Technical Report
Packet Sampling for Network Monitoring

Ryszard Erazm Jurga, Miłosz Marian Hulbój
{rjurga,mhulboj}@cern.ch

CERN — HP Procurve openlab project
CH-1211, Genèva 23, Switzerland
http://cern.ch/openlab

10th December, 2007

Abstract

In this paper, the emphasis is placed on various packet sampling methods and their application to network monitoring. We present the motivations for packet sampling on high-speed network links. We give an overview of all known packet sampling techniques, point out their strengths and weaknesses in terms of reliability and accuracy in the estimation in different network traffic parameters. We also provide more details about the adaptive packet sampling, as a way to increase this accuracy.

One of the key objectives of our study is to gain some insight about the feasibility of the packet sampling in context of network anomaly detection. It was proven that packet sampling provides reliable estimates in traffic monitoring applications. However there is not much research on whether packet sampling provides sufficient amount of information for the anomaly detection.

1 Introduction

Packet monitoring is an integral part of network measurements. In packet monitoring one is interested with the information contained within packets traversing a link. Initially packet monitoring was performed by specially designed probes installed on the network [1, 2, 3, 4, 5]. These probes copied the entire contents of every packet for further analysis. This approach allowed relatively easy monitoring of whole network traffic on shared medium networks like a FDDI ring or the early Ethernet. However processing the entire contents of every packet imposed high demands on packet probes and their resources. For this reason probes usually captured only the initial part of the packet which contains valuable information – the protocol headers and initial contents of the application data. Document [6] explains why packet protocols headers are of central of interest in network measurements. Even so, this became infeasible with the advent of high-speed switched networks when the bandwidth and the volume of data increased [7, 8]. The next approach to packet monitoring was to embed the probe functionality within the network equipment.

Switches and routers have limited computing power and resources. Their main purpose is definitely not packet monitoring (as opposed to a monitoring probe). These arguments were a driving force for another way of data reduction. Embedded packet monitors might reduce the volume of data by filtering, aggregation and sampling [9]. Amer and Cassel in [7] give more reasons for choosing packet sampling.

Quite recently yet another trend in network monitoring had appeared. Thanks to the progress in massively parallel processing within specialized FPGA circuits, the Deep Packet Inspection attracted the researchers and hardware manufacturers. It is further discussed in section 3.3.

In this report we would like to address packets sampling methods and their application in network measurements.

2 Typical features of the network traffic

Recent studies of traffic measurements have shown that network traffic possesses the property of long-range dependence [10, 11, 12]. Such property manifests itself in the correlation structure where the autocorrelation functions obey some power-law. Network traffic exhibits burstiness at a wide range of time scales and could be described as self-similar. Self-similarity is a term from fractal theory which states that the object appears the same regardless of the scale. Publication [10] was a breakthrough in modeling of the network traffic, as previously used Poisson models assumed that burst length would tend to be smoothed by averaging over a long time-scale. The fact that network traffic exhibits long-range dependency have severe consequences both on network traffic modeling (prediction) [13, 14] and sampling process. Uniform sampling from heavy-tailed distributions may yield poor estimates since a relatively small amount of samples can have severe impact on the resulting estimates.

3 Packet selection techniques

3.1 Packet sampling

As it is pointed out in [6], the packet sampling techniques were categorized initially in [7]. They are currently being standardized by the Packet Sampling (PSAMP) Working Group of the Internet Engineering Task Forces. We give an overview of all known packet sampling techniques, point out their strengths and weaknesses in terms of reliability and accuracy of the estimation of different network traffic parameters. We also provide more details about the adaptive packet sampling, as a way to increase the accuracy.

