Is Sampled Data Sufficient for Anomaly Detection?

Jianning Mai Chen-Nee Chuah
University of California, Davis
2064 Kemper Hall
Davis, CA 95616-5294
{jnmai,chuah}@ece.ucdavis.edu

Ashwin Sridharan Tao Ye Hui Zang
Sprint Advanced Technology Labs
One Adrian Court
Burlingame, CA 94010
{Ashwin.Sridharan,Tao.Ye,Hui.Zang}@sprint.com

ABSTRACT

Sampling techniques are widely used for traffic measurements at high link speed to conserve router resources. Traditionally, sampled traffic data is used for network management tasks such as traffic matrix estimations, but recently it has also been used in numerous anomaly detection algorithms, as security analysis becomes increasingly critical for network providers. While the impact of sampling on traffic engineering metrics such as flow size and mean rate is well studied, its impact on anomaly detection remains an open question.

This paper presents a comprehensive study on whether existing sampling techniques distort traffic features critical for effective anomaly detection. We sampled packet traces captured from a Tier-1 IP-backbone using four popular methods: random packet sampling, random flow sampling, smart sampling, and sample-and-hold. The sampled data is then used as input to detect two common classes of anomalies: volume anomalies and port scans. Since it is infeasible to enumerate all existing solutions, we study three representative algorithms: a wavelet-based volume anomaly detection and two portscan detection algorithms based on hypotheses testing. Our results show that all the four sampling methods introduce fundamental bias that degrades the performance of the three detection schemes, however the degradation curves are very different. We also identify the traffic features critical for anomaly detection and analyze how they are affected by sampling. Our work demonstrates the need for better measurement techniques, since anomaly detection operates on a drastically different information region, which is often overlooked by existing traffic accounting methods that target heavy-hitters.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*; C.4 [**Performance of Systems**]: Design studies

IMC'06, October 25–27,2006,Rio de Janeiro, Brazil.

General Terms

Measurement, Performance, Experimentation

Keywords

sampling, anomaly detection, volume anomaly, portscan

1. INTRODUCTION

Accurate traffic measurement is essential for both network management tasks and security forensics. To cope with increasing link speed, sampling techniques are often deployed at routers, e.g., Cisco's NetFlow [1] and Juniper's Traffic Sampling [2], to reduce measurement overhead in terms of router CPU, memory, and bandwidth. Traditionally, sampled traffic data is used by network providers for capacity planning and traffic engineering (TE) tasks, such as computing traffic matrices. In recent years, sampled traffic data has also been used as input for anomaly detection [4,16,19], e.g., detecting denial-of-service (DoS) attacks or worm scans. It is well known that sampling distorts traffic statistics such as mean rate and flow size distribution. There have been numerous studies on improving the estimation accuracy of flow-level statistics from sampled data [7–9, 13]. Alternative traffic accounting methods have also been proposed to track and improve measurement accuracy of heavy-hitters for TE purposes. However, anomaly detection often operates on a different region of information than TE metrics. For example, information such as address access patterns, connection status, or per source behavior of small flows is important for portscan detection. Hence, the impact of sampling on anomaly detection remains an open question.

On the other hand, conventional intrusion detection systems are typically deployed at network edges where detailed payload data is available. It is not clear if similar solutions can be effective in diagnosing network-wide anomalies using only sampled packet headers from high-speed backbone networks. Nevertheless, monitoring at backbone transit networks offers a unique opportunity to study a more diverse traffic mix from various vantage points. A more global traffic view can better capture anomalous patterns and detect a wider range of scanning activities [10]. It becomes increasingly critical for Internet service providers (ISPs) to detect anomalous traffic to ensure quality of service and provide value-added services.

This paper focuses on the use of traffic measurements from high-speed IP-backbone networks for anomaly detection and seeks to answer this question: *Does sampled data capture sufficient information for effective anomaly detection?*

Our initial work [17] shows that random packet sampling indeed distorts traffic features that are critical for different portscan detection techniques. This paper documents a more thorough study that compares four popular sampling methods: random packet sampling, flow sampling, smart sampling [11], and sample-and-hold [9]. Packet traces captured from a Tier-1 IP-backbone are sampled using these four methods and then used as input to detect two common classes of anomalies that are visible to ISPs: volume-based change detection and portscan detection. Port scanning [3, 14, 18, 19] is usually associated with worm or virus propagation, while volume anomalies [4, 5, 21] can be due to a variety of reasons, including DoS attacks and flash crowds.

Since it is not possible to evaluate and compare all the anomaly detection algorithms that have been previously proposed, we choose to study three representative techniques. For volume anomaly detection, we evaluate a wavelet analysis approach previously proposed by Barford *et al.* [4]. For portscan detection, we evaluate the Threshold Random Walk (TRW) [14] and Time Access Pattern Scheme (TAPS) [19]. These three schemes differ in the ways they profile the traffic for detection. The wavelet-based approach detects local abrupt changes in volume "relative" to global statistics. TRW follows the philosophy of doing stateful analysis of the underlying traffic, while TAPS leverages the knowledge of the "behavioral pattern" of port scanners. While these algorithms merit evaluation in their own respect, more importantly, they cover a reasonably wide range of metrics, e.g. connection pattern, scanning rate, and volume changes that are typically used in anomaly detection algorithms. We not only evaluate the performance of these algorithms, but also identify how the metrics used in decision making are affected by the different sampling techniques and the sampling rate. Since these metrics are commonly used in anomaly detection, we expect that our results will be widely applicable.

Our contributions and findings are summarized as follows:

- Through experiments using real traffic traces, we quantify how four different sampling schemes (random packet sampling, random flow sampling, smart sampling, and sample-and-hold) affect the performance of a wavelet-based volume anomaly detection method and two portscan detection algorithms. We also perform sensitivity analysis with respect to different sampling rates and detection parameters. Our results show that sampling adversely impacts the ratio of successful detection for both volume anomalies and port scans which are non-volume based.

- Through analysis, we pinpoint important traffic "features" that are distorted due to sampling and have adverse impacts on different anomaly detection schemes. We demonstrate that sampling causes the following fundamental bias in data: (a) a decrease in the deviation of local variance from global variance, which diminishes the effectiveness of the wavelet-based volume anomaly detection method (Section 3), and (b) a shortened flow size distribution, which causes high false positive and false negative ratios for scanner detection schemes that rely on source access patterns (Section 4).

- Among the four sampling schemes, random flow sampling introduces the least amount of distortion to the traffic features critical for volume anomaly and portscan detections. Since random flow sampling was originally proposed to improve estimation accuracy of flow statistics, it seems to be an ideal solution that works well for both traffic engineering and anomaly detection purposes. However, its heavy resource requirements and poor scalability will continue to prohibit its wide-spread deployment. On the other hand, random packet sampling performs the worst whereas it is the easiest to implement. Packet sampling decreases the deviation of local variance from global variance in the flow arrival time series, hence adversely impacting the wavelet-based volume detection method. Its strong flow shortening effect introduces a high number of false positives in portscan detection, which is undesirable from a network administrator point of view. Smart sampling and sample-and-hold are less resource-intensive than random flow sampling, but they perform poorly in the context of anomaly detection. Being 'biased' towards accurate estimation of heavy-hitters, they fail to capture small-sized flows, which are often the sources of many attacks.

The remainder of the paper is organized as follows. Section 2 provides background discussion on the sampling schemes, anomaly detection algorithms, and data set we analyzed. In Section 3, we demonstrate and analyze the effect of sampling on wavelet-based volume change detection technique. Section 4 compares the impact of sampling on two specific portscan detection schemes TRW and TAPS. We discuss the implications of this study and some future directions in Section 5.

2. BACKGROUND AND METHODS

2.1 Sampling

Sampling has been introduced in network measurement to alleviate large memory and CPU power requirements on the routers as well as high bandwidth usage in the network to transport the collected data records [7]. Two categories of sampling have been widely discussed in literature: packet sampling and flow sampling. *Packet sampling* is simple to implement with low CPU power and memory requirements. However, extensive research [8,13]has shown it to be inaccurate for inference of flow statistics such as the original flow size distribution. *Adaptive* packet sampling techniques that adjust the sampling rate to traffic load to further reduce memory consumption [10] or to improve accuracy [6] have also been presented.

Flow sampling emerges as an alternative to overcome the limitations of packet sampling. It is shown [13] to improve accuracy but still suffers from prohibitive memory and CPU power requirements. To partially address these issues, especially to reduce memory and bandwidth requirements, techniques like *smart sampling* [8] and *sample-and-hold* [11] have been proposed as two variants of flow sampling with a focus on accurate estimation of heavy-hitters. In this study, we evaluate how the performance of anomaly detection algorithms is affected by these sampling algorithms. In particular, we focus on random packet sampling and its flow equivalent, random flow sampling as well as the two aforementioned variants: smart sampling and sample-and-hold. Below we briefly describe the four sampling techniques.

Random packet sampling Random packet sampling simply samples a packet with a small probability $r < 1$. The sampled traffic is then classified into flows based on the five-tussle: (Source IP address, destination IP address, source port, destination port, protocol). Flows are terminated by either a default timeout of 1 minute or explicit TCP protocol semantics. This process emulates the behavior of the NetFlow [1] random sampling process.

Random flow sampling Random flow sampling first classifies packets into flows based on the flow construction rules mentioned above. It then samples each flow with some probability $p < 1$.

Smart sampling Proposed by Duffield *et al.* [8], this technique is a size-dependent flow record selection algorithm and applies to complete flow records. Given a set of flows of sizes $S = \{x_i : i = 1, \cdots, n\}$, smart sampling selects a flow of size x with a probability $p(x)$ to form a set of selected flows of S'. The goal is to achieve an unbiased estimator $X' = \sum_{x' \in S'} x'/p(x')$ of the total byte count $X = \sum_{x \in S} x$. The following solution was shown to be optimal in terms of balancing the opposing constraints of keeping the variance of X' small, while reducing the sample size $N' = |S'|$:

$$p(x_i) = p_z(x_i) = \begin{cases} \frac{x_i}{z} & \text{if } x_i < z \\ 1 & \text{if } x_i \geq z \end{cases}$$

where z is a threshold that trades off accuracy for reduction in bandwidth requirement.

Sample-and-hold (S&H) Sample-and-hold [11] performs a flow table lookup for each incoming packet to see if a flow entry for it exists. If found, the entry is updated. Otherwise, the packet is randomly sampled and a flow entry created with a probability h_s. h_s is chosen as if each byte is sampled with a probability h. Thus the sampling probability for a packet of size s is given by $h_s = 1 - (1 - h)^s \approx h \cdot s$. Unlike random packet sampling, a flow entry gets updated by all the subsequent packets once it is created in S&H. Hence, it requires flow table lookups for all incoming packets, though the memory size for the flow table is reduced due to its non-uniform sampling biased toward "elephant" flows.

As mentioned earlier, network operators are increasingly utilizing sampled traffic data for detecting anomalies such as denial-of-service attacks or zero-day attacks. These attacks can be broadly classified into two categories: volume anomalies which capture the former type of attack, and portscan anomalies which characterize the latter. We describe them in detail below and also discuss the detection algorithms for each class that we utilize in our evaluation.

2.2 Volume Anomaly Detection

Network traffic anomalies such as DoS attacks or flash crowds often manifest themselves as abrupt changes in packet or flow count measurements. In other words, they cause volume anomalies. This has drawn enormous attention over the years and a number of detection schemes have been proposed [4, 5, 15, 21]. Signal processing techniques [4, 21] have been adopted to detect this type of anomaly. Barford *et al.* [4] applied wavelet filters to both SNMP MIB data and IP flow data to detect sudden increase in the local variance of the time series, while Thottan and Ji [21] used an auto-regressive (AR) process to detect abrupt changes in a few MIB variables. Wavelet analysis facilitates multi-resolution analysis (MRA) of time-frequency traffic characteristics, and has proved to be effective at detecting volume anomalies. We focus our evaluation on a discrete wavelet transform (DWT) based detection procedure [4] and briefly discussed below.

DWT-based detection is an off-line algorithm that applies wavelet decomposition to either the packet or flow rate time series obtained from traffic traces to detect volume changes at various time scales. It basically comprises of three steps: decomposition of the time series, re-synthesis, and detection.

Decomposition: Broadly speaking, the goal of this step is to decompose the original signal so as to be able to identify changes taking place at various time scales. This is done as follows. The DWT calculates wavelet coefficients at dyadic scales and can be treated as filtering operations. The original signal $\{X[n]\}$ of length N is passed through a real-valued wavelet filter (high pass) $\{h_l\}$ of even width L. The output is a set of wavelet coefficients $\{W_{1,t}\}$ at the original time scale (first level) with length $\frac{N}{2}$. Similarly by filtering the signal with the corresponding scaling filter (low pass) $\{g_l = (-1)^{l+1} h_{L-1-l}\}$, we obtain the first level scaling coefficients $\{V_{1,t}\}$ of length $\frac{N}{2}$. By applying the inverse DWT on $\{W_{1,t}\}$ and $\{V_{1,t}\}$ independently, we separate the first level detail D_1 from approximation A_1. The decomposition process is then repeated using A_1 as an input, which yields D_2 and A_2 at the second level. The maximum level of DWT we can perform is $j \leq \log_2 N$. The MRA thus satisfies $X = \sum_{i=1}^{j} D_i + A_j$. Each level j represents the strength of a particular frequency in the signal, with a higher value of j indicating a lower frequency.

***Re-synthesis*:** Once various frequency levels (or decomposition levels j) in the signal have been identified, they are aggregated into *low*, *mid* and *high* bands. The low-band signal $X_{low} = \sum_{i=l}^{j} D_i + A_j$ consists of the highest $(j - l + 1)$ levels of decompositions and identifies slow-varying/long-term trends. The high-band signal $X_{high} = \sum_{i=1}^{h} D_i$ is the sum of the lowest h levels of details and highlights sudden variations in the volume of the traffic. The mid band is the sum of the rest.

***Detection*:** Volume anomalies are detected as follows. The local variance of the high and mid-band signals is computed over a time interval determined by a sliding window. A metric called the *deviation score*, which is the ratio between the local variance within the window and the global variance is computed for each such window. Windows with deviation scores higher than a predefined threshold are marked as volume anomalies. It was shown [4] that the technique is quite effective at capturing different types of volume anomalies.

2.3 Portscan Detection

Several portscan detection techniques have been proposed in literature. For instance, Snort [3] is a flexible open-source intrusion detection system that issues scan alerts based on

user-defined connection patterns and rates. SPICE [20] performs a complex off-line Bayesian analysis to detect stealthy port scans. In this paper, however, we focus on two effective "on-line" portscan detection techniques: *Threshold Random Walk* (TRW) [14] and *Time Access Pattern Scheme* (TAPS) [19]. Both algorithms are described briefly below. A summary with more details can be found in our previous paper [17].

TRW and TRWSYN TRW forms two hypothesis, H_0 that a source is a "benign" host and H_1 that a source is a "scanner", characterized by the likelihood of the success or failure of a connection. The rationale for this definition is that a benign host is far more likely to have successful connections than a scanner which randomly probes the address space. Hence H_0 is the hypothesis associated with high connection success probability and H_1 with failure. The technique performs hypotheses testing on a sequence of observed events, which in this case is connection status, to determine which hypothesis is more likely for the source. In particular, let $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_i\}$ represent the random vector of connections observed from a source, where $Y_i = 0$ if the i^{th} connection is successful and $Y_i = 1$ otherwise. For each observed value Y_i, the likelihood ratio defined as:

$$\Lambda(Y) = \prod_{i=1}^{n} \frac{Pr[Y_i|H_1]}{Pr[Y_i|H_0]} \quad (1)$$

gets updated. It is not hard to see that the behavior of $\Lambda(Y)$ is similar to a random walk where the i^{th} jump is governed by the status of the i^{th} connection. When $\Lambda(Y)$ crosses either one of two predefined thresholds, the corresponding hypothesis is selected as the most likely. It was shown [14] that TRW only requires ~ 6 observed events to detect scanners successfully.

TRWSYN [19] is the backbone adaptation of TRW to accommodate the fact that observed aggregate traffic on backbone links is usually uni-directional, which makes predicting whether a connection "failed" or "succeeded" much more difficult. The modified oracle marks a single SYN-packet flows as failed connections, while others are assumed to be successful. Therefore TRWSYN can detect TCP portscan only.

TAPS Unlike TRW, which traces the state of connections, TAPS [19] utilizes the the *access pattern* to separate scanners from benign hosts. This is based on the observation that a scanner often initiates connections to a larger spread of destination IP addresses (horizontal scan), or port numbers (vertical scan). In other words, the ratio γ between distinct destination IP addresses and port numbers (or its reciprocal, whichever is bigger) for a scanner is far larger than a non-scanner.

Instead of using a simple threshold like Snort (which also utilizes connection pattern behavior), TAPS combines a rate limiting scheme for event generation with a sequential hypotheses test similar to TRW in order to achieve fast detection as well as lower false positive rate. In order to be protocol agnostic, TAPS associates a single packet flow with a failed connection. The algorithm works as follows. In each time bin (say i), for each source, the ratio γ is computed and compared to a predefined threshold k. The event variable Y_i associated with that time bin is then set to 0 or 1 depending on whether γ exceeds or lies below the threshold. The likelihood ratio is then updated in the same manner as for TRW based on the value of Y_i and a decision regarding which hypothesis applies to the source is made based on which of the two thresholds is crossed first. It is clear from the above discussion that the threshold k affects the accuracy of detection, while time bin size controls the promptness of decision making.

2.4 Trace Data

Our experiments used packet traces collected on three links in a Tier-1 ISP's backbone network: **BB-West** and **BB-East** were from two OC-48 links between backbone routers on the west coast and east coast respectively, while the **Wireless** trace was collected from a link connecting gateway routers to a nation-wide cellular network. These traces were collected by IPMON [12], a passive monitoring system that captures the first 64 bytes of the IP header of every IP packet traversing a monitored link. The large volume *BB-East* trace contains know DoS attacks and is ideal for evaluating the wavelet-based anomaly detection algorithm. The *BB-West* trace and the *Wireless* trace have a relatively low traffic volume but contain a large percentage of scanning traffic, making them interesting for portscan analysis. Statistics of the traces are presented in Table 1.

Table 1: Trace Data Statistics

Trace	Date	Average Rate	Duration
BB-East	04-07-2003	207Mbps	17h
BB-West	03-08-2003	55Mbps	1 hour
Wireless	04-01-2004	7Mbps	3 hours

2.5 Methodology

We generated *sampled traces* from the *original traces* using the four sampling schemes, namely random packet sampling, random flow sampling, smart sampling and sample-and-hold which were then used as inputs for the anomaly detection algorithms.

Since all the four sampling schemes use different parameters for sampling, we require a common metric that allows us to obtain a fair comparison. A possible choice could be based on resource consumption, namely CPU utilization or the required storage memory. However, for the purposes of this work, we chose the *percentage* of sampled flows as the common metric. Specifically, we set the parameters of each sampling algorithm such that they sampled approximately the same number of flows. We now motivate our choice of the number of sampled flows for the purposes of normalization. First and foremost, the focus of this work is on evaluation of these sampling algorithms within the framework of anomaly detection which is heavily based on flow measurement. In particular, flow arrival time series are utilized by the wavelet-based volume anomaly detection, and exported flow records are required for TRWSYN and TAPS, respectively. The anomaly detection results for different sampling schemes would not be comparable unless the total number of flows sampled are the same. Secondly, it can be argued that memory requirements for these algorithms (at least the slow DRAM memory) is dominated by the storage requirements

for the sampled flow records and hence also normalized since we normalize the number of sampled flows. Third, the CPU utilization (as well as fast SRAM memory access) is strongly influenced by the complexity of flow-table lookups for each selected packet. This can be highly dependent on various algorithmic implementations, CPU configurations etc., hence potentially hard to normalize. For example, flow table lookups need be performed only for the sampled packets in packet sampling, but need to be done for every packet in all other schemes. Packet sampling is clearly the least CPU intensive, but the other three schemes would have to be differentiated based on algorithms and data structures used in their respective implementations. In any case, for completeness, we list the fraction of sampled packets in Table 2 for each sampling scheme which loosely reflects the comparative CPU usage.

Guided by this metric, the parameters for the various sampling schemes were chosen as follows. We first generated sampled traces with the random packet sampling, where each packet is sampled with probability $r = 1/N$ and the average sampling interval N takes values from the set $\{10, 20, 50, 100, 200, 500, 1000\}$. The parameters for the flow-based sampling schemes were then chosen as follows. Assume the probability of a flow having n packets in the original trace is $f(n)$, and the maximum flow size is M packets. For smart sampling with a threshold z, the ratio of flow selection is given by:

$$\sum_{n=1}^{z-1} f(n) \cdot \frac{n}{z} + \sum_{n=z}^{M} f(n).$$

In sample-and-hold, the flow sampling ratio can be calculated as:

$$\sum_{n=1}^{M} f(n)(1 - (1 - h_s)^n) \approx f(n)nhs,$$

where s is the average packet size. In each case, the flow sampling ratio was chosen such that it resulted in the same number of flows as with packet sampling. The choice of the sampling parameter for random flow sampling is straightforward: it is simply the ratio of the number of sampled flows (with packet sampling) to the original number of flows in the trace. Table 2 lists the parameters setting in various sampling methods for the *BB-West* trace. Note that although the proportions of the exported flows are fixed, smart sampling and sample-and-hold sampled much higher percentages of packets due to their bias toward heavy-hitters.

To compare the impact of these schemes on anomaly detection, the wavelet-based volume change detection and portscan detection algorithms were applied to both the original and sampled traces and their performance compared.

3. IMPACT OF SAMPLING ON VOLUME ANOMALY DETECTION

Our measurement data used for volume anomaly detection consists of the flow arrival rate time series. We obtained the time series by counting the number of new flows in each time interval. For ease of exposition, we shall refer to the time series extracted from the original trace as *the original time series*, and that from a sampled packet trace file by *sampled time series*. We apply wavelet analysis to detect volume anomalies in the original time series, and use the result as the baseline to compare detections from the sampled time series. Note that we have not attempted to identify the root causes of those anomalies, since our focus is on studying the impact of sampling schemes on the efficacy of wavelet-based abrupt change detection method.

In our experiment, we use the *BB-East* Trace, which contains about 17 hours of packet headers on a backbone link. The "db5" wavelet family we choose has a support length of 9 and vanishing moments of 5, a good trade-off between time and frequency localizations for our purpose. We decompose the signal with a 12-level DWT as explained in Section 2.2. To re-synthesize it, we set $h = 5$ and $l = 10$ so that the high band signal details short-term ($\sim$ seconds) changes, the mid band displays variation on a scale of a minute, and the low band shows trend over 15-minute intervals. The sliding window size for computing local variance at high and mid bands is one minute. We set the average deviation threshold at 4.5, simply to make sure that every single obvious *spike* will be detected in the original time series. Thus detection results from the sampled time series can be easily compared.

3.1 Volume Anomaly Detection Results

Figures 1 shows the volume anomalies detected in between the dotted lines across high and mid-bands from the original time series. There are a total of 21 abrupt changes in the flowing arrival process.

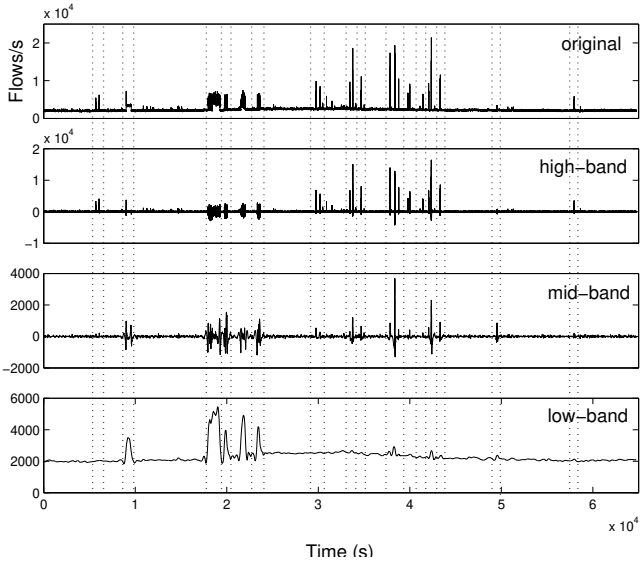


Figure 1: The *BB-East* Trace: Detection from the original trace, 21 volume anomalies found.

Table 3 summarizes the number of volume anomalies detected from various sampling techniques. The percentage of flows sampled corresponds to sampling interval of 10, 100 and 1000 in random packet sampling. As is to be expected, the number of detections goes down as sampling interval increases. Our results indicate that random flow sampling performs the best over all sampling rates amongst all sampling schemes. For example, it experiences no performance degradation at a sampling interval $N = 10$. Detection results from smart sampling and sample-and-hold drops much faster, and when the sampled flow percentage drops below 10%, they can only detect one or two volume surges. We

Table 2: Sampling Parameters and Sampled Packet Percentage for the *BB-West* Trace

% flows	random packet sampling		random flow sampling		smart sampling		sample-and-hold	
	r	% pkts	p	% pkts	z	% pkts	h	% pkt
34.4%	0.1	10.0%	0.344	34.4%	11	84.5%	1.26×10^{-3}	75.9%
22.3%	0.05	5.00%	0.223	22.6%	19	77.7%	5.38×10^{-4}	68.7%
11.7%	0.02	2.00%	0.117	11.5%	41	68.9%	1.70×10^{-4}	58.8%
6.91%	0.01	1.00%	0.691	6.96%	75	62.7%	7.10×10^{-5}	51.5%
3.90%	0.005	0.500%	0.0390	4.15%	145	56.6%	3.16×10^{-5}	44.7%
1.73%	0.002	0.201%	0.0173	1.78%	364	49.3%	1.10×10^{-5}	36.1%

note that there are no false positives in detection for any of the sampling methods. Next, we attempt to identify the causes behind the differing performance of the sampling schemes.

Table 3: Number of Volume Anomalies Detected under Various Sampling Methods

Sampling interval	10	100	1000
Percentage of flows (%)	36.7	8.03	1.47
Random packet sampling	19	6	1
Random flow sampling	21	18	13
Smart sampling	18	1	1
Sample-and-hold	18	2	1

Figure 2 shows the detected anomalies from the random packet sampling under increasing sampling intervals. Figures 3(a), 3(b), and 3(c) presents the results from random flow sampling, smart sampling, and sample-and-hold under effective sampling interval $N = 100$, respectively.

3.2 Feature Variation Due to Sampling

Even though random flow sampling generates the highest number of successful detections as shown in Table 3, the number of anomalies detected actually decreases when sampling interval increases. We observe also that signals become much noisier, especially at the high frequency band. Since the original technique is based upon utilizing the local variance and sampling reduces the number of observed events, we hypothesize that sampling introduces distortion in the variance of the time series.

To help explain the source of this distortion, we provide a simple, approximate explanation based on random flow sampling. We consider the flow arrivals as belonging to a stationary i.i.d point process, given by $\{X_t\} = N(t-1,t], t = 0, 1, 2, \ldots$, where $N(t-1,t]$ is the number of incoming flows in a unit time interval. Denote the variance of this process by σ_X^2 and the average rate by $E[N(t-1,t]] = \lambda$.

Now, if we were to model the impact of sampling as simply scaling down the original time series $\{X_t\}$ by a fraction p, it is easily seen that the variance of the new *scaled-down* process $\{pX_t\}$ should be $\sigma_{pX}^2 = p^2\sigma_X^2$, i.e., scaling should *reduce* the variance. However, in practice, sampling involves removal of discrete points from the process rather than simple scaling. In other words, we sample the original point process binomially in each time unit. We assume that this process *adds* a variance term given simply by that of a binomial random variable. In particular, if there were n flows in a time unit, then the variance is given by $np(1-p)$. Hence, on an average we approximate the variance term added by $E[n]p(1-p) = \lambda p(1-p)$.

Therefore, the total variance of the sampled process becomes

$$\sigma_{X^{(p)}}^2 = p^2\sigma_X^2 + \lambda p(1-p) \tag{2}$$

The extra part of the variance in Eqn. (2) is due to the random sampling process. Therefore we call it the sampling variance $\sigma_S^2 = p(1-p)\lambda$. It depends only on the sampling probability and flow arrival rate of the original trace.

Intuitively, after random flow sampling, each point in the flow counting process $N_s(t-1,t]$ varies around the center value $pN(t-1,t]$. Although the mean scales down to $E(N_s(t-1,t]) = pE(N(t-1,t])$, the variance may increase as shown above. To illustrate how much increase is observed, we plot the ratio of σ_S^2/σ_{pX}^2 in Fig. 4. We find that the percentage of the sampling variance in the total variance increases with the sampling interval, and almost reaches 80% at the sampling interval of 1000.

