# Online Detection of Network Traffic Anomalies Using Behavioral Distance

Hemant Sengar Xinyuan Wang† Haining Wang ‡ Duminda Wijesekera† Sushil Jajodia†

Technology Development Dept.
NuVox Communications
Greenville, SC 29601
*hsengar@nuvox.com*

†Center for Secure Information Systems
George Mason University
Fairfax, VA 22030
{*xwangc,dwijesek,jajodia*}*@gmu.edu*

‡Department of Computer Science
College of William and Mary
Williamsburg, VA 23187
*hnw@cs.wm.edu*

***Abstract*— While network-wide anomaly analysis has been well studied, the on-line detection of network traffic anomalies at a vantage point inside the Internet still poses quite a challenge to network administrators. In this paper, we develop a behavioral distance based anomaly detection mechanism with the capability of performing on-line traffic analysis. To construct accurate on-line traffic profiles, we introduce *horizontal* and *vertical* distance metrics between various traffic features (i.e., packet header fields) in the traffic data streams. The significant advantages of the proposed approach lie in four aspects: (1) it is efficient and simple enough to process on-line traffic data; (2) it facilitates protocol behavioral analysis without maintaining per-flow state; (3) it is scalable to high speed traffic links because of the aggregation, and (4) using various combinations of packet features and measuring distances between them, it is capable for accurate on-line anomaly detection. We validate the efficacy of our proposed detection system by using network traffic traces collected at Abilene and MAWI high-speed links.**

## I. Introduction

In an effort to run networks smoothly and efficiently, the network operators are regularly confronted with traffic anomalies. Some of the anomalies are innocuous in nature such as network equipment outages or flash crowd events, whereas others are malicious in nature (e.g., DoS attacks, worms, port scans etc.). Nevertheless, both types of anomalies need to be accurately detected, accurately and preferably in real-time. In the past few years, we have witnessed a rich research in network-wide anomaly detection [12], [15], [20], [28] and a shift from simple volume-based analysis to traffic feature (i.e. packet headers) distribution analysis [13], [25], [29]. The traffic feature distributions provide an approach to conducting fine-grained network traffic analysis [12]. In these works, traffic matrix is used to describe statistical behaviors, *entropy* is used to reveal the changes in distributional aspect of the traffic features [12], [14], and PCA (principal Component Analysis) is used to perform network-wide anomaly analysis on input links and random aggregation. However, most of these network-wide anomaly detection and *machine-learning* approaches are performed offline. Data analysis and mining are performed after the fact and on the repository data. On-line anomaly detection at the backbone links or high-speed network vantage points is a challenge, mainly due to the link speed and diversity of the Internet traffic.

Detections of network traffic anomalies at high-speed links require traffic aggregation and profiling. The aggregation makes it feasible to compute traffic statistics in high-speed links. However, it is difficult to construct an accurate profile of normal traffic behaviors. The use of aggregation also suffers from *behavioral aliasing* and *spoofing* problems, as pointed out by Kompella et al. [10]. The *behavioral aliasing* problem occurs when an aggregation of bad behavior looks like a good behavior. The *spoofing* problem is caused by attackers flooding traffic with bogus spoofed IP addresses and port numbers. These spoofed packets could make an aggregated traffic behavior seemingly benign.

In this paper, we develop a behavioral distance based anomaly detection mechanism, with the capability of performing online traffic analysis, to meet the challenges mentioned above. For the construction of accurate online traffic profiles, we introduce horizontal and vertical distance metrics between various traffic features (i.e., packet header fields) in the traffic data streams. The horizontal distance measures the variability of a particular traffic feature distribution, one at a time. Meanwhile, we can group a number of traffic features together, and the vertical distance measures the variability in terms of traffic groups. The number of false positives and negatives due to behavioral aliasing can be significantly reduced by correlating horizontal and vertical distances measured from the various combinations of traffic features. The vertical distance metric also helps to mitigate the spoofing problem. The measurement of behavioral distance between traffic features is based on the *Hellinger distance*, an information theoretic approach which computes the variability between two probability measures independent of parameters. Moreover, to captures the essence of the dynamics of network traffic, a moving window mechanism is applied in the construction of normal traffic profile. We evaluate the effectiveness of the proposed approach for on-line network anomaly detection, based on real Internet traces collected at different high-speed backbone links including Abilene and MAWI.

The remainder of this paper is structured as follows. Section II surveys related work. Section III presents the relatively stable nature of network traffic attributes. Section IV briefly describes the basic approach of our detection scheme, including Hellinger distance. Section V details the anomaly detection

methodology, including the design of distance metrics. Section VI evaluates the effectiveness of the proposed scheme. Some issues with our approach are discussed in Section VII and finally, Section VIII concludes the paper.

## II. Related Work

High-speed network monitoring exposes the detection device to a large number of flows. Intrusion detection systems Snort [21] and Bro [24] maintain per flow state and match packets to a pre-defined set of rules, making them unscalable to high-speed links. To speed up rule matching, some vendors (such as NetScreen [17] and Fortinet [8]) have implemented detection rules in hardware. The rule matching approaches for anomaly detection are unable to detect unknown anomalies.

Statistical approaches can detect unseen anomalies in the traffic streams where most of the statistical anomaly detection mechanisms are based on the variations of *change detection* methods. In Holt Winter forecasting model [3], the history of the network traffic is used to predict the future traffic rate. An alarm is raised if the current traffic deviates too much from the future prediction. Similarly, Barford et al. [1]'s wavelet analysis and Wang et al. [26]'s non-parametric cumulative sum (CUSUM) method are used to study the variance of the network traffic. These detection mechanisms can typically handle a relatively small number of time series.

A variety of statistical anomaly detection techniques including PCA and entropy have demonstrated the accuracy and efficiency in detecting network-wide anomalies [12], [15], [20], [28]. These techniques detect anomalies in the traffic matrix time series. In other words, the time series of traffic between each pair of origin and destination routers in the Internet. The rationale behind using PCA for network-wide anomaly detection is that network traffic has low intrinsic dimensionality, and PCA works well by analyzing the origin-destination flow aggregation and entropy time series of traffic features. The entropy-based approaches rely on *entropy* or *relative entropy* to measure the degree of dispersal or concentration of a traffic feature distribution. Lakhina et al. [12] used entropy and subspace methods to mine traffic anomalies from network wide traffic data repositories. Gu et al. [9] used maximum and relative entropy to develop a behavior-based anomaly detection method. In Gu et al.'s approach, the maximum entropy-based baseline distribution is constructed from pre-labeled training data, but how this baseline is adapting itself to the dynamics of network traffic remains unclear.