3.1.1 Systematic sampling

Systematic packet sampling involves the selection of packets according to a deterministic function. There are two ways to trigger the selection: count-based with the periodic selection of every k-th packet or time-based driven, where a packet is selected every constant time interval. It is pointed out in [15] that the first approach gives more accurate results in terms of the estimation of traffic parameters. Systematic sampling is straightforward to implement, i.e., the original Sampled NetFlow [16], Juniper Packet Sampling [17]. However, it is vulnerable to bias if the metric being measured itself exhibits a period which is rationally related to the sampling interval [18]. Systematic sampling can be easily anticipated, hence open to susceptible to manipulation [19, 20]. In some cases systematic sampling can perform better that random sampling, see [21, 22]. In case of sampling self-similar process, the systematic sampling provides lower sampling method average variance than in case of simple random sampling [23, 24] (if variance of sampling results is high, then one cannot rely on single sampling instance to infer the entire process).

3.1.2 Random Sampling

In random packet sampling the selection of packets is triggered in accordance to a random process. The unbiased estimation can be achieved, since each selection is an independent experiment.

Random additive sampling The potential problems of systematic sampling are avoided by using independent, randomly generated triggers in order to select packets. These triggers have a common statistical distribution. It avoids synchronization and predictability problem [25]. Duffield in [26] suggests choosing the triggers to be geometrically distributed (for count based sampling) or exponentially distributed (for time-based sampling). See [20] for more details how to generate sampling intervals using Poisson distribution. Zhang [27] shows that count-based Poisson sampling has better performance than time-based Poisson Sampling [28].

Simple Random Sampling In this technique n samples are selected out of N packets, hence it is sometimes called n-out-of-N sampling. For this sampling schema each packet has an equal chance of being drawn. One way of achieving a simple random sample is to randomly generate n different numbers in the range of 1 to N and then choose all packets with a packet position equal to one of these n numbers. This procedure is repeated for every N packets. For this kind of sampling the sample size is fixed. Examples of implementation include sFlow [29] and Random Sampled Netflow [30]. Both examples use 1-out-of-N sampling

Probabilistic Sampling In probabilistic sampling samples are chosen in accordance to a pre-defined selection probability. The sample size can be different for consecutive intervals. For uniform probabilistic sampling each packet is selected independently with a fixed probability p. When a probability p depends on the input (i.e. packet content) then this is non-uniform probabilistic sampling. This non-uniform approach can be used to weight sampling probabilities in order to boost the chance of sampling packets that are rare but are deemed important. More details about non-uniform sampling can be found in [31] as well as in (4.2).

Adaptive sampling

Motivation The sampling rate translates (directly or indirectly) into the accuracy of estimation procedure [32, 33, 21, 34]. Some network behavior may not be accurately detected with low sampling rates. This particularly refers to anomaly detection. On the other hand, sampling with high sampling rate produces vast amounts of data that needs to be sent to the collector and processed afterwards. In the periods of high traffic the network equipment might not cope with the required sampling rate and drop the excessive packets. The increased number of samples can have influence on the overall traffic. As in most cases sampled packets are being sent via connectionless datagram protocol, it is important to avoid congestion situations. Thus it is obvious that there is a trade-off between accuracy and the broadly defined performance. What is not obvious is how to choose the right sampling rate. It might be a challenging or an impossible task. The network traffic itself exhibits variability in a number of packets traversing links for different time periods. What is distinctive about the network behavior are the sudden bursts of the traffic (because the network traffic can be characterized by heavy-tailed distributions).

Network traffic does not exhibit stationary behavior. It exhibits temporal cycles (daily, weekly) and can be easily affected by network reconfigurations (e.g. manual reconfiguration, dynamic routing change), link failures and deployment of new machines and applications. Although the packet sampling is a lightweight process that does not consume much of the network device resources (memory and CPU) there are certain limitations with respect to the maximum number of samples the device can produce within a given time frame. As mentioned earlier, in periods of high traffic the device may become saturated. This could cause a situation in which the device produces fewer samples in the busy periods than in case of 'normal' traffic. This is unfortunate as sometimes the packets appearing during the high load are the most important ones to capture. And even if there were no limits in hardware, the excessive sampling traffic could be interpreted as an anomaly or make the already existing anomaly worse.

The issues mentioned beforehand (estimation error and network device resources) are being addressed by a broad group of adaptive sampling techniques. These methods employ either a special heuristic for performing the sample process or certain prediction mechanisms for predicting future traffic and adjusting the sampling rate.

Sampling rate adaptation has some potential drawbacks. There is some latency in the adaptation process and in case of unanticipated traffic burst the saturation of the device will be possible. In order to avoid it one would have to allow a certain safety margin by employing systematic undersampling.

Adaptive sampling mechanisms One group of solutions focuses on implementing the custom sampling methods. [35] describes two methods of adaptive sampling for managing the processor usage in the network device. One method uses the information about the current processor utilization in order to adjust the packet selection rate. The other utilizes the packet interarrival times (which can be used to anticipate incoming burst of traffic) in conjunction with the knowledge about the processing time required for processing a sample. The study found that proposed adaptive methods yielded more accurate estimates of network traffic parameters (Hurst Parameter used for describing self-similar phenomena) under predefined resource constraint than simple static sampling. In [36] the application fast control loop for adjusting the sampling interval had been tested. Linear prediction and fuzzy logic approach are being discussed with application to different types of traffic (normal Internet traffic and bursty video traffic). Authors had performed offline simulations and did not analyze the feasibility of embedded implementation in the device. [37] proposes to use the weighted least squares predictor and certain set of heuristic rules for determining the sampling rate. It does not provide the complexity analysis of the algorithm which requires more numerical operations than simple linear prediction. [38] describes an approach to flow sampling that allows to control the expected volume of samples and to minimize the variance of usage estimates arising from the sampling. The proposed schema (smart sampling) adapts the sampling process by associating the probability that flow is selected with the size of the flow. This process shifts focus towards the larger 'elephant' flows which have a severe influence on the traffic volume. These ideas are further extended in [39] to work under strict resource constraints by sampling into a buffer of fixed size. Reservoir sampling [40] is an example of rate constrained sampling (approach that can select specified number of objects in a specified time frame). Reservoir is a special buffer holding a predefined number of current samples. As more samples are being processed the contents of the reservoir may be replaced. The process is done in such way that at each time instance the contents of the reservoir represent a true random sample.

However these methods require that the special sampling mechanism is present in the network devices. Vendors do not want to implement sophisticated sampling schemes that give good results under certain circumstances. They want to implement simple and robust solutions that are well described by some form of a standard (i.e. sFlow, NetFlow). Thus special sampling mechanisms remain currently in the area of academic research and are being tested either offline or with custom made probes.

Adapting the sampling rate The network devices available on the market usually are capable only of performing a certain type of sampling. There is no possibility of changing the algorithm and the user can only adjust the sampling rate. Despite this limitation it is still possible to realize the closed control loop for adapting the sampling rate to the current network traffic load. The dynamics of these control mechanisms is much slower than for adaptation within the device – as there is additional overhead (time to receive data from the device and time to send new settings to the device). It is possible to perform the control in the tens of seconds scale. Due to slower intervention rate and more computational power it is possible to employ more complex prediction algorithms than in case of embedded solutions. Still, the performance is important, as the management machine will likely manage many devices with multiple ports.

There are various approaches to traffic rate prediction and adaptation which differ vastly both in accuracy and complexity.

One of the simplest solutions is the 'naïve' prediction, in which one assumes that the predicted number of packets for the next time interval would be equal to the number of packets for the current time frame. It requires virtually no calculations, however the accuracy of estimations is lacking. Most of the prediction methods that have been mentioned in (3.1.2) can also be used in this case. However access to some internal network device data is not possible (like state of the queues, packet interarrival rate, real-time resource utilization, etc). The main information that can be used for predicting future traffic is the previous packet counts and previous packet volume.

The fact that network traffic exhibits self-similarity [2] had raised many disputes on how this information should be exploited in traffic prediction. [14] compares predictors based on short-term correlations and examines whether including long-range dependence is beneficial. The conclusion is that primarily the short-term correlations dominate the performance of the predictor. As a consequence linear prediction with relatively short correlation structure is sufficient for prediction applications.

Another comparison of linear prediction method (based on linear minimum mean square error) with methods exploiting the long-range dependent characteristics (Fractional Brownian Motion, Fractional ARIMA[1]) is described in [13]. Linear predictor provides sufficient accuracy when compared to other methods.

[41, 42] describe the specific structure of neural networks that is especially suited for handling time series data. The memory of a recurrent neural network does not have a definitive temporal limitation [41]. Past inputs are being processed and few characteristics are stored with the aid of feedback loops. Recurrent neural networks have outperformed other prediction mechanisms in short-term traffic prediction [41, 42]. However the complexity of such neural networks is very high and the solution may not scale very well.

The application of adaptive random sampling had been extensively studied in [33, 21]. These technical reports describe the means of obtaining constant level of error of flow volume estimation. Linear prediction method based on AR[2] model utilizes the information both about the packet count and volume. Detailed mathematical analysis follows. In addition to that a method for load change detection is being described.

Packet sampling and self-similarity None of previously mentioned methods take into account the self-similar nature of the network traffic [23, 24]. For a stationary process with finite mean and variance both the systematic sampling (3.1.1) and simple random sampling (3.1.2) provide an unbiased estimate of the real mean. However in case of self-similar process the variance is infinite and sampled mean approaches the real mean slowly. This is because a small fraction of self-similar process accounts for the majority of traffic volume. Extremely high sampling rates are required to capture this influence. [23, 24] proves that static systematic sampling, stratified random sampling and simple random sampling provide accurate estimates of the Hurst parameter (second order statistics for self-similar process), yet they all fail to provide estimate of real mean (first order statistics). Authors modified the systematic sampling method by introducing a bias which allows to capture the extremely large values more accurately. The new schema outperforms the other mentioned methods in accuracy of mean estimation while maintaining comparable level of error when estimating the Hurst parameter.

3.2 Packet filtering

Filtering is the deterministic selection of packets based on their content. The packet is selected if its content matches the specified mask. The selection decision is not biased by the packet position in the packet stream. This approach requires the packet content inspection, since packets can have different formats and a fixed length mask can not be applied. For encrypted packets the paper [31] proposes to

[1]Autoregressive Integrated Moving Average

[2]Autoregressive

use the router state [31] to trigger the selection process instead of the packet content. Although filter based selection decisions are deterministic, they can approximate random sampling. Guang in [43] gives three factors which should be satisfied for assuring statistical randomicity of sampled packets. The masking bits should be invariant to ensure the consistency of sample, the sampling ratio should be independent of packet content and the masking bits should have high randomicity. He analyzed entropy of IP header bits and found high randomicity of bits in identification field. He can control a sampling rate by changing the mask length. He in [44] expands his algorithm by multi-mask sampling model, the threshold sampling model and the modulus sampling model in order to facilitate a control over sampling parameters in distributed environment. The packet identification field it can be also used in trajectory sampling. Duffield in [45, 46, 47] provides more details how to reconstruct packet trajectories.

Instead of a simple mask, hash function can be used to trigger the packet selection. In order to assure statistical randomicity of sampled packets, the hash function must have good mixing properties. More detailed discussion about a good choice of hash function as well as security vulnerabilities is in [31, 19]

3.3 Deep Packet Inspection

Deep Packet Inspection (DPI) is an interesting technology that can become useful for detecting malicious attacks. The most important parts of DPI are regular expression matching and signature based scanning. In this technique the payload of all the packets is checked against the set of known malicious signatures at the wire speed. Snort can be considered as a DPI software, but it is not able to cope with the high-speed traffic. This is mainly due to the limitations of the Von Neumann sequential architecture [48] and also due to poor optimization of the regular expressions used for matching [49, 50].

In case of deep packet inspection it is often necessary to match the patterns at every byte offset. Most likely many signatures will have to be matched against packet payload. Thus the process requires a substantial number of comparison operations. The sequential processing is not suited to this mode operation and for this reason custom parallel approaches are being employed.

Publication [51] lists the functionality that a DPI implementation should provide. First, it should be able to provide at least the pattern index number and the information about the location (some rule checking software could then perform more detailed analysis). It should also support the grouping of patterns and a group of patterns should only be tested if the packet belongs to that class. [49] suggests that a more efficient and accurate DPI system can be built by using the header classifier running in conjunction with the payload matcher. Header processing is much simpler, as header location within the packet is predefined. Another important issue is related with worst-case performance which should remain constant and invariant of the amount of known signatures within the analyzed traffic. The final requirement is to allow fast, easy and non-interrupting update of the pattern database.

As the sequential architecture is not well suited for realization of the DPI task, the researchers focused on developing parallel FPGA implementations. Usually one of three algorithms is used for efficient multi-pattern matching: (1) Bloom Filter algorithm, (2) Aho-Corasick (AC) algorithm [52], (3) Boyer-Moore Algorithm [53]. By far the most common one is the Bloom Filter [54, 55, 56] and its extensions. It utilizes multiple hash functions and can yield false positives (but never a false negative). Computation time involved in performing the query is independent of the number of patterns in the database. Results obtained both with Bloom Filters and other algorithms [53, 51] are interesting, as it was possible to obtain pattern matching against numerous rules at the speed of modern Ethernet: [51] – 10Gbps, [53] – 1Gbps, [55] – 2.4Gbps, [52] – 1.6Gbps. The solutions vary in price, but the cost and power consumption of an FPGA chip is not prohibitively high.

3.4 Hybrid techniques

Hybrid techniques are realized by combining a few packet selection approaches. For instance, Schöller in [57] proposes to add packet sampling into the packet filter in NodeOS (an operating system for Active Network nodes).

Another example is Stratified Random Sampling. In this approach packets are grouped into subsets according to the given characteristic. Then number of samples are drawn randomly from each group. Stratified random sampling gives smaller variance of single packet statistics than simple random sampling if the variance within group is small compared to the variance between groups [58, 21]. Zseby in [59] shows how stratified sampling beats systematic and random sampling when it is applied for SLA[3] validation. She in [58] also investigates criteria for packet grouping and shows that with all cases the number of sampled packets can be reduced. Another stratification technique based on packet count is proposed in [15].

4 Packet sampling applications

Packet sampling serves as a basis for a wide range of network monitoring, management and engineering tasks. It gives a dynamic insight into network by providing packet headers. This detailed information can be grouped and used to build different network traffic estimations, statistics, distributions, time series and aggregates. Examples include distributions of packet counts, packet sizes, packet interarrival times and protocols, traffic flows. Depending on the application as well as on algorithms used for further analysis, these metrics can be group together on the basis of different criteria in order to form some clusters with data significant to detect particular network problems. The accuracy of packet sampling depends on sampling rate, application and periodicity of measured metric. Periodic events are more likely to be detected by packet sampling.

There is a lot practical applications where packet sampling plays significant role and can provide accurate results while reducing the monitoring costs. It can be used for troubleshooting network problems, like controlling congestion, detection of broken links, misconfigured devices and rouge network servers. Packet sampling helps to verify quality of service in network, this for instance includes validation of SLA, point-to-point delay measurements, i.e. [60]. Sampling techniques can be used to meter and bill for network usage and provide users with an itemized breakdown highlighting top users and applications. Also packet sampling can be used to build trends and forecast bandwidth and other resources requirements as well as to profile network routes.

All these mentioned applications can profit from packet sampling and still preserve a decent accuracy. However there are domains where packet sampling might introduce some significant errors. There are a lot of packet sampling applications in the domain of network security and anomaly detection. Examples include detections of protocol violations, user policy violations, unauthorized access to an address from a blacklist. For this kind of anomaly sometimes only one packet is enough to raise the true alarm. For other well know network anomalies, violations and attacks, like DoS attack, P2P applications, portscans more sophisticated methods are needed in order to provide accurate results. These advanced methods include for instance a flow analysis. This is a relatively new topic in the area of anomaly detection and more studies have to be done in order to better understand how packet sampling can meet the accuracy and not to raise false alarms. In next sections we focus with more details on flow estimations as well as security applications.

4.1 Traffic flows

The idea of flow is one of the more important concepts in network management and monitoring. There are numerous definitions of flow, however there is one common denominator for all of them. A flow of

[3]Service Level Agreement

traffic is a set of packets sharing some common property (known as the *flow key* or *flow specification*) and having some temporal locality as analyzed within at a given measurement point.

The detailed definition of flow and the selection of the key depends on the further usage of flow data. Typical examples of flow keys are: source/destination MAC addresses, source/destination IP addresses, port numbers, protocols type, or combinations of these elements. Many researchers [33, 21, 61, 62, 63] use 5-tuple flow composed of source/destination IP address, port numbers and protocol type. Flow record can be thought of as an abstraction describing a set of packets related with a certain activity on the network. The set of packets included in the flow depends solely on the algorithm for assigning packets to flows.

Using flow records instead of packet headers yield a considerable compression of information, as flow is summarized by a fixed size record invariant on the number of underlying packet samples. The obvious trade-off is loss of the details contained within the packets and interpacket relationships.

Many packet sampling methods do not preserve the characteristic features of traffic flows [6, 64]. It is easy to miss the short flow (many flows can be only few packets long). It might be difficult to estimate the length of the flow (due to the fact, that only few packets belonging to the flow are sampled). Flows may be mis-ranked (smaller flow may appear larger in the sampled statistics) and large flows may be split into smaller ones. Naïve approach can thus yield considerable errors. Section 4.2 describes some methods for better flow estimates.

4.2 Identification of mice and elephant flows

Flow statistics, similarly as the network traffic itself, exhibit strong heavy-tailed behavior in various networks [65]. Many observations show that small percentage of flows accounts for a large percentage of the traffic. For many traffic measurements, detecting and measuring these large flows ('elephants') is an important element. Examples include active queue management [66], billing schemes [67], QoS mechanisms [68], decisions about network upgrades and peering [69]. If flows are built from traffic sampled at the packet level, the detection accuracy is strongly dependent on the sampling rate. Bakarat in [70] shows that for some scenarios (i.e., the top 10 flows) a sampling rate higher than 10% is required. However if one is only interested in an unsorted list of the largest flows, sampling requirements decrease by an order of magnitude.

Many researchers have addressed these problems and in this subsection we present a short summary of various approaches.

Estan in [65] proposes sample-and-hold sampling and multistage filters. The idea is to sample each packet with a certain probability. If a sampled packet and its flow has no entry in the flow memory, a new entry is being created. However, after an entry is created for a flow, the flow entry is updated for every subsequent packet belonging to the flow, as in the approach used in [71]. This sampling method involves processing of each packet arrival. Manku in [72] uses a similar approach – sticky sampling, however the packet sampling probability is being changed during the measurement period. [64] analyzes several methods of sample-and-hold sampling and its influence on flow inversion.

Duffield in [67] proposes size-dependent sampling. He defines a size threshold. Flows of byte size greater than the threshold are sampled with probability 1 and the others are sample with the probability proportional to their size. He develops this sampling theory in [73, 38, 74, 39] using NetFlow records.

Mori in [75] on the basis of Bayes theorem develops techniques and schemes to identify elephant flows in periodically sampled packets. The key is to find the threshold of per-flow packets in sampled packets which can reliably indicate whether or not a flow is actually an elephant flow in unsampled packets. The prior knowledge of distribution of flow size is required in order to calculate this threshold. He proposes three possible approaches to obtain the threshold. Using the power-law characteristic of Internet traffic he concludes that the threshold obtained for one network can be used as an approximation for other networks as long as their flow statistics also exhibit heavy-tail characteristics.

Kodialam in [76] proposes a novel approach based on sampling two-runs to estimate per-flow traffic. A flow has a two-run when two back-to-back samples belong to the same flow. Two-run detecting registers contains the flow id of the last sample and this id is compared against next incoming packets. This scheme automatically biases the samples towards the larger flows, because they have a larger chance of forming two-run. This makes the estimation of large flows more accurate.

In many traffic engineering applications researchers focus on minimizing the number of needed samples for the estimates. This is can be achieved by smart sampling [38] or proper selection of network devices that performs the sampling [77]. This approach might be well suited for traffic monitoring, however it does not seem appropriate for intrusion and anomaly detection.

4.3 Packet sampling for security

Network anomalies such as failures, attacks, worms, port scans, etc are common in today's computer network. These threats cause perturbations of the normal network behavior. Tracking down the sudden and unexpected (abrupt) changes [78] of network traffic is crucial to the network reliability. Detection techniques rely on the analysis of network traffic and the characterization of the dynamic statistical properties of traffic 'normality'. So far, packet sampling techniques have been mainly used for network traffic accounting and billing. Many researches addressed this problem and they proved that the desired accuracy of various traffic estimations can be achieved using only partial traffic data. These results could be useful for techniques which use only estimated thresholds to detect anomalies.

There is a wide range of common network anomalies that only require a single sample in order to provide 100% accuracy of detection [79], for instance protocol violations, a connection to an IP address from a blacklist, NAT devices [80]. If a single packet sample contains a signature that identifies one of these anomalies, then it is likely that the packet was crafted with malicious intent. A single packet can be also sufficient for host and operating system identification [80]. The passive OS fingerprinting tool (`p0f`) is widely used both for such identification and masquerade detection [81].

There are network anomalies which cannot be detected by observing only a single sample. An example is an anomaly which misuses protocol for purposes which were not meant for the protocol. This requires more advance techniques than a simple signature check. For instance DNS protocol can also transport user traffic which has nothing to do with a DNS service [82]. In order to detect these anomalies, one has to perform statistical anomaly detection on the network. We need more samples in order to estimate a threshold between normal and abnormal packets. In order to keep a reasonably accuracy of these thresholds, we need a minimum number of these samples [32]. Packets which exceed the threshold could be considered as an evidence of anomaly. For instance, a DNS tunnel could be detected by observing that the total amount of data transferred over port 53 is much higher than usual. More advanced detection method tries to detect a significant difference in the format entropy of these tunnel lookups as compared to the regular ones. Another example where thresholds can be used is detection of anomaly where one host communicates with an unusual number of destinations in a specified time window. It is important to determine proper 'normal' profile for different types of hosts (user machine, printer, file server, application server, etc).

Other network anomalies to be detected require more sophisticated analysis based on the observations of time series of network parameters, traffic patterns, trends and flows. When the network traffic data is sampled the detection of anomalies becomes more complex. An impact of packet sampling on more complex anomaly detection processes has not been studied that intensively as compared to statistical estimation of traffic parameters. Those packet sampling techniques that were succeed in the traffic accounting and billing might not be necessarily appropriate choices for the accurate and effective anomaly detection. We present below our findings in this domain.

Androulidakis in [83] investigates the impact of various packet sampling techniques on two anomaly detection techniques – a sequential non-parametric-change-point detection method(CPD) [84, 85] and an algorithm based on Principal Component Analysis (PCA) [86]. He uses different metