whether there are significant changes in the stream.

Finally, in [8] the authors introduce the concept of deltoid for HC detection, where a deltoid is defined as an item that has a large variability. The authors propose a framework based on a structure of Combinational Group Testing to find significant deltoids in high speed networks.

The method proposed in this paper extends all these previous works, by first proposing the idea of detecting HC in the distribution of the HH in the network traffic, without any need of reconstructing the original time series.

III. Theoretical Background

A. Data streaming model

In the last years, several data models have been proposed in the literature. In this paper, we describe the streaming data, by using the most general model: the Turnstile Model [23].

According to this model, the input data are viewed as a stream that arrives sequentially, item by item. Let $I = \sigma_1, \sigma_2, \ldots, \sigma_n$ be the input stream.

Each item $\sigma_t = (i_t, c_t)$ consists of a *key*, $i_t \in (1, \ldots, N)$, and a *weight*, c_t. The arrival of a new data item causes the update of an underlying function $U[i_t] \mathrel{+}= c_t$, which represents the sum of the *weights* of a given *key* over the time.

Given the underlying function $U[i_t]$ for all the keys of the stream, we can define the total sum S_t, at step t, as follows:

$$S_t = \sum_{i_t} U[i_t] \tag{1}$$

This model is very general and can be used in quite different scenarios. As an example, in the context of network anomaly detection, the key can be defined using one or more fields of the packet header (IP addresses, L4 ports), or entities (like network prefixes or AS number) to achieve higher level of aggregation, while the underlying function can be the total number of bytes or packets in a flow.

B. Sketch

Sketches are powerful data structures that can be efficiently used for keeping an accurate estimate of the function U.

In general, sketches are a family of data structures that use the same underlying hashing scheme for summarizing data. They differ in how they update hash buckets and use hashed data to derive estimates. Among the different sketches, the one with the best time and space bounds is the so called count-min sketch [9], which is basically the one used in this work.

In more detail, the sketch data structure is a two-dimensional $D \times w$ array $T[l][j]$, where each row l ($l = 1, \ldots, D$) is associated to a given hash function h_l. These functions give an output in the interval $(1, \ldots, w)$ and these outputs are associated to the columns of the array. As an example, the element $T[l][j]$ is associated to the output value j of the hash function l.

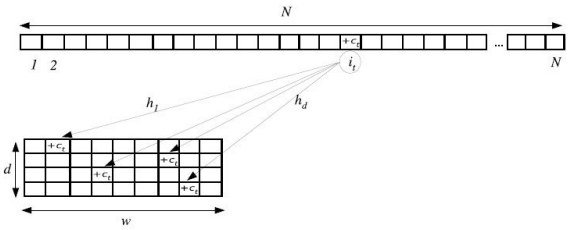

Fig. 1. Sketch: Update Function

Let $I = \{(i_t, c_t)\}$ be an input stream observed during a given time interval. When a new item arrives, the sketch is updated as follows:

$$T[l][h_l(i_t)] \leftarrow T[l][h_l(i_t)] + c_t \tag{2}$$

The update procedure is realized for all the different hash functions as shown in figure 1.

In this work, the sketches have been used for two distinct reasons, which will be clearer in the following: on one hand they allow the storing of big quantities of data (in our case we have to store the traffic generated by more than 220000 IP addresses) with big memory savings, on the other hand they permit a random aggregation of the traffic flows.

To be noted that, given the use of hash functions, it is possible to have some *collisions* in the sketch table. In more

detail, this last fact implies that each traffic flow will be part of several random aggregates, each of which will be analyzed to check if it presents any anomaly. This means that, in practice, any flow will be checked more than once (within different aggregates), thus, it will be easier to detect an anomalous flow. Indeed an anomalous flow could be masked in a given traffic aggregate, while being detectable in another one.

C. Heavy Hitter

A HH, in a data set, is an element whose relative frequency exceeds a specified threshold.

In more detail, given an input stream $I = \{(i_t, c_t)\}$ with the associated total sum S, a HH is a key, whose associated underlaying function $U[i]$ is no smaller than a specified proportion of the expected size of the whole data set.

The problem can be formalized as follows. Given a threshold ξ $(0 < \xi < 1)$ the set of HH is defined as:

$$HH = \{i \mid U[i] > \xi \cdot S\} \tag{3}$$

In the context of network anomaly detection, a HH is an entity which accounts for at least a specified proportion of the total activity measured in terms of number of packets, bytes, connections, etc. A HH could correspond to an individual flow or connection. It could also be an aggregation of multiple flows/connections that share some common property, but which themselves may not be HH.

Given this, we define the HH detection problem as the problem of finding all the HHs, and their associated values, in a data stream.

As an example, let us consider that the destination IP address is the *key*, and the byte count the $weight$; then the corresponding HH detection problem is to find all the destination IP addresses that account for at least a proportion ξ of the total traffic.

D. Heavy Change

A Heavy Change (HC) is a key i, whose associated underlaying function $U[i]$, evaluated in a given time bin, significantly differs in size from the same function evaluated in the previous time bins.

For sake of simplicity, let us suppose that we want to detect the HC related to two adjacent time bins. In this case, a key is a HC if the difference between the values of $U[i]$ in the two time bins exceeds a given threshold ψ.

The problem can be formalized as follows. Let $U^1[i]$ and $U^2[i]$ be the values associated to the key i, evaluated in the time bin 1 and 2 respectively, and let D_i be the difference, defined as $D_i = |U^1[i] - U^2[i]|$. Then the set of HC is defined as follows:

$$HC = \{i \mid D_i > \psi\} \tag{4}$$

As an example, in the context of network anomaly detection, the goal of HC detection can be to identify the flows that have significant changes in traffic volume from one time period to another.

E. Forecasting algorithms

A forecasting algorithm is an algorithm able to forecast the next value of a given time series starting from the past samples. In our work these algorithms will be used to forecast the next value of the sketch table.

In more detail, we have inspected two commonly used forecasting algorithms:

- Exponentially Weighted Moving Average (EWMA)
- Non-Seasonal Holt-Winters (NSHW)

Note that we have not analyzed algorithms belonging to the class of linear time series forecasting techniques since they model the time series behavior on the basis of several past samples requiring the storing of a huge quantities of data, hence not resulting suitable for the on-line detection of network anomalies.

1) Exponentially Weighted Moving Average: the EWMA algorithm [19] forecasts the object at time bin i starting from the forecasted sample and the observed sample, both in time bin $i-1$.

In more detail, let Z^i be the forecast for time bin i, Z^{i-1} the previous forecast, and Y^{i-1} the observed value at time bin $i-1$, then the forecast model can be described as follows:

$$Z^i = \begin{cases} \alpha Y^{i-1} + (1-\alpha) Z^{i-1} & \text{if } i > 2 \\ Y^1 & \text{if } i = 2 \end{cases} \tag{5}$$

The parameter α is called *smooting constant* and can assume values in the range $[0, 1]$. It reflects the weight given to the current observation (Y_{i-1}) in calculating the forecasted values Z_i. In more detail, the value of α determines the degree of smoothing and how responsive the model is to fluctuation in the time-series data. The value for α is arbitrary and is determined both by the nature of the data and the feeling by the forecaster as to what constitutes a *good response* rate. In practice, a smoothing constant close to zero leads to a stable model while a constant close to one is highly reactive.

2) Non-Seasonal Holt-Winters: simple exponential smoothing usually works best for series exhibiting no marked seasonality or trend. When a series does have a strong trend or cyclical component, we can use a more complex smoothing model. The NSHW algorithm [3] is a double exponential smoothing algorithm and, in this case, two components must be updated at each step: a smoothing component, Z_s, and a trend component, Z_t. The forecast is then $Z^i = Z_s^i + Z_t^i$.

The equations for the computation of the two components are reported in the following:

$$Z_s^i = \begin{cases} \alpha Y^{i-1} + (1-\alpha) Z^{i-1} & \text{if } i > 2 \\ Y^1 & \text{if } i = 2 \end{cases} \tag{6}$$

$$Z_t^i = \begin{cases} \beta (Z_s^i - Z_s^{i-1}) + (1-\beta) Z_t^{i-1} & \text{if } i > 2 \\ Y^2 - Y^1 & \text{if } i = 2 \end{cases} \tag{7}$$

where $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are two tunable smoothing parameters.

IV. System architecture

In this section we detail the system we have implemented to detect anomalies in the network traffic.

Figure 2 presents the general architecture of the system The proposed approach is called Method 2 in the figure, while Method 1 refers to a "classical" HC-based method [24] that is used as a benchmark for results comparison. The following subsections describe the significant blocks of the proposed system.

A. System Input

First of all the input data are processed by the data formatting module. Indeed, this module is responsible of reading the Netflow [6] traces and of transforming them in ASCII data files, by means of the Flow-Tools [1]. The output of this first block is given by text files containing on each line an IP address and the number of bytes received by that IP in the last time bin.

In more detail, in our implementation we have in input Netflow data, measuring the traffic gone through a given router over five minutes time-bins. Thus, this module will output a distinct file for each considered time bin; let us denote by N the distinct time bins.

Note that the modularity of the system allows great flexibility. Indeed, instead of considering the number of bytes sent by a given IP, the system administrator can easily choose of using another traffic descriptor that better allows her to detect the different attacks.

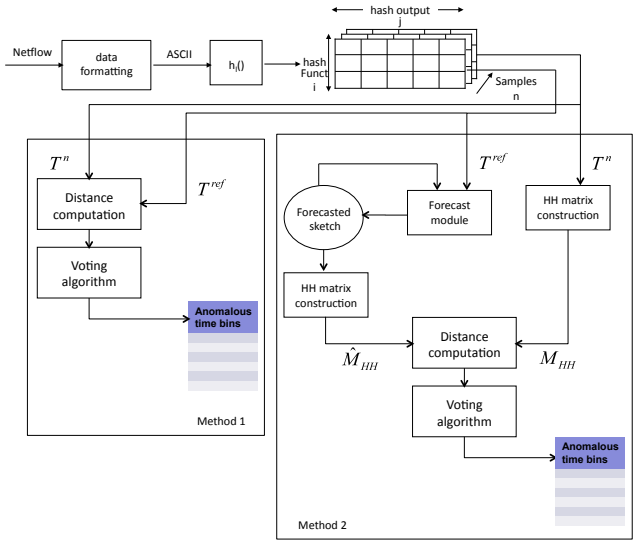

Fig. 2. System Architecture

B. Sketch module

After the data have been correctly formatted, they are passed as inputs to the hash functions responsible for the construction of the sketch tables. In more detail, for each line of each file, the IP address is considered as the key i_t, while the number of bytes is considered as the weight c_t. Each file, corresponding to a time bin, is thus used to build a distinct sketch table.

Note that in our implementation we have used $D = 16$ distinct hash functions, which give output in the interval $(0, 1, \ldots, w-1)$, that means that the resulting sketch tables will be $\in \mathbb{N}_{D\times w}$, where w can be varied. As far as the the hash functions are concerned, we have chosen to use functions belonging to the 4-universal hash family[1] [25], obtained as:

$$h(x) = \sum_{i=0}^{3} a_i \cdot x^i \ mod\ p\ mod\ w \tag{8}$$

where the coefficients a_i are randomly chosen in the set $(0, 1, \ldots, p-1)$ and p is a random prime number (we have considered the Mersenne numbers).

At this point, given that we had N distinct time bins, we have obtained N distinct sketch tables $T^n_{D\times w}$, where $n \in (1, 2, \ldots, N)$ is the time bin.

C. Detection Phase

1) Method 1: Method 1 represents a classical HC-based method, used in this work as a benchmark for evaluating the system performance. Hence, we have chosen a slightly improved version of a simple, yet effective, model known in the literature [24].

In more detail the system compares the data related to two adjacent time bins, that is two consecutive sketch tables.

In practice, the system computes the euclidean distance d_{ij} between each element $T[i][j]$ of the current sketch table and the corresponding element $T^{ref}[i][j]$, where $T^{ref}[i][j]$ is the element $T[i][j]$ corresponding to the last non anomalous time bin.

In more detail, assuming that the system is processing the time bin n, then each element of the sketch table T^n is compared with the corresponding element of the "reference" sketch table T^{ref}, where T^{ref} is equal to the "last occurred" non anomalous sketch table, i.e., $T^{ref} = T^{n-r}$ for some $(r = 1, 2, \ldots, n)$. Note that this algorithm has been introduced to avoid the "masking effect" that can be caused by anomalies that span over multiple time bins.

If the computed distances exceed a given threshold ($d^n_{ij} > \xi$) in at least H distinct rows of the sketch table (where H is a tunable parameter), the system considers the current time bin as anomalous and then performs the anomaly identification for revealing the responsible IP flows.