Computing real-time statistics at a high-speed link suffers from the vast amount of data. Duffield et al. [6] showed that even packet sampling is not scalable, especially after aggregation. There can be up to $2^{64}$ flows defined by only considering source and destination IP addresses [22]. Recently, Cormode et al. [4]'s *deltoids* and Krishnamurthy et al. [11]'s *k-ary sketch* methods have been proposed for heavy change detection in high speed traffic. The deltoid expands k-ary sketch with multiple counters for each bucket in the hash tables. The reverse hashing method further improves sketch-based change detection, which is more efficient and can infer the keys of culprit flows [22].

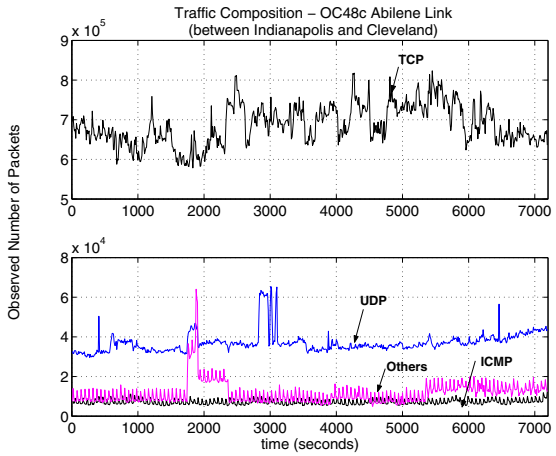



Fig. 1. Typical Traffic Composition at Various Links

The work done by Xu et al. [27] is closest to our work, in which traffic behavior profiling is conducted in real-time. However, compared to their approach of using *flows* (based on the 5-tuple of protocol, srcIP, destIP, srcPort, destPort) to do traffic analysis, we take a reverse approach. To avoid the memory intensive flow-based analysis, we consider these attributes as individual entities. The traffic data is partitioned into separate groups based on the selected attributes. Then, using statistical analysis, we endeavor to establish relationship among these groups.

## III. Nature of Traffic Features

We conduct extensive statistical analysis to profile the normal behaviors of backbone network traffic, based on the three traces collected from various Internet backbone (high-speed) links. The first trace is collected at Atlanta (inbound traffic) from an Abilene OC48c backbone link, which connects Atlanta to Indianapolis. The trace consists of 8 samples of 90 seconds each, taken on the same day of June 18, 2004 [19]. The second trace is also an OC48c Packet-over-SONET data set, collected at the Indianapolis CISCO GSR 12015 router node on May, 2002 [18]. This backbone trace is a two-hour contiguous outbound traffic collection (towards Cleveland). Finally, the third and fourth trace is from a public domain MAWI network traffic archive of the WIDE project [16]. It is contiguous 6 and 2 hours worth of traffic collected from the trans-Pacific link (between US and Japan) on September 22, 2005 and on January 10, 2007 respectively.

The observed nature of network traffic is dynamic and diverse, the volume and its composition varies with the change of time and day. The typical traffic composition consists of $\simeq 80-90\%$ of TCP packets, $\simeq 7-10\%$ of UDP packets and the remainder is of ICMP and *Others* (a heterogeneous group of protocols such as GRE, RSVP, OSPF, IGMP, PIM etc.) as shown in Figure 1 for (Indianapolis $\rightarrow$ Cleveland) link. From this diverse traffic mix, we selected some of the packet header fields to study its distributional aspect. The selected packet headers are defined as traffic features in this paper. Lakhina et al. [12] observed that under traffic anomalies, for example, worm propagation, DoS attack towards a particular victim, there will be detectable and disproportional changes

on the distributional aspect of these chosen traffic features. Our premise of network traffic anomalies detection is also based on the distributional aspect of traffic features and any changes thereof, reveals a considerable large set of anomalies. However, our problem domain is quite different. Instead of analyzing the repository data for discovering anomalies, our proposed scheme aims to detect any distributional changes in high-speed link's traffic features at real-time.

The online study of traffic feature's distribution at a high-speed link is a difficult task, mainly due to vast amount of data and limited computational resources. To overcome both of these problems, our approach introduces the following techniques: (1) The network traffic features are aggregated at a very coarse level. There is no per-flow state maintenance and we even do not attempt to relate source IP addresses with its corresponding destination IP addresses. (2) The variations of traffic feature distributions are measured using computationally simple and efficient Hellinger distance. (3) To reduce the memory requirement for traffic feature collection, we use hash tables that only consume a limited number of *keys*.

To reveal the inherent nature of traffic feature distributions, we parse all the traces and classified them into four broad categories ICMP, UDP, TCP, and "Others", depending upon protocol-type values. From the packet headers, we extracted IP addresses and port numbers over a ten-second time window. Figures 2 (a.), (b.), (c.) and (d.) plot the time-varying distributions for distinct IP addresses and port numbers observed in the backbone traffic streams. Figure 2 shows that compared to the unpredictability of volume metric such as the number of packets, byte counts etc., the distributions of distinct IP addresses and port numbers are relatively stable with time. For example, an OC48c (Indianapolis $\rightarrow$ Cleveland) link carrying $\simeq$ 0.8 million packets within a ten-second time window requiring only $\simeq 6 - 12K$ key entries in the hash table for a particular traffic feature's data collection. We also observe that compared to source IP addresses and destination port numbers, the destination IP addresses and source port numbers fluctuate, but still remains bounded. A plausible reason for this behavior could be explained as these links aggregate the traffic originated from hosts behind the NAT boxes. Further, in the case of MAWI trans-Pacific trace as shown in Figure 2 (c.), we observe some interesting large spikes in an otherwise stable distribution of traffic features. These spikes are caused by traffic anomalies, which are further discussed as a part of performance analysis in Section VI.

Overall, we observe that some traffic features (such as distinct source, destination IP addresses and distinct source, destination port numbers) are relatively stable with time. At any instant of time, these features hold an intrinsic correlation among themselves. Given any two different instants of time and the associated feature distributions, we can measure (quantify) the observable similarity of the features for these two instants. This similarity measure provides a unique way to compare an unknown observed traffic behavior with the known legitimate (i.e., normal) traffic behavior. Depending upon the value of the similarity measure, we can classify the observed traffic as either normal or an instance of an attack.

## IV. The Basic Approach

There are two working phases in our anomaly detection scheme: training period and testing period. During the training period, we assume that traffic is devoid of any attack and the characterization of traffic features acts as a normal profile. During the testing period, we detect traffic anomalies. Our scheme depends upon two key issues, first, capturing traffic features during the *training* and *testing* period and secondly, finding a way to measure the similarity (or dissimilarity) between these traffic features captured in the two different periods. The normal profile learned during the training period is used as a reference to measure the distance between itself and the current profile of traffic features in the testing period. The compositional (i.e., features) distance between the normal and current profiles can be measured using Steinhaus (e.g., Odum or Bray-Curtis distance) similarity measure or Hellinger distance. Our choice of Hellinger distance for an online statistical detection mechanism is due to its desirable properties: (1) it is based on the proportional abundance and (2) it has the range of similarity values between 0 and 1.