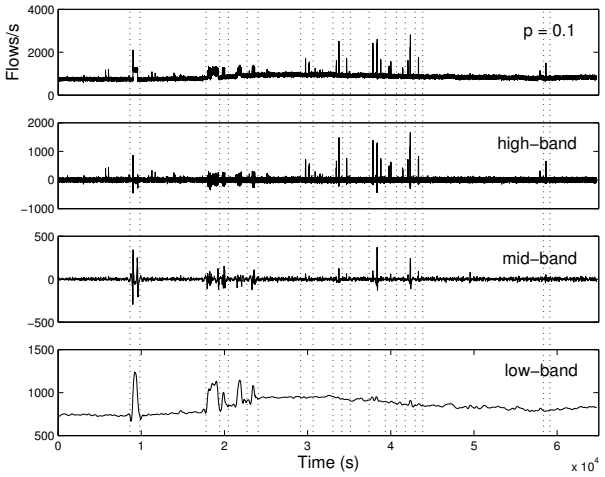
Interestingly, the simple analysis above also highlights why we see *no false positives* (unlike those observed for portscan detection in Section 4) relative to the original time series. Specifically, if we detect a spike in traffic volume in the sampled time series, it must have existed in the original time series also and is not an artifact of sampling. Let the local variance over any time window in the sampled time series is given by v_L' and the global variance by σ_{pX}^2. Based on the algorithm, if we choose a threshold t, then we detect anomalies whenever, $v_L' \geq t \cdot \sigma_{pX}^2$.

If we were to replace v_L' and σ_{pX}^2 with their counterparts in the original time series from Equation 2, we obtain:

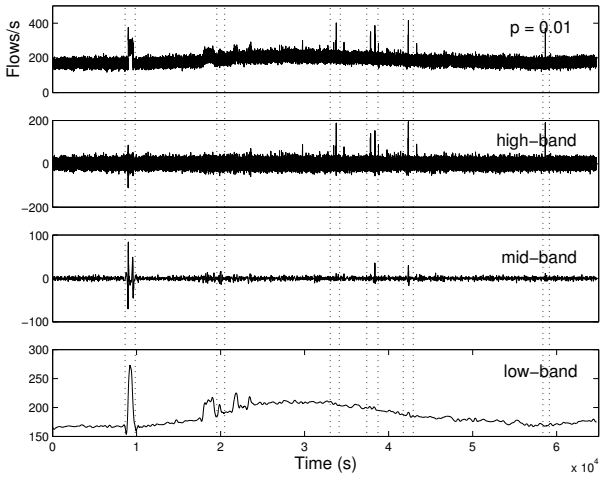
$$v_L \geq t \cdot \sigma_X^2 + (t-1)\lambda(\frac{1-p}{p}). \tag{3}$$

The above equation states that the local variance for the original time series would also exceed the threshold compared to the original global variance.

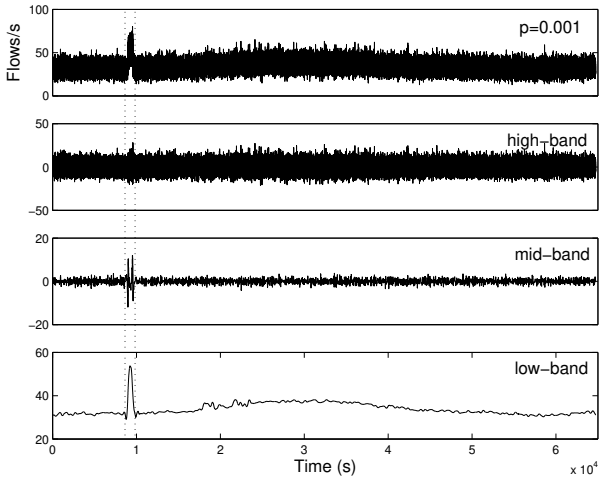
Finally, we discuss potential reasons for the the difference in performance observed in Table 3. Most of the volume spikes observed in our traces are caused by a sudden increase in *small* packet flows. Note that random flow sampling is completely unbiased by flow size, while packet sampling, sample-and-hold and smart sampling are biased towards sampling large flows. Given the unbiased nature of flow sampling, it is able to sample a sufficient number of such small flows thereby providing good detection results. Smart sampling and sample-and-hold are specifically designed to track heavy hitters and hence are even more biased towards heavy flows than packet sampling. This explains their relatively poor performance compared to packet sampling in terms of detecting volume anomalies.



(a) sampling rate $p = 0.1$; anomalies = 19

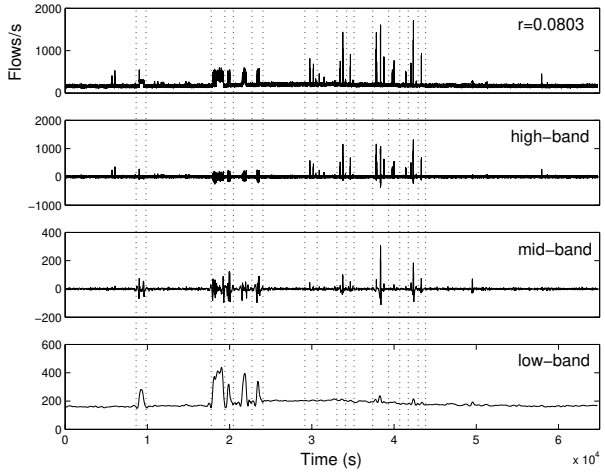


(b) sampling rate $p = 0.01$; anomalies = 6

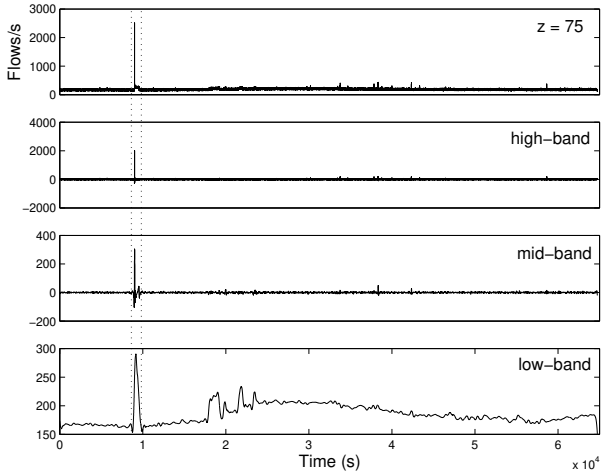


(c) sampling rate $p = 0.001$; anomalies = 1

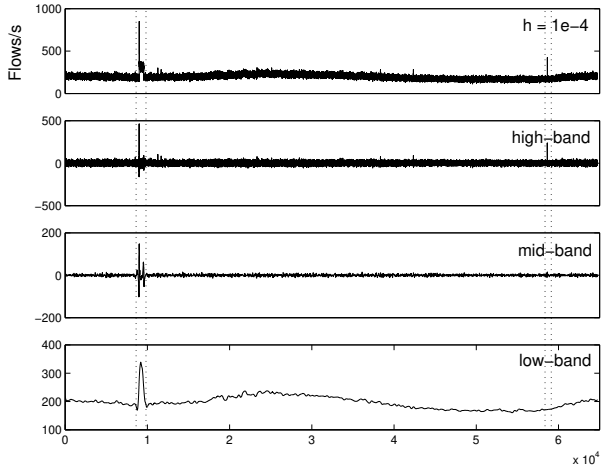
Figure 2: The *BB-East* Trace: Detection results from random packet sampling with sampling interval $N = 10, 100,$ and 1000.



(a) random flow sampling $r = 0.08$; anomalies = 18



(b) smart sampling $z = 75$; anomalies = 1



(c) sample-and-hold $h = 10^{-4}$; anomalies = 2

Figure 3: Detection results from random flow sampling, smart sampling, and sample-and-hold at effective sampling interval $N = 100$.

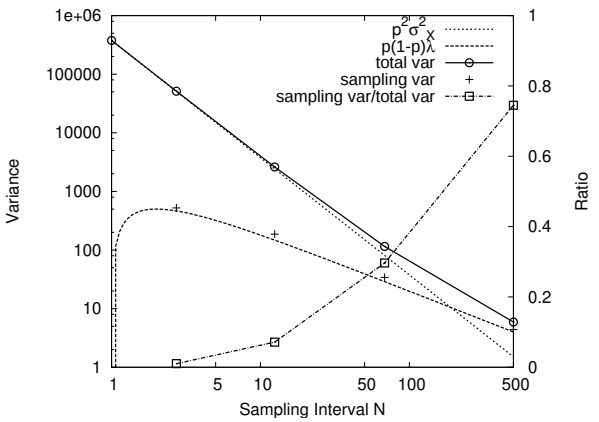


Figure 4: Total Variance, Sampling Variance and the percentage of Sampling Variance.

4. IMPACT OF SAMPLING ON PORTSCAN DETECTION

In this section, we evaluate the impact of the four different sampling schemes described in Section 2.1 on the efficacy of portscan detection. As mentioned earlier, TRWSYN and TAPS are chosen as examples of portscan detection algorithms. One of our goals is to empirically investigate how much (if any) do the various sampling schemes distort traffic fingerprints used by detection algorithms such as TRWSYN and TAPS, which rely on metrics other than traffic volume. We have previously explored how packet sampling affects the performance of TRWSYN and TAPS [17], and some of the results and discussions are included here to compare against three other sampling schemes: flow sampling, sample-and-hold, and smart sampling. We adopt the same methodology and metrics [17], which is described here for completion.

We performed the experiments using both *Trace BB-West* and *Trace Wireless*, which are known to contain portscan traffic (Section 2.4). Due to space limitations, we only include the results based on *Trace BB-West*, with the sampling parameters listed in Table 2. Similar results are observed in the case of *Trace Wireless*.

We use the following metrics previously defined by Sridharan *et al.* [19] to quantify the performance of the portscan detection algorithms:

$$\begin{aligned} \textit{Success Ratio:}R_s &= \frac{\#(\text{true scanners detected})}{\#(\text{true scanners})} \\ \textit{False Negative Ratio:}R_{f-} &= \frac{\#(\text{true scanner missed})}{\#(\text{true scanners})} \\ \textit{False Positive Ratio:}R_{f+} &= \frac{\#(\text{false scanner detected})}{\#(\text{true scanners})}. \end{aligned}$$

The success ratio R_s indicates the effectiveness of the detection algorithm, while the false positive ratio R_{f+} measures the relative error. It is desirable for an anomaly detection algorithm to have *high* R_s and *low* R_{f+}. we only need to discuss the success and false positive ratios because $R_s + R_{f-} = 1$. These metrics are generic to be used for evaluating different algorithms under various sampling schemes.

A challenging task in the computation of the above metrics, however, is to catch the true scanners. We use the final list of scanners manually generated by Sridharan *et al.* [19] as the ground truth. Even though the ground truth is an approximation of the actual set of scanners, it is sufficient for the purpose of our study since we are less interested in the *absolute* accuracy of these detection algorithms. The approximated ground truth allows us to study the *relative* performance of the portscan detection algorithms as a function of sampling scheme and sampling rates. Similar bootstrapping methods have also been used by other researchers [4, 14].

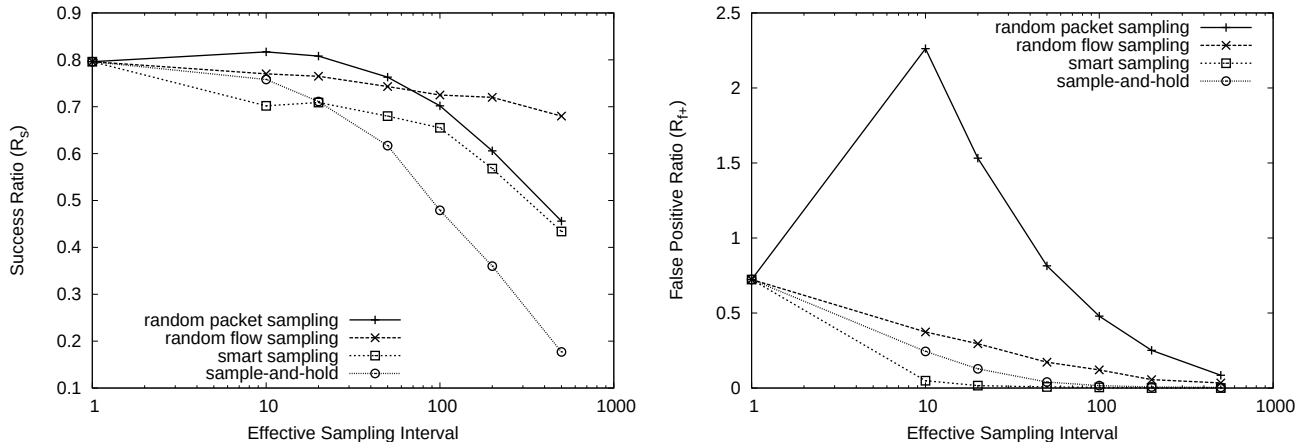
4.1 TRWSYN under Sampling

First, we discuss how TRWSYN performs when its input data is sampled using four different methods. In particular, we are interested in identifying the potential sources of inaccuracy in estimating traffic features that can affect its performance. Note that throughout our evaluation, we retain original values of the hypothesis test parameters, $\theta_0 = 0.8$ and $\theta_1 = 0.2$, which were used for TRWSYN on *non-sampled* traffic. This is because we wish to evaluate how different sampling schemes affect the performance of algorithms designed to work on non-sampled traffic.

Figure 5 plots the R_s and R_{f+} ratios for the *BB-West* trace as functions of effective sampling rates for all four sampling schemes. We first discuss random packet sampling as the base case for comparisons, since results with this particular scheme were previously obtained [17], where potential causes of distortion were presented and analyzed in detail.

Figure 5(a) indicates that the success ratio R_s of TRWSYN initially increases slightly for low sampling intervals before dropping off for larger values of N as the traffic is increasingly thinned. For example, R_s increases from 79.6% for the non-sampled trace to around 81% with $N = 10, 20$ before dropping off to 45.6% at $N = 500$. While this may seem advantageous, the false positive ratio R_{f+} also follows similar behavior but on a much larger scale. Specifically, the false positive ratio increases by a factor of 3 for low sampling interval of 10, indicating a large number of sources erroneously tagged as scanners, before dropping at large sampling intervals.

In our previous work [17], we have identified two key effects of packet sampling: *flow-reduction*, where the number of flows observed are reduced, and *flow-shortening*, wherein a multi-packet flow is reduced to a single packet flow by random packet sampling. Recall that the TRWSYN algorithm associates a single SYN packet flow with a failed connection attempt, which indicates a potential scanners. At small sampling intervals, the number of flows is not dramatically reduced and hence the algorithm can observe enough failed connections to retain the success ratio. However, it was shown [17] that even small sampling intervals can substantially shorten the flow size (in packets), thus increasing the number of single packet flows. This has a twofold impact. First, some ground truth scanners that may have transmitted multi-packet flows, but initially missed by TRWSYN, are now 'shortened' and hence 'detected', which explains the slight increase in the success ratio. On the other hand, flow shortening converts a large number of regular multi-packet flows into single SYN packet flows, that are now erroneously tagged as scanning traffic. This results in a dramatic increase in R_{f+}. When the sampling interval N increases, flow-reduction dominates, i.e., the number of flows observed from a single source falls off substantially. Consequently, the algorithm makes fewer decisions and hence both R_s and



(a) success ratio (b) false positive ratio

Figure 5: The *BB-West* Trace: TRWSYN detection results R_s and R_{f+} under different sampling intervals.

R_{f+} decrease.

We apply the same reasoning principles to the other three sampling schemes. Since random flow sampling, smart sampling and sample-and-hold are all effectively flow sampling methods, flow-shortening does not apply to the TRWSYN algorithm. This is because sampling decisions are based on the entire flow, not the individual packet. It is worth noting that sample-and-hold does hold an exception and a *mid-flow-shortening* does happen. Flows are only possibly shortened from a *mid-flow* packet to the end of the flow. If the first (SYN) packet is sampled, the entire flow will be kept. Since TRWSYN algorithm only makes decision on SYN packet flows, no new false positives or new detection can be introduced. Therefore flow-reduction is solely responsible for producing fewer samples with increased sampling interval and reduces. Evidence of our hypothesis can be seen in Figure 5, where both R_s and R_{f+} decrease almost monotonically for the three flow-based sampling schemes when the sampling interval N increases. More importantly, all three schemes result in extremely low false positives compared to packet sampling. As shown previously, packet sampling suffers from extensive flow shortening, generating high false positives in TRWSYN. This is not desirable from network administrative point of view. Hence, the flow-based sampling schemes clearly claim an advantage in producing low false positives over packet sampling.

Though all three remaining sampling schemes seem to affect portscan detection by TRWSYN in a similar manner, their relative impact on the performance degradation is quite different. Random flow sampling is the most robust in terms of R_s: it drops by only 15% from $N = 1$ to $N = 500$, and it outperforms other schemes when $N \geq 100$. The success ratio for sample-and-hold drops by more than 75% at $N = 500$, while the performance from smart sampling lies in the middle.

Interestingly, random packet sampling outperforms both smart sampling and sample-hold for all sampling rates in terms of the success ratio. As mentioned earlier, the latter two schemes favor large traffic flows and sample short flows at a lower sampling rate compared to large flows. Since most flows emitted by a scanner are single packet flows, they suffer more from flow-reduction in the same sampling interval as random flow sampling. However, since random flow sampling is not biased by flow size, it performs better than random packet sampling at larger sampling intervals where even packet sampling is biased towards large flows[1]. This is shown in Table 4.

Table 4: Percentage of single SYN-packet flows sampled from the *BB-West* Trace

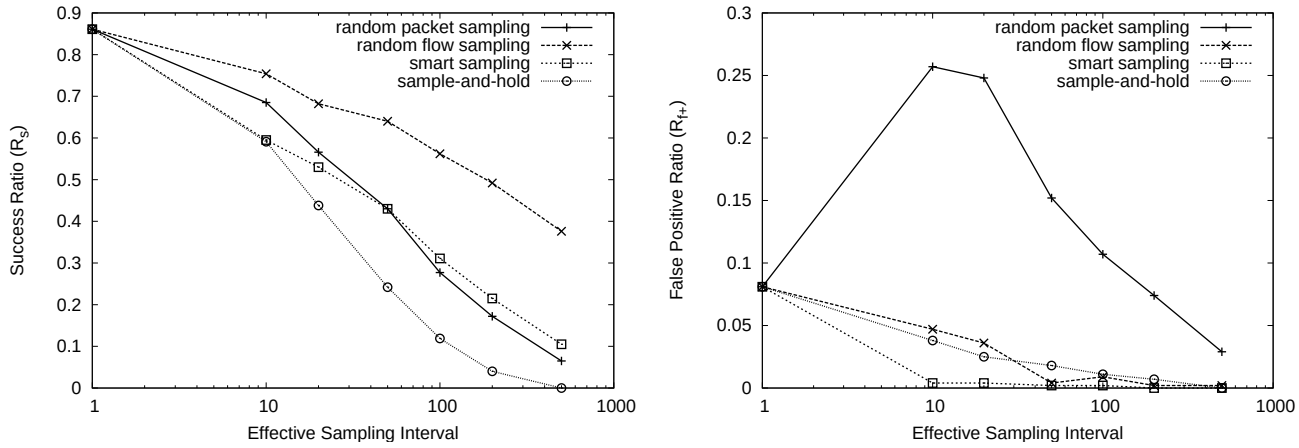
N	$\Pr^p$	$\Pr^r$	$\Pr^z$	$\Pr^h$
10	18.1%	34.4%	9.10%	6.98%
20	9.89%	22.3%	5.27%	3.06%
50	4.24%	11.7%	2.42%	0.987%
100	2.15%	6.90%	1.32%	0.414%
200	1.09%	3.88%	0.695%	0.186%
500	0.449%	1.72%	0.278%	0.0624%

4.2 TAPS under Samplings

We also evaluated the performance of TAPS when applied to traffic traces sampled using the four different schemes. Recall from Section 2.3 that time-bin is a critical parameter for TAPS. Hence, for each scheme, and for each sampling rate, we also varied the size of the time bin over a range of values in order to understand how R_s and R_{f+} changes as a function of the time bin and sampling interval. It was empirically shown [17] that for random packet sampling, there is an optimal time bin t for each sampling interval that maximizes R_s. Furthermore, the value of the optimal time bin size is an increasing function of the sampling interval. We found this to be true for the other three flow-sampling schemes as well.

Results of portscan detection with TAPS for the Trace BB-West are shown in Fig. 6 for all of the four sampling methods. For each scheme and each sampling rate, we show results using optimal time bin size that maximizes R_s. The behavior of TAPS under all four schemes is similar to that of TRWSYN. Specifically, Fig. 6(a) shows that the success ratio R_s of TAPS decreases for all four sampling schemes

[1]At small sampling intervals packet sampling leverages flow shortening to actually perform better.



(a) success ratio (b) false positive ratio

Figure 6: The *BB-West* Trace: TAPS detection results R_s and R_{f+} under different sampling intervals.

with random flow sampling performing the best and the most robust. Furthermore, random packet sampling performs nearly as well as the remaining two flow-based sampling schemes. Our reasoning for this behavior is similar to that for TRWSYN. Specifically, since smart sampling and sample-and-hold favor sampling of large flows, they miss out on small flows that are critical to identifying scanners.

The false positive ratio R_{f+}, shown in Fig. 6(b), initially increases in the case of packet sampling due to flow-shortening before dropping off at large sampling intervals due to flow reduction. Since flow shortening is not a major factor for the remaining schemes, all of them exhibit extremely low false positives that monotonically decrease with sampling interval. Although sample-and-hold does generate *mid-flow-shortening*, it does not produce a significant number of single packet flows to increase R_{f+} significantly. It was shown [17] that TAPS uses the address range distribution in its detection technique. As a result, TAPS with random packet sampling yields 1/10 of the R_{f+} observed by TRWSYN (a comparison of Fig. 5(b) and 6(b) confirms that). The same behavior is also seen with the other sampling schemes, in that they also yield far lower false positives under TAPS compared to that with TRWSYN. This is to be expected, since all of them are "insensitive" to the address range distribution and hence do not introduce any distortion in that aspect.

In summary, for both TAPS and TRWSYN portscan detection, flow-based sampling schemes do well in generating low false positives compared with random packet sampling. Random flow sampling overall is the most desirable scheme under large sampling intervals ($> 1/100$), since it performs well on both success ratio and low false positives. Under low sampling intervals, random packet sampling does the best in success detections and smart sampling does the best in low false positives. The main source of performance degradation for TRWSYN and TAPS in sampling environment is the inaccuracy in classifying single SYN-packet flows for the case of random packet sampling [17].

5. CONCLUSIONS AND FUTURE WORK

Various sampling techniques have been proposed for traffic measurements in high-speed backbone networks to reduce storage and processing overhead. There are well-studied trade-offs between the accuracy, efficiency, and scalability in choosing a specific sampling method or rate for traffic engineering purposes. However, the question of whether sampled data is sufficient for anomaly detection, which has become increasingly critical to network providers, remains an open question.

This work is among the first effort to measure and analyze the impact of sampling on traffic features critical for anomaly detections. Real packet traces collected from a Tier-1 backbone network were sampled using four popular methods: random packet sampling, random flow sampling, smart sampling, and sample-and-hold. The sampled data was then used as input to the following three representative anomaly detection algorithms: (a) a wavelet-based volume change detection, (b) TRWSYN for detecting TCP scanners based on inference of connection status, and (c) TAPS, which performs statistical hypothesis testing based on scanner access patterns of destination addresses/ports. Our results show that the sampling schemes degrade the performance of all three anomaly detection algorithms in terms of success detection and false positive ratio. All four sampling methods introduce varying degrees and forms of distortion, which we characterize based on the sampling technique, to traffic features such as traffic variance and flow size distributions.

5.1 Implications of our results

Through comprehensive study and analysis, we identify specific fundamental bias due to sampling that has significant implications on both classes of anomaly detection algorithms.

The Wavelet-based approach for *volume anomaly detection* utilizes a deviation score based on the relative difference between the local and global variance in the flow volume intensity at different time scales. Our evaluation indicates that both packet and the flow-based sampling schemes compress this difference mainly by inducing lossy data, thus weakening the ability of the algorithm to detect abrupt changes in volume. We expect to see similar effect on other volume-change detection schemes.

With respect to *portscan detection*, while all sampling schemes introduced degradation, they did so through different mechanisms. Random flow sampling, which performed the best, introduced degradation primarily through lossy but distortion free data. Packet sampling causes severe "shortening" of flows resulting in benign multi-packet flows getting tagged as scanner flows. Sample-and-hold and smart sampling are biased towards large flows and thus ignore small flows characteristic of scanners. We now discuss results for each sampling scheme in more detail below :

- Random flow sampling has been proposed to improve the accuracy in estimating flow statistics. We show that it is also performs better for both volume anomaly and portscan detections compared to other sampling methods. It introduced the least amount of distortion in the global variance of the flow arrival time series, flow size distribution as well as access patterns of IP sources. This can be attributed to the fact that it is neither biased towards particular flow sizes (like sample-and-hold and smart sampling), nor does it cause flow size degradation like packet sampling. Both these properties are highly desirable for anomaly detection. However, this performance comes at the expense of very demanding and hence prohibitive resource requirements.
- As expected random packet sampling causes significant deterioration in performance of both the volume anomaly and portscan detection algorithms. The volume anomaly detection algorithm is affected by lossy data caused due to flow omissions thus resulting in an increase in false negatives, *i.e.,* a loss of detectability. *Port scan detection* however, suffers from both lossy data *and erroneous data*, the latter caused by the well-known flow shortening due to packet sampling. This not only increases the false negative rate but also induces false positives. This particular result was also previously observed by the authors [17].
- Smart sampling and sample-and-hold are designed for accurate estimation of heavy-hitters with reduced complexity. They are not meant for anomaly detection and not suitable for the job either. Both methods degrade volume anomaly and portscan detection dramatically since many attacks of both classes usually comprise of small-sized flows, which are ignored by these biased sampling methods. Interestingly, this bias actually causes them to perform *poorer* than packet sampling in terms of volume anomaly detection.

5.2 Future Work

We believe that the lessons learned in this paper can be leveraged to address these accuracy and efficiency tradeoffs in designing better sampling techniques. Anomaly detection operates on a significantly different information region, which is often overlooked by existing traffic accounting methods that target heavy-hitters. Better measurement techniques need to adapt to the needs of anomaly detections as well as traffic engineering, whether it be flow statistics or access patterns. On the other hand, anomaly detection algorithms can be improved under sampling if the information loss and distortions is compensated or better avoided. Another relevant open question is whether correlating sampled traces from multiple vintage points could improve the anomaly detection process at relatively low sampling rates, hence avoiding the need for detailed packet trace collection.

6. REFERENCES

[1] Cisco IOS Software NetFlow. http://www.cisco.com/warp/public/732/Tech/nmp/netflow/.

[2] Juniper Networks: JUNOS 7.2 Software Documentation. http://www.juniper.net/techpubs/software/junos/junos72/index.html.

[3] Snort. http://www.snort.org.

[4] P. Barford, J. Kline, D. Plonka, and A. Ron. A Signal Analysis of Network Traffic Anomalies. In *Proc. ACM SIGCOMM IMW'02*, pages 71–82, Marseille, France, Nov. 2002.

[5] P. Barford and D. Plonka. Characteristics of Network TRaffic Flow Anomalies. In *Proc. ACM SIGCOMM IMW'01*, pages 69–73, San Francisco, CA, USA, Nov. 2001.

[6] B.-Y. Choi, J. Park, and Z.-L. Zhang. Adaptive Random Sampling for Traffic Load Measurement. In *Proc. IEEE International Conference on Communications (ICC'03)*, Anchorage, Alaska, USA, May 2003.

[7] N. Duffield. Sampling for Passive Internet Measurement: A Review. *Statistical Science*, 19(3):472–498, 2004.

[8] N. Duffield, C. Lund, and M. Thorup. Properties and Prediction of Flow Statistics from Sampled Packet Streams. In *Proc. ACM SIGCOMM IMW'02*, Marseille, France, Nov. 2002.

[9] N. G. Duffield, C. Lund, and M. Thorup. Estimating Flow Distributions from Sampled Flow Statistics. In *Proc. ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.

[10] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a Better NetFlow. In *Proc. of SIGCOMM'04*, Portland, Oregon, USA, Aug. 2004.

[11] C. Estan and G. Varghese. New Directions in Traffic Measurement and Accounting. In *Proc. of SIGCOMM'02*, Pittsburgh, Pennsylvania, USA, Aug. 2002.

[12] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-level traffic measurements from the sprint IP backbone. *IEEE Network*, 2003.

[13] N. Hohn and D. Veitch. Inverting Sampled Traffic. In *Proc. ACM SIGCOMM IMC'03*, Miami Beach, Florida, USA, Oct. 2003.

[14] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proc. of 2004 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2004.

[15] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *Proc. ACM SIGCOMM IMC'03*, Miami Beach, Florida, USA, Oct. 2003.

[16] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. In

Proc. ACM SIGCOMM '05, Philadelphia, PA, USA, Aug. 2005.

[17] J. Mai, A. Sridharan, C.-N. Chuah, T. Ye, and H. Zang. Impact of Packet Sampling on Portscan Detection. Technical Report RR06-ATL-043166, Sprint ATL, 2006. (*accepted by IEEE JSAC Special Issue on Sampling the Internet: Techniques and Applications*).

[18] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proc. 1999 USENIX LISA Conference*, Seattle, WA, USA, Nov. 1999.

[19] A. Sridharan, T. Ye, and S. Bhattacharyya. Connection Port Scan Detection on the Backbone. In *Malware Workshop held in conjunction with IPCC*, Phoenix, Arizona, USA, April 2006.

[20] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. *J. of Computer Security*, 10(1-2):105–136, 2002.

[21] M. Thottan and C. Ji. Anomaly Detection in IP Networks. *IEEE Trans. on Signal Processing*, 51(8):2191–2204, Aug. 2003.