2) Method 2: This methods, that represents the novel contribution of this work, is much more complex than the first one, but it still results to be (as demonstrated in the experimental section) suitable for on-line detection of anomalous flows. Basically, it tracks the variations in the HH distribution of the network traffic.

[1]A class of hash functions $H : (1, \ldots, N) \rightarrow (1, \ldots, w)$ is a *k-universal hash* if for any distinct $x_0, \cdots x_{k-1} \in (1, \ldots, N)$ and any possible $v_0, \cdots v_{k-1} \in (1, \ldots, w)$:

$$Pr_{h\in H} = \{h(x_i) = v_i; \forall i \in (1, \ldots, k)\} = \frac{1}{w^k}$$

The forecast module takes in input its own output at the previous step and the "reference" sketch table T^{ref}, that is -as in the previous case- the "last occurred" non anomalous sketch table, and uses these two elements for forecasting the value of the next sketch table.

Note that the use of T^{ref} has been introduced, as in method 1, to avoid the "masking effect" due to "long" anomalies.

The prediction phase is performed by using either the EWMA algorithm,

$$\widehat{T}^n = \alpha T^{ref} + (1-\alpha)\widehat{T}^{n-1}$$

or the NSHW algorithm,

$$\widehat{T}^n = \widehat{T}^n_s + \widehat{T}^n_t$$

with

$$\widehat{T}^n_s = \alpha T^{ref} + (1-\alpha)\widehat{T}^{n-1}$$

$$\widehat{T}^n_t = \beta(\widehat{T}^n_s - \widehat{T}^{n-1}_s) + (1-\beta)\widehat{T}^{n-1}_t$$

Given this step, regardless of the used algorithm, the system has two distinct values for the sketch table at time bin n, the real value T^n and the predicted value $\widehat{T}^n$. Both these tables are fed to a module, responsible for computing an *empirical* distribution of the HHs.

This "distribution" is computed by evaluating the HHs present in the traffic, that is the traffic aggregates (namely the sketch buckets) that exceed a given threshold, given by a percentage of the total traffic in the time bin, S^n. The related buckets are then updated by inserting the quantity of traffic for which that aggregate exceeds the threshold, while all the other buckets are set to one byte. Finally each row of the matrix is normalized so as that its elements sum to one.

This matrix is named M^n_{HH} if computed starting from T^n and $\widehat{M}^n_{HH}$ if calculated starting from $\widehat{T}^n$.

Given these two matrices, the system compares the actual HH *distribution* in M_{HH} with the forecasted one in $\widehat{M}_{HH}$. To perform such task the system computes the Jensen-Shannon Divergence (JSD) [14] between each line of the two matrices. Note that JSD has been introduced in this system to overcome the potential limitations given by the asymmetric nature of more "classical" statistical distance, as like as the Kullback Leibler divergence.

To decide if the considered time bin is anomalous, we have implemented a voting algorithm, that is if the computed distance exceeds a given threshold ψ for more than H rows of the matrix, the system reveals an anomalous time bin and the anomaly is thus identified.

V. Experimental results

As it is known a serious issue in testing IDSs is represented by the lack of complete datasets provided with a ground truth. Indeed, the only one (DARPA IDEVAL) dates back to 1999 [18] and it is not representative of real traffic [20]. Thus, it is a common choice to use a real data-set and to synthetically add some anomalies [17].

The proposed system has been tested using a publicly available data-set, composed of traffic traces collected in the Abilene/Internet2 Network [2], a hybrid optical and packet network used by the U.S. research and education community.

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	1969	154
ξ_2	1920	48
ξ_3	1381	28
ξ_4	1269	23

TABLE I
Experimental Results: HC-based method

Threshold	Total Anomalies	Synthetic Anomalies
ξ_1	2015	154
ξ_2	2000	153
ξ_3	1976	138
ξ_4	1946	91
ξ_5	1900	70
ξ_6	1687	46
ξ_7	1660	22
ξ_8	1657	17
ξ_9	1652	15
ξ_{10}	1650	15

TABLE II
HC-based method with forecasting ($w = 512$, EWMA $\alpha = 0.2$, JSD)

The used traces consist of the traffic related to nine distinct routers, collected in one week, and are organized into 2016 files, each one containing data about five minutes of traffic (netflow data). To be noted that the last 11 bits of the IP addresses are anonymized for privacy reasons; nevertheless we have more than 220000 distinct IP addresses.