Hellinger distance measures the distances between probability measures independent of their distributions, and is closely related to the *total variation distance* but with several advantages. Let $\mathbb{P}$ and $\mathbb{Q}$ be the two probability measures on a finite event space $\Omega$. Probability measure $\mathbb{P}$ on $\Omega$ is an N-tuple $(p_1, p_2, ..., p_N)$, which satisfies $p_\alpha \geq 0$ and $\sum_\alpha p_\alpha = 1$. Similarly, $\mathbb{Q}$ on $\Omega$ is also an N-tuple $(q_1, q_2, ..., q_N)$, which satisfies $q_\alpha \geq 0$ and $\sum_\alpha q_\alpha = 1$. The *Hellinger distance* between $\mathbb{P}$ and $\mathbb{Q}$ is defined as :

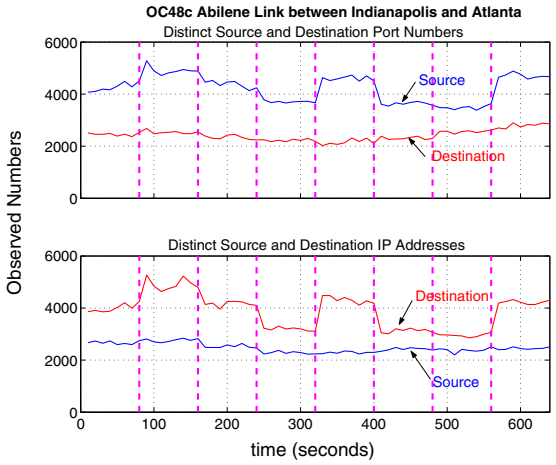
$$d_H^2(\mathbb{P}, \mathbb{Q}) = \frac{1}{2} \sum_{\alpha=1}^{N} (\sqrt{p_\alpha} - \sqrt{q_\alpha})^2$$

The Hellinger distance satisfies the inequality of $0 \leq d_H^2 \leq 1$. The distance is 0 when $\mathbb{P} = \mathbb{Q}$, and 1 when $\mathbb{P}$ and $\mathbb{Q}$ are disjoint.

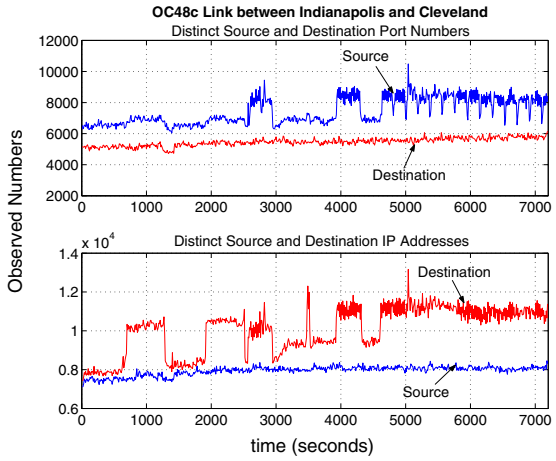
## V. Diagnosis Methodology

In this section, we demonstrate the application of Hellinger distance to detect abnormal behaviors of network traffic based on packet features distributions. There are basically four steps involved from the observation of traffic stream to raising an alarm. The first is to gather information from streaming traffic. At a particular link, the *traffic monitoring* module observes packet's headers and collects statistics. The second step is to quantify any variations observed in the distribution of packet features. This is the *distance measurement* process, where the profiles of packet features are compared and their similarity (or dissimilarity) is given a numerical value between 0 and 1. The third step is to set a threshold of measured distances that can clearly differentiate network anomalies from normal behaviors. Finally, the fourth step is to gather all the information and classify the attack alerts with the possibility of identifying the source causing changes. Now we describe all the steps in detail, including the architecture of the on-line detection system and its packet feature collection mechanism.

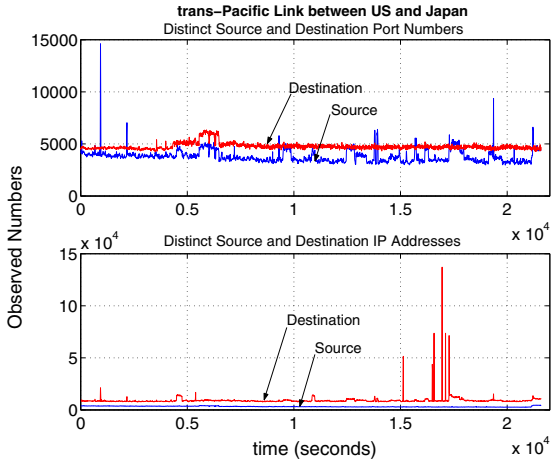
**Traffic monitoring.** Figure 3 shows the *traffic monitoring* module with an DAG [7] capture card, which captures every



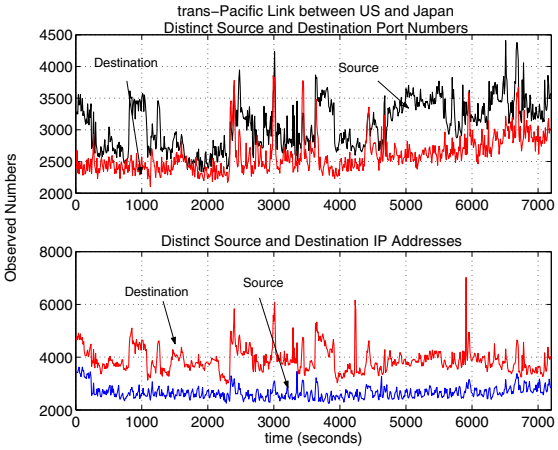

(a.) OC48c Abilene Link [Whole Day, June 18 2004]




(b.) OC48c Abilene Link [2 Hours, May 2002]




(c.) MAWI tarns-Pacific Link [6 Hours, Sept. 22 2005]




(d.) MAWI tarns-Pacific Link [2 Hours, Jan. 10 2007]

Fig. 2. Behavior of Distinct IP Addresses and Port Numbers in Various Traces

packet at a particular network link. The packet attributes (such as source/destination IP addresses and source/destination port numbers) in the packet header are extracted and hashed to unique values. These unique hash values of selected features correspond to *hash bins*, with an initial counter value of 1. Each time, with the arrival of a next incoming packet, if the hashed value of its traffic feature points to an already existing hash bin, then the traffic monitoring module just increments the counter value otherwise creates a new hash bin with an initial counter value of 1. The creation of hash bins and incrementing counter values occur in each epoch. In this paper, epochs are generally referred to as windows. Our implementation of hash bins and counters are based on the hash table. At the end of epoch, hash table is exported for offline analysis.

**Developing distance metrics.** At the end of an epoch, the number of hash bins and its size (i.e., counts) are collected for each of the chosen traffic features. Now, to effectively use this collected information, we develop a *vertical* and a *horizontal* distance metric. The vertical metric computes the distances across a set of selected traffic features in different periods. For example, with a set of traffic features $S = \{src\_ip, dest\_ip, src\_port, dest\_port\}$, only the number of hash bins (not the size of the bins) of each individual feature

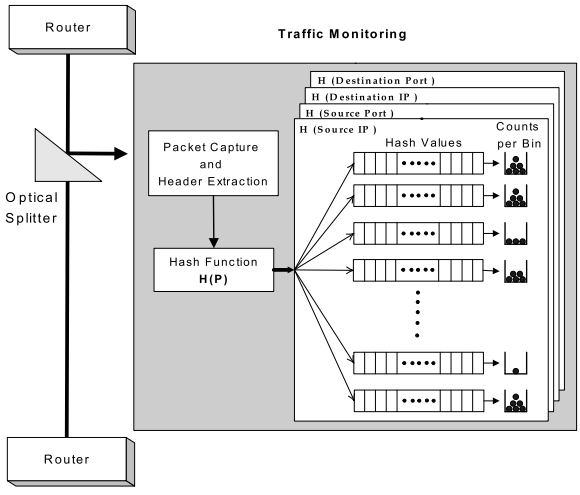



Fig. 3. Architecture of Traffic Monitoring Module

is considered to calculate the vertical distance. More formally, suppose $hb_{\texttt{src\_ip}}$, $hb_{\texttt{dest\_ip}}$, $hb_{\texttt{src\_port}}$, and $hb_{\texttt{dest\_port}}$ are the numbers of observed hash bins during the training period of previous $n$ time windows. Now, the vertical distance between `src_ip` and `dest_ip` (i.e., $S^{\backprime} = \{src\_ip, dest\_ip\}$ and $S^{\backprime} \subset S$) can be calculated by assuming that $\mathbb{P}$ is an array of normalized frequencies of $p_{\texttt{src\_ip}}$ and $p_{\texttt{dest\_ip}}$ collected

over $n$ time windows.

$$p_\alpha = hb_\alpha / hb_{Total}, \text{ where } \alpha \in S^{'} \text{ and } hb_{Total} = \sum_{\alpha}^{S^{'}} hb_\alpha.$$

During the testing period (i.e., at the $(n+1)^{th}$ time window), $\mathbb{Q}$ is an array of normalized frequencies of $q_{\texttt{src\_ip}}$ and $q_{\texttt{dest\_ip}}$. In this $(n+1)^{th}$ time window, we observe $hb^{'}_{\texttt{src\_ip}}$ and $hb^{'}_{\texttt{dest\_ip}}$ hash bins.

$$q_\alpha = hb^{'}_\alpha / hb^{'}_{Total}, \text{ where } \alpha \in S^{'} \text{ and } hb^{'}_{Total} = \sum_{\alpha}^{S^{'}} hb^{'}_\alpha.$$

To calculate the vertical Hellinger distance (or simply the vertical distance) between $\mathbb{P}$ and $\mathbb{Q}$ at the end of $(n+1)^{th}$ sampling period, we use $d^2_H(\mathbb{P}, \mathbb{Q})$. Similarly, we can also calculate the vertical distance between `src_port` and `dest_port`.

The motivation behind the calculation of a vertical distance lies in its considerable diagnostic power in detecting a large set of anomalies. For example, many Internet worms spread by sending random probes (i.e., towards randomly generated destination IP addresses) from an infected computer in an attempt to infect other vulnerable computers. Similarly, in the case of distributed denial-of-service (DDoS) attacks or during the flash crowd events, many hosts send traffic towards a particular host. In all these and similar cases, traffic will demonstrate a dispersed distribution for source (or destination) IP addresses. Port scanning is a popular method for searching open doors through which intruders gain access to computers. An intruder may use vertical (scanning several ports on a single host) scan, horizontal (a particular port is scanned over several hosts) scan, or block (hybrid of vertical and horizontal) scans. In port scanning attacks, destination port numbers or destination IP addresses will show a dispersed distribution. The vertical distances calculated from hash bins across IP addresses or port numbers can accurately capture the dispersion of traffic features, and indicate the occurrence of such attacks mentioned above.

In contrast to the vertical distance, the calculation of a horizontal distance involves only one traffic feature at a time and also takes both hash bins and their size distributions into consideration. Consequently, there will be two horizontal distances for each chosen traffic feature, representing its hash bin's and its size distributional variation with time, respectively. For example, suppose we have chosen a traffic feature $\alpha \in S$, where $\alpha = \{n_i, i = 1, ..., N\}$, meaning that feature $\alpha$ contains $N$ bins and $i^{th}$ bin's size is $n$. Assume that $\mathbb{P}$ is an array of normalized frequencies of bin's size distribution (i.e., $p_i = n_i / \sum_{i=1}^{N} n_i$) collected over the previous $n$ windows data set. During the testing period, i.e., at the $(n+1)^{th}$ time window, $\mathbb{Q}$ is an array of normalized frequencies of the same feature's bin size distribution (i.e., $q_i = n^{'}_i / \sum_{i=1}^{N^{'}} n^{'}_i, i = 1, ....., N^{'}$). Note that in both adjacent training and testing periods, the number of bins and bin size distribution may differ (e.g., $n_i \neq n^{'}_i$ and $N \neq N^{'}$). The Hellinger distance between $\mathbb{P}$ and $\mathbb{Q}$ at the end of $(n+1)^{th}$ sampling period can be calculated using $d^2_H(\mathbb{P}, \mathbb{Q})$.

In the previous example, we compute the horizontal distance for a chosen traffic feature $\alpha$, using the bin's size distribution.

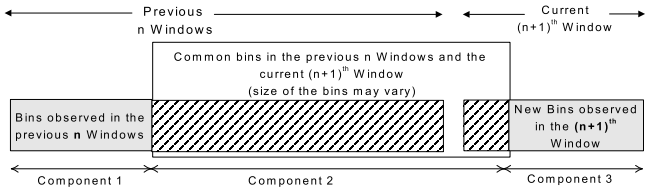



Fig. 4. Behavioral Distance Calculation

We can also calculate another horizontal distance for the same traffic feature $\alpha$ using only the hash bin's distribution (not its size). In this case, $\mathbb{P}$ is an array of normalized frequencies of the hash bin's distribution (i.e., $p_i = 1/N$, i = 1,...,N) in $n$ time windows and $\mathbb{Q}$ is an array of normalized frequencies (i.e., $q_i = 1/N^{'}$, i = 1,...,$N^{'}$) in the $(n+1)^{th}$ time window.

As shown in Figure 4, the horizontal distance $HD_\alpha$ consists of three components. The diagnostic power of horizontal distances in detecting anomalies is mainly attributed to the contributions of components 2 and 3. At the end of $(n+1)^{th}$ time window, component 2 measures how a bin of feature $\alpha$ deviates (or is similar) from that of the previous $n$ windows. For example, if feature $\alpha$ is *source IP address* and a particular host launches a flooding attack in the $(n+1)^{th}$ time window compared to previous $n$ windows, then we observe a sudden jump in the overall distance due to the contribution of component 2. Similarly, if feature $\alpha$ is *destination IP address* and that particular address receives a large number of packets in the $(n+1)^{th}$ time window, then possibly this host is a victim of DDoS attack. There will be a significant impact on component 2 and consequently, on the overall distance. Component 3 measures the deviation due to the generation of new bins of a particular traffic feature $\alpha$. Under DDoS attacks or probing of Internet worms, the contribution of component 3 will be significant and will be reflected in the overall Hellinger distance.

**Traffic features distance measurement.** In the following examples, we choose $N$ unique traffic attributes, which satisfy $p_\alpha, q_\alpha \geq 0$ and $\sum_\alpha p_\alpha = 1$, $\sum_\alpha q_\alpha = 1$. In the case of vertical distance calculations, $\alpha$ represents a traffic feature in the set of $N$ selected traffic features. Whereas, in the case of horizontal distance calculations, $\alpha$ is a hash bin out of $N$ members of a particular traffic feature. Probability measure $\mathbb{P}$ is defined over the training data set and probability measure $\mathbb{Q}$ is defined over the testing data set. Either $\mathbb{P}$ or $\mathbb{Q}$ is hypothesized to be an array of normalized frequencies of all $N$ attributes. The training data set is collected over $n$ sampling periods with duration $\triangle t$ each. This training data set acts as a base for comparison with the data set collected in the testing (i.e., $n+1^{th}$) period.

Using the *Hellinger distance* approach, we measure the similarity between these two data sets. If the measured distance exceeds a threshold, then an alarm is raised to network administrators. If the distance is below the threshold, then the testing data set is included into the previous $(n-1)$ sampled traffic data to derive a new training data set. This moving window mechanism helps the training data set to adapt with the dynamics of network traffic. In our detection scheme, we set the sampling period to 10 seconds to achieve high detection resolution and relatively low CPU and memory overhead. A

longer training period provides more accurate (i.e. Hellinger distance) anomaly detection during the testing period, whereas a shorter training period adapts to the dynamics of network traffic more quickly. To avoid the high cost of maintaining large *Hash tables* and to balance the accuracy and responsiveness, we initially set the training period to 10 seconds (i.e. $n = 1$ sample) in all of the traces while $n$ is a reconfigurable parameter.

**Detection threshold setup.** Based on Hellinger distance, we can profile the normal network traffic behaviors. Now our goal is to seek a threshold of measured distances that can clearly differentiate network anomalies from normal behaviors. During the testing period, once the measured distance is higher than the threshold distance, we will raise an attack alert. The measured distances during the *training period* are used to compute mean $\mu$ and variance $\sigma^2$. Given these two parameters, our task is to determine the validity of the observed distance $d$ in the *testing period*. Let us assume, the measured distance $X$ be a random variable with mean $E(X) = \mu$ and variance $\sigma^2 = var(X)$. Then, the Chebyshev inequality states that :

$$P(|X - \mu| \geq t) \leq \frac{\sigma^2}{t^2}, \quad for \quad any \quad t > 0.$$

Therefore, we can define a band of $\mu \pm 2 * \sigma$ as a *normal region*, where the proportion of observed distances falling in the region is at least $75\%$. Beyond this normal region, the observed distances are anomalous and assigned a severity level depending upon its distance from the normal region. For example, a *low* severity level can be assigned to those distances which fall within the regions $\mu + 2\sigma < d < \mu + 3\sigma$ or $\mu - 3\sigma < d < \mu - 2\sigma$ and *high* severity for those distances where $d > \mu + 3\sigma$ or $d < \mu - 3\sigma$.

## VI. Performance Evaluation

To evaluate the effectiveness and performance of the proposed behavioral distance based anomaly detection mechanism, implemented a software prototype that measures the horizontal and vertical distances of selected traffic features and have tested it with real world traffic traces [16], [19]. The prototype parses traffic traces, extracts header field values and collects the traffic feature data in a hash table to be used for distance calculation. The effectiveness of Hellinger distance and the associated distance metrics is evaluated in terms of its ability to detect and classify network anomalies.

**Performance of vertical distance metric.** In the Internet, most of the protocols are *unicast* in nature, packets are send to a particular host at a time. Our intention of measuring a vertical distances between IP addresses and also between port numbers is to see how closely a traffic stream in a link follows a unicast nature of the traffic. A naive obvious approach is to maintain a relationship between source and destination addresses and observe its behavior. Instead of maintaining such a large number of flows and its prohibitive state maintenance cost, the vertical distance works on the aggregation of distinct IP addresses and port numbers.

In the following example, we choose distinct IP addresses and port numbers as our four traffic features. For the trace obtained from MAWI trans-Pacific link between USA and