Since the data provided by the Internet2 project do not have a ground truth file, we are not capable of saying *a priori* if any anomaly is present in the data. Because of this reason we have performed a manual verification of the data (according to the method presented in [17]), analyzing the traces for which our system reveals the biggest anomalies. Moreover we have synthetically added some anomalies in the data, so as to be able to correctly interpret the offered results.

In more detail, we have added anomalies that can be associated to DoS and DDoS attacks, represented by four or five distinct traffic flows, each one carrying a traffic of $5 \cdot 10^8$ bytes (154 anomalies in total), either spanning a single or multiple time bins. It is worth noticing that in all the cases the original traces already presented traffic flows of that order.

As already stated in the previous section, we have considered as input to the system the number of bytes received by a given IP address. This choice is supported by the obtained experimental results. Nevertheless, it is possible to feed the system with another metric, just simply modifying the first block, if the "new" metric can result more suitable for detecting some attacks.

Since, given the nature of the data set, we cannot plot a ROC curve, in the presented tables we report the total number of detected anomalies and the number of synthetic anomalies detected by the system. Note that the tables have been obtained varying the values of the thresholds, ξ and ψ for the two algorithms. The real values of such thresholds are not reported since are not significant in themselves, just consider that the

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	310	154
ψ_2	256	152
ψ_3	199	148
ψ_4	179	144
ψ_5	172	142
ψ_6	167	137
ψ_7	163	133
ψ_8	151	123
ψ_9	135	111
ψ_{10}	115	94

TABLE III
OUR METHOD (EWMA $\alpha = 0.2$)

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	473	154
ψ_2	333	153
ψ_3	246	150
ψ_4	208	146
ψ_5	190	144
ψ_6	172	137
ψ_7	167	134
ψ_8	150	121
ψ_9	1697	117
ψ_{10}	125	100

TABLE IV
OUR METHOD (EWMA $\alpha = 0.5$)

first values (namely ξ_1 or ψ_1) always correspond to the highest threshold value for which the system is able to detect all the 154 synthetic anomalies. Hence, in general, the system is always able to obtain a 100% detection rate (revealing all the 154 synthetic anomalies), but the performance can be strongly different depending on the total number of detected anomalies that has a direct impact on the number of false alarms.

For the sake of brevity we do not present the results related to the study on the impact of the sketch dimension w on the system performance; all the presented results correspond to a sketch table of dimension $w = 512$.

To start with, let us analyze the performance offered by the "classical" HC-based system, reported in Table I. In this case, we can easily see that for detecting all the synthetic anomalies, we have to accept a total number of detection equal to 1969, which is not acceptable. Moreover the number of detected synthetic anomalies suddenly decreases when increasing the threshold, while the number of total detected anomalies remains quite stable, making very hard the application of the system in the "real world".

The same considerations are valid for the results presented in Table II, which shows the performance offered by the "classical" HC-based system, applied together with a forecasting algorithm (EWMA, $\alpha = 0.2$) and JSD.

Concerning the method presented in this paper, two distinct sets of experimental tests have been conducted to tune the parameter of the forecasting algorithms, EWMA and NSHW respectively.

In more detail, Tables III, IV, and V present the results achieved by the system using three distinct values of the smoothing parameter of the EWMA algorithm, namely $\alpha = 0.2$, $\alpha = 0.5$, and $\alpha = 0.8$.

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	688	154
ψ_2	344	152
ψ_3	284	149
ψ_4	225	146
ψ_5	1320	142
ψ_6	276	136
ψ_7	157	122
ψ_8	1697	119
ψ_9	1767	109
ψ_{10}	126	85

TABLE V
OUR METHOD (EWMA $\alpha = 0.8$)

From the tables we can notice that when increasing the value of the smoothing parameter, we have an increase in the number of total detection. Indeed for detecting all the synthetic anomalies the system detects a total number of anomalies equal to 310 (case $\alpha = 0.2$), 473 (case $\alpha = 0.5$), or 688 (case $\alpha = 0.8$).

In this case, to really evaluate the performance of the system, we have performed a manual verification of the data set, checking the additional detections of the system. From that, we can conclude that, the cases $\alpha = 0.5$ and $\alpha = 0.8$ take to a significant number of false alarms, while almost all the additional detections obtained with $\alpha = 0.2$ (310 total detections minus the 154 synthetic anomalies) are real anomalies already present in the traces.

Moreover, note that, in any case, event though all of the additional detections obtained with $\alpha = 0.2$ would not be "real" anomalies they would correspond to a maximum false alarm rate of 8.3% that could be considered as "acceptable".

We can also easily notice, by analyzing Table III, that the number of detected synthetic anomalies varies quite slowly when increasing the value of the threshold, while the number of total detection decreases much faster that make easy the tuning of the system.

Finally, additional tests (not shown for the sake of brevity) have demonstrated that varying the smoothing parameter around the value 0.2 (i.e., $\alpha \in [0.1, 0.3]$, does not take to any significant variation in the system performance.

From these considerations we can conclude that the best performance are obtained when $\alpha = 0.2$. Note that this is also supported by the literature, indeed it is known that 0.2 is in the typical range for the smoothing parameter.

Moreover, using a low value for the smoothing parameter implies the use of a model not much responsive to the fluctuations in the data. This has a direct impact on our system performance. Indeed, by analyzing Tables IV and V we can notice that the system present a "strange" behavior. Indeed the total number of detections is not always decreasing, when increasing the value of the threshold. This is due to the presence of "noisy samples" in the data and it is hence mitigated when the parameter α tends to zero.

Analogously to what done for tuning the smoothing parameter of the EWMA algorithm, Tables VI - VIII present an analysis of the system performance obtained varying the value of the parameter β of the NSHW algorithm (for sake of brevity

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	402	154
ψ_2	299	152
ψ_3	237	146
ψ_4	206	144
ψ_5	187	139
ψ_6	166	133
ψ_7	159	128
ψ_8	137	111
ψ_9	122	98
ψ_{10}	107	87

TABLE VI
OUR METHOD (NSHW $\alpha = 0.2$ $\beta = 0.2$)

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	531	154
ψ_2	372	153
ψ_3	279	150
ψ_4	237	144
ψ_5	196	139
ψ_6	185	133
ψ_7	165	129
ψ_8	154	118
ψ_9	1695	113
ψ_{10}	122	94

TABLE VII
OUR METHOD (NSHW $\alpha = 0.2$ $\beta = 0.5$)

we do not show the results corresponding to different values of α, since they are similar to those obtained for EWMA). In more detail the presented results have been obtained by using $\alpha = 0.2$ and three distinct values of β, namely $\beta = 0.2$ (Table VI), $\beta = 0.5$ (Table VI), and $\beta = 0.8$ (Table VI).

For these tables, the considerations done for the previous set of tests are still valid and take us to conclude that the best value for the β parameter is $\beta = 0.2$.

Given these results, we can make a comparison between the use of the EWMA and the NSHW algorithms, by comparing Table III and VI that correspond to the best settings for the two considered cases. The inspection of the data set takes us to conclude that the best performance are achieved when using the EWMA algorithm.

Table IX present the results of the system when disabling the forecasting module ($\widehat{M}_{HH}$ is directly computed starting from T^{ref}). By comparing this table with the previous ones, we can see that disabling the forecasting module takes to worsen the performance. This result was predictable, indeed disabling the forecasting module is equivalent to use the EWMA algorithm

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	660	154
ψ_2	436	152
ψ_3	322	150
ψ_4	252	145
ψ_5	206	140
ψ_6	186	137
ψ_7	185	132
ψ_8	158	121
ψ_9	1696	115
ψ_{10}	146	99

TABLE VIII
OUR METHOD (NSHW $\alpha = 0.2$ $\beta = 0.8$)

Threshold	Total Anomalies	Synthetic Anomalies
ψ_1	968	154
ψ_2	538	153
ψ_3	329	150
ψ_4	276	147
ψ_5	1326	146
ψ_6	412	136
ψ_7	1319	127
ψ_8	1768	122
ψ_9	1767	111
ψ_{10}	659	89

TABLE IX
OUR METHOD (NO FORECASTING)

with $\alpha = 1$, and the previous analysis had already highlighted that the best performance are achieved with low values of the smoothing parameter.

After this analysis we can thus conclude that the presented system outperforms the "classical" HC-based methods and that the best settings are those corresponding to the results presented in Table III, that is EWMA algorithm with $\alpha = 0.2$.

Finally, to evaluate the computational complexity in time and memory space of the proposed system, we have used a general purpose PC, equipped with an Intel Core 2 Duo processor at 3GHz and 2GB of RAM. The experimental results have shown that the system (using the best configuration parameters) is able to process a whole week of traffic from the Abilene/Internet2 network in about 531s, with a maximum memory consumption of 0.9% (about 1.8 MB).

In more detail the system has demonstrated to be able to analyze a single time-bin of 5 minutes of traffic related to a single router in about 29ms, demonstrating to be suitable for the on-line detection of anomalies in backbone networks.

VI. CONCLUSIONS

In this paper we have presented a novel anomaly detection method, based on the analysis of the behavior of the HH in the network traffic. In more detail, our system is based on the monitoring of the HCs in the HH distribution, by means of a combined use of sketches, forecasting algorithms and statistical distances.

To assess the validity of the proposed solution, we have tested the system over a week of traffic collected in the Internet2/Abilene network. The performance analysis has been targeted at first to the tuning of the different parameters of the method and then to extensively verify the effectiveness of the system. Such analysis has highlighted that, for a proper choice of the parameters, the implemented system obtains very good results, detecting all the synthetic anomalies and some more anomalies already present in the original data-set.

In conclusion, the proposed system has proved to outperform the "classical" HC-based methods, demonstrating that improvements in the performance can be obtained by performing the change detection on the set of the HHs, instead of using the whole traffic, and by applying a forecasting module. Moreover the system has resulted to be suitable for the on-line detection of anomalies in backbone networks.

ACKNOWLEDGMENT

This work was partially supported by DEMONS, a research project supported by the European Commission under its 7th Framework Program (contract-no. 257315). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DEMONS project or the European Commission.

REFERENCES

[1] Flow-Tools Home Page. http://www.ietf.org/rfc/rfc3954.txt.

[2] The Internet2 Network. http://www.internet2.edu/network/.

[3] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day Incorporated, 1990.

[4] C. Callegari, L. Gazzarrini, S. Giordano, M. Pagano, and T. Pepe. When randomness improves the anomaly detection performance. In *Proceedings of the International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, 2010.

[5] M. Charikar, K. Chen, and M. Farach-colton. Finding frequent items in data streams. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 693–703, 2002.

[6] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), Oct. 2004.

[7] G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proceedings of the ACM Principles of Database Systems*, pages 296–306, 2003.

[8] G. Cormode and S. Muthukrishnan. What's new: Finding significant differences in network data streams. In *Proceedings of the IEEE Infocom*, pages 1534–1545, 2004.

[9] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58 – 75, 2005.

[10] G. Cormode, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in data streams. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 464–475, 2003.

[11] T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, and K. Yi. Change (detection) you can believe in: Finding distributional shifts in data streams. In *IDA*, pages 21–34, 2009.

[12] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 21:270–313, 2003.

[13] K. L. Fox, R. R. Henning, J. H. Reed, and R. P. Simonian. A neural network approach towards intrusion detection. In *Proceedings of the National Computer Security Conference. Information Systems Security. Standards - the Key to the Future, Vol. I*, pages 124–134, 1990.

[14] B. Fuglede and F. Topsøe. Jensen-Shannon Divergence and Hilbert space Embedding. In *Proceedings of the International Symposium on Information Theory*, 2004.

[15] R. M. Karp, C. H. Papadimitriou, and S. Shenker. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems*, 28:2003, 2003.

[16] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams, 2004.

[17] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *SIGCOMM Comput. Commun. Rev.*, 35(4):217–228, 2005.

[18] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

[19] J. M. Lucas and M. S. Saccucci. Exponentially weighted moving average control schemes: Properties and enhancements. *Technometrics*, 32:1–12, 1990.

[20] M. V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 220–237. Springer-Verlag, 2003.

[21] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 346–357, 2002.

[22] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 346–357, 2002.

[23] S. Muthukrishnan. Data streams: algorithms and applications. In *Proceedings of the annual ACM-SIAM symposium on Discrete algorithms*, pages 413–413, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[24] B. K. Subhabrata, E. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 234–247, 2003.

[25] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 615–624, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[26] J. Tolle and O. Niggemann. Supporting intrusion detection by graph clustering and graph drawing. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*. Springer-Verlag, 2000.

[27] N. Ye. A Markov chain model of temporal behavior for anomaly detection. In *Proceedings of the Workshop on Information Assurance and Security*, 2000.

[28] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online identification of hierarchical heavy hitters: Algorithms, evaluation, and applications. In *In Proceedings of the 4th ACM SIGCOMM Internet Measurement Conference*, pages 101–114. ACM Press, 2004.