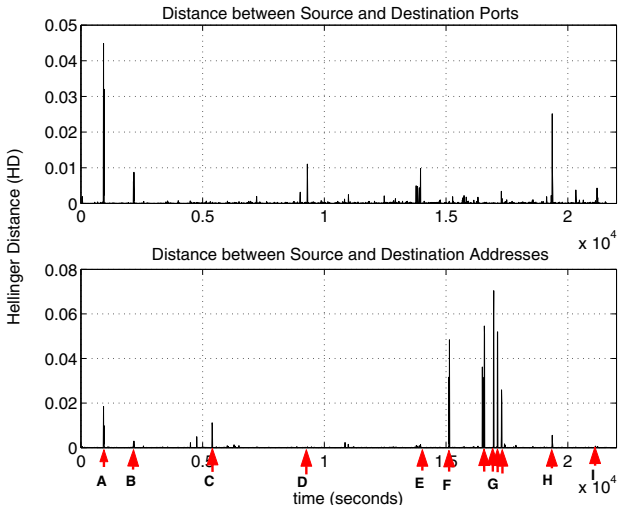


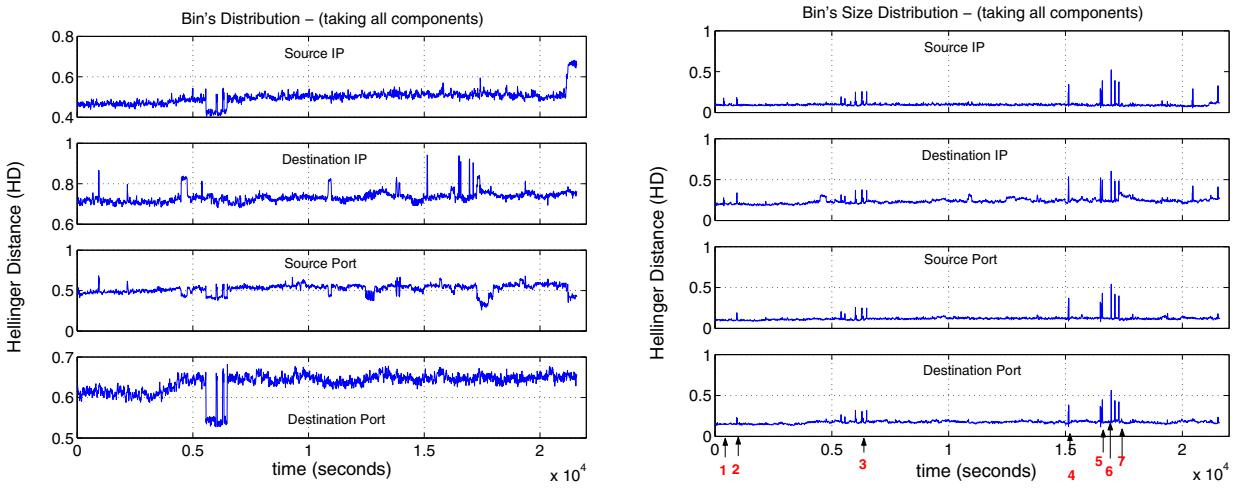

Fig. 5. Vertical Distances in MAWI trans-Pacific Link (US - Japan)

Japan [16], we measured two vertical distances as shown in Figure 5. Throughout the six hours duration, both vertical distances show remarkable similarity except for few spikes. These exceptional distance values represent magnitude of the traffic feature's distributional variations taken as an aggregate. From the Figure 5, we picked sample snapshots of time where spikes are observed. At times $C$, $F$ and $G$, it can be observed that there are spikes in IP addresses vertical distance but at the corresponding times in ports vertical distance there are no spikes. This observation has two implications, first, either one source is trying to reach many destinations (i.e., multicasting) or secondly, many hosts are sending packets to one particular host (i.e., clients are accessing a server). It should also be noted that the second case may ultimately leads to flash crowd event or a DDoS attack towards a particular victim. At times $A, B$ and $H$, both IP addresses and port numbers are showing spikes in their vertical distances. It means that one feature in each of the features set (i.e. address and ports set) is diverging. Finally, at times $D, E$ and $I$, there are spikes in ports vertical distance but IP addresses do not show any corresponding spike. The plausible reason for this behavior could be that at this point of time either source ports or destination ports are diverging. A particular address is either receiving or sending packets at many of its ports, such as in ports scan.

The vertical distance measurements provide a first hand information about distributional changes in the chosen traffic features, but does not classify attacks. Its main shortcomings are:

- Vertical distance measurement cannot tell which traffic feature in a chosen set is varying and that information is important in deriving a real cause of the anomalous behavior.
- It cannot detect any anomaly that arise solely due to traffic feature's size. For example, in a particular flow, if a source suddenly starts sending a large number of packets then this anomalous behavior remains undetected.

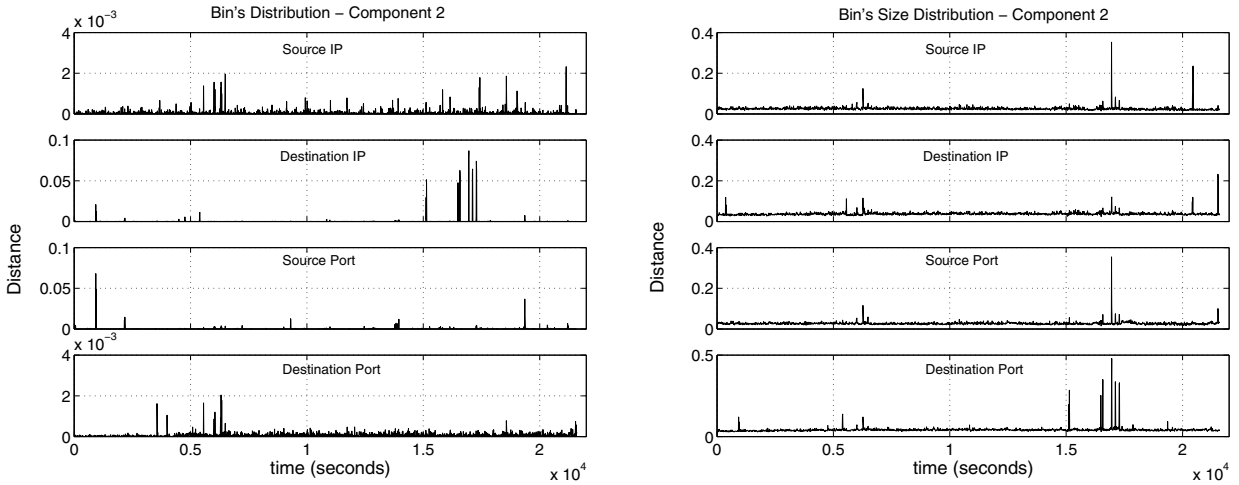
The traffic feature's horizontal distance overcomes these shortcomings of vertical distances and allows fine-grained analysis and classification of anomalous behaviors.




(a.) Variations in bin's distributions

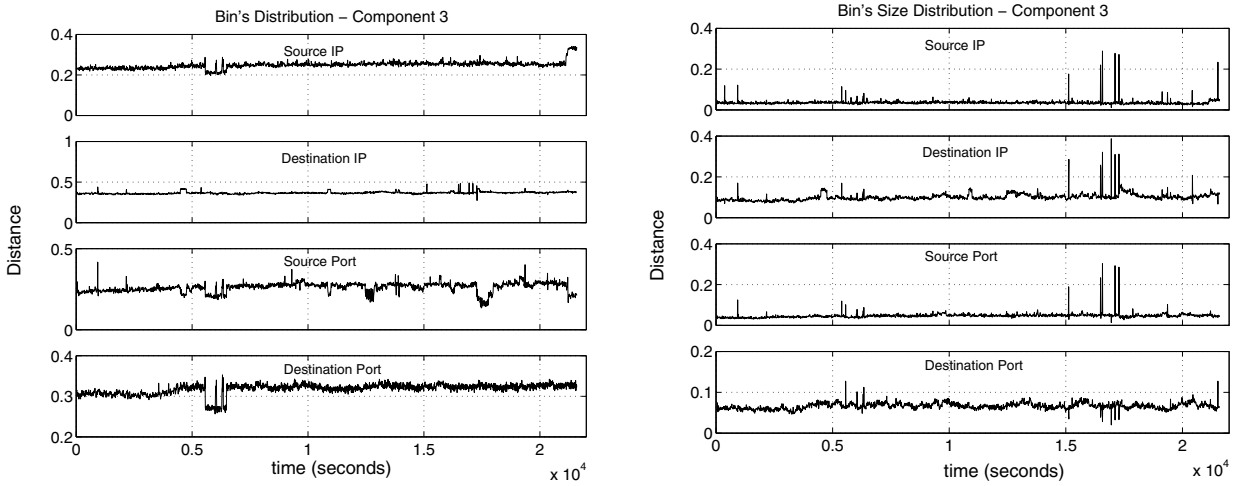
(b.) Variations in bin's size distribution

Fig. 6. Horizontal Hellinger Distance Measurements for MAWI trans-Pacific Link Between US and Japan




(a.) Variations in bin's distributions - Component 2

(b.) Variations in bin's size distribution - Component 2




(c.) Variations in bin's distributions - Component 3

(d.) Variations in bin's size distribution - Component 3

Fig. 7. Horizontal Distance Components for MAWI trans-Pacific Link Between US and Japan

**Performance of horizontal distance metric.** As discussed in section V, the calculation of horizontal distances involves feature's hash bins and its size distributions. In this example, to calculate horizontal distances, we take the same traffic trace as in the previous example. The horizontal distance's *component 2* measures any variations in the existing flows and the effect

of new arrivals. The *Component 3* measures effect of new connections (arrivals) in an observation period. The horizontal distance based on bin's distribution (i.e., distinct instances of a traffic feature) is sensitive to traffic feature's variations and provides a microscopic view of feature's behavior. Whereas, a horizontal distance based on bin's size distribution provides a macroscopic view. It considers overall traffic size and feature's proportional abundance in the traffic data.

Figure 6 plots horizontal Hellinger distance for bin's distribution in (a.) and bin's size distribution in (b.). Throughout the six hours duration, we observed few abnormal behaviors represented as small spikes in Figure 6 (b.). The time of occurrence of these spikes are numbered from 1 to 7. Instead of depending upon a fixed threshold value to raise an alarm, we use components 2 and 3 of the horizontal distance to do the fine-grained analysis. By correlating component's distances, we can find the cause of anomalous behavior and the spikes observed in Figure 6. This correlation-based analysis reduces the number of false alarms. Figure 7 plots horizontal components for bin's and bin's size distributions.

The manual correlation of spikes observed in various component plots are presented in Table I. In the table, column's entry of "Yes" or "No" represents whether a spike is observed or not. Based on these entries, we derive our conclusions which are later verified with the trace also. Out of 7 spikes selected for analysis, we conclude that except the spike at time stamp 3, others are examples of anomalous behavior.

**Detection of Sapphire or Slammer worm.** The Sapphire Worm was the fastest spreading computer worm, at its peak that performed nearly 55 millions scans per second. Many individual sites' bandwidth was saturated with worm copies and there were several reports of Internet backbone disruptions. Sapphire exploited a buffer overflow vulnerability of the computers running Microsoft's SQL servers or Desktop Engines. Once a machine is compromised, the worm propagated by crafting UDP packets and sending them to randomly chosen IP addresses on port number 1434. The Slammer worm infected trace was collected at an OC-3 link connecting Colorodo State University to the Front Range GigaPoP (FRGP) [19]. The trace consists of 8 samples of 90 seconds each, representing the whole day of January 25, 2003, the period overlapping with the onset of Slammer worm.

Figure 8 shows the graphs of horizontal distances of selected traffic features such as $src\_ip$, $dest\_ip$, $src\_port$ and $dest\_port$. The bin's size distribution of each traffic feature is used to calculate the horizontal distances. Figure 8 shows that the traffic samples 3 to 6 are infested with worm packets. The distance between port numbers dips and the distance between destination IP addresses shows a remarkable jump and remains at a high value of 0.8. This clearly explains the onslaught of worm propagation. Because the Sapphire worm used a fixed destination port, it decreases the variability of the destination port number in the traffic. Therefore, the Horizontal distance value for the port number decreases during the propagation of the worm traffic. At the same time, most of the worm packets in the traffic trace have different destination IP addresses. Such a divergence of destination IP addresses results in a high dissimilarity value.

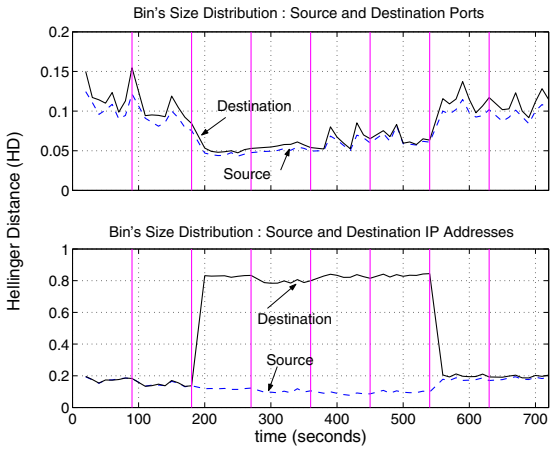



Fig. 8. Horizontal distance using Bin's Size Distribution

## VII. Issues With Our Approach

Our analysis shows that Hellinger distance-based anomaly detection scheme can detect subtle behavioral changes in the traffic features. We now address some of the questions that may arise regarding its capability and detection circumvention.

*A1. Because, the profile adapts during the operational phase, it is possible for a sophisticated attacker to gradually* train *the profile into accepting attacks as normal behavior.* Our profile model is capable of detecting *low rate* attacks that occur for a longer duration of time. In the network anomaly detection phase, we compared previous $n$ windows with the current $(n+1)^{th}$ window. However, low rate attack detection requires the very first window of the training period to be compared with the remaining $(n+1)$ windows. This reverse comparison will make the Hellinger distance to keep on growing for the next $n$ windows and after the low rate attack is fully absorbed in the training profile then the distance suddenly drops. This peculiar behavior of distance distribution identifies low rate attacks within $n$ windows time frame.

*A2. After detecting an abnormal behavior, how do we identify the culprit flows?* This is basically a problem of finding keys in two hash tables, whose values contribute most towards the Hellinger distance measurement.

*A3.Theory vs. Practice - Space Efficient Data Structure.* In our experiments, we used hash table (and hash function) provided by SUN Java library. However, considering the load factor and collision, the construction of hash table containing digests of packet features is a challenging job. We need to adopt a space efficient data structure known as *Bloom filters* [2] for implementing digest tables. Previously, it has been shown that hardware-based Bloom filter (implemented in Field Programmable Gate Array) digest tables are scalable to multi-Gigabit per sec. network links [5], [23].

## VIII. Conclusion

Online detection of network traffic anomalies at high speed links is a challenging task. This paper provides a practical answer to this challenge by showing that (1) real-time network traffic anomaly detection can be made scalable to high speed links; and (2) effective traffic anomaly detection can be built without maintaining per-flow state information. Based on Hellinger distance, we developed a two-dimensional distance

TABLE I

FINE-GRAINED ANALYSIS OF TRAFFIC ANOMALIES

| | Bin's Distribution | | | | | | | | Bin's Size Distribution | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Component 2 | | | | Component 3 | | | | Component 2 | | | | Component 3 | | | |
| | SIP | DIP | SPort | DPort | SIP | DIP | SPort | DPort | SIP | DIP | SPort | DPort | SIP | DIP | SPort | DPort |
| Time Stamp 1 | No | No | No | No | No | No | No | No | No | Yes | No | No | Yes | No | No | No |
| | **Conclusion :** In this time period, there is a sudden surge of packet arrivals where packet's source IP addresses are new (i.e. not seen in the training period) and destination IP addresses are old (i.e. already exist in the training period). It can also be observed that this sudden surge of packets is in between few source and destination IP addresses. There is no effect on the distribution of source and destination port numbers. **Verification :** In this time period, two sources have send 8365 ICMP packets to one particular destination. | | | | | | | | | | | | | | | |
| Time Stamp 2 | No | Yes | Yes | No | No | Yes | Yes | No | No | Yes | No | Yes | Yes | Yes | Yes | No |
| | **Conclusion :** In this time period, there is a sudden surge of packet arrivals that use quite a few new source IP addresses and many new destination IP addresses. There are many new source port numbers, while the destination port numbers are quite few and old (i.e., already exist in the training period). This could be an example of **horizontal port scan** attack. **Verification :** This is an example of *Messenger Spam*, where a particular source has sent 13637 UDP packets to 11214 distinct destination IP addresses using various source port numbers and only two 1026/1027 destination port numbers. | | | | | | | | | | | | | | | |
| Time Stamp 3 | Yes | No | No | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| | **Conclusion :** These are small bursts of flows with an interesting property. The destination ports of the packets are almost concentrating to few particular port numbers. In the traffic there is a considerable amount of packets with the same source IP address and port numbers. **Verification :** This is an example of *BitTorrent P2P* application. | | | | | | | | | | | | | | | |
| Time Stamps 4,5 and 7 | Yes | Yes | No | No | No | Yes | No | No | No | No | No | Yes | Yes | Yes | Yes | Yes |
| | **Conclusion :** In this time period, there is a sudden surge of packet arrivals where packet's source and destination IP addresses are new (i.e., not seen in the training period) and few sources send a large number of packets to many distinct destinations. These packets have same source and destination port numbers, source ports are new and destination ports are old. **Verification :** This is an example of *W32.Spybot* worm, where an infected source IP address performs scans by sending thousands of TCP packets to distinct destination IP addresses using 6000 as source and 1433 destination port numbers. | | | | | | | | | | | | | | | |
| Time Stamp 6 | No | Yes | No | No | No | Yes | No | No | Yes | Yes | Yes | Yes | No | Yes | No | Yes |
| | This example is same as above except that at this time stamp, source IP address already exists in the training period. | | | | | | | | | | | | | | | |

based anomaly detection method that captures both the intra-stream and the inter-stream relationships of chosen attributes. By maintaining running averages over multiple streams, it provides an evolving estimate of complex inter-stream relationships with the efficiency of differential updates, and is therefore more suitable to be used on ISPs. Our empirical results with real Internet traffic confirm that our traffic anomaly detection scheme is able to detect port scans, DoS attacks and slammer worm propagation from the backbone in real-time.

## REFERENCES


[1] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *IMW*, 2002.

[2] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 1970.

[3] J. Brutlag. Aberrant behavior detection in time series for network service monitoring. In *Proceedings of the 14th System Administration Conference*, 2000.

[4] G. Cormode and S. Muthukrishnan. What's new: finding significant differences in network data streams. *IEEE/ACM Trans. on Networking*, 13(6):1219–1232, 2005.

[5] S. Dharmapurikar, P. Krishnamurthy, T. S. Sproull, and J. W. Lockwood. Deep packet inspection using parallel bloom filters. *IEEE Micro*, 24(1):52–61, 2004.

[6] N. Duffield, C. Lund, and M. Thorup. Properties and prediction of flow statistics from sampled packet streams. In *IMW*, 2002.

[7] ENDACE. Network Monitoring Cards. , http://www.endace.com/, 2006.

[8] Fortinet. . Firewall, http://www.fortinet.com/solutions/firewall.html/.

[9] Y. Gu, A. McCallum, and D. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *IMC*, 2005.

[10] R. R. Kompella, S. Singh, and G. Varghese. On scalable attack detection in the network. In *IMC*, 2004.

[11] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. In *IMC*, 2003.

[12] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM*, 2004.

[13] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM*, 2005.

[14] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang. Data streaming algorithms for estimating entropy of network traffic. *SIGMETRICS Perform. Eval. Rev.*, 34(1):145–156, 2006.

[15] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina. Detection and identification of network anomalies using sketch subspaces. In *IMC*, 2006.

[16] MAWI Working Group Traffic Archive. Packet traces from WIDE backbone. 24 Hour Long Traces, http://tracer.csl.sony.co.jp/mawi/, 2006.

[17] Netscreen. Netcreen 100 Firewall Appliance. Firewall, http://www.netscreen.com/.

[18] NLANR, Abilene. NLANR network traffic traces. Special Tarces, http://pma.nlanr.net/, 2005.

[19] NLANR, Front Range GigaPOP. NLANR network traffic traces. Daily traffic traces, http://pma.nlanr.net/Traces/, 2005.

[20] H. Ringerg, A. Soule, J. Rexford, and C. Diot. Sensitivity of pca for traffic anomaly detection. In *SIGMETRICS*, 2007.

[21] M. Roesch. Snort: Lightweight intrusion detection for networks. In *USENIX LISA*, 1999.

[22] R. Schweller, A. Gupta, E. Parsons, and Y. Chen. Reversible sketches for efficient and accurate change detection over network data streams. In *IMC*, 2004.

[23] A. C. Snoeren. Hash-based ip traceback. *SIGCOMM Comput. Commun. Rev.*, 31(4):3–14, 2001.

[24] R. Sommer and V. Paxson. Enhancing byte-level network intrusion detection signatures with context. In *CCS*, 2003.

[25] A. Soule, K. Salamatian, and N. Taft. Combining filtering and statistical methods for anomaly detection. In *IMC*, 2005.

[26] H. Wang, D. Zhang, and K. G. Shin. Detecting syn flooding attacks. In *IEEE INFOCOM*, June 2002.

[27] K. Xu, F. Wang, S. Bhattacharyya, and Z.-L. Zhang. A real-time network traffic profiling system. In *DSN*, 2007.

[28] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *IMC*, 2005.

[29] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online identification of hierarchical heavy hitters: Algorithms, evaluation, and applications. In *IMC*, 2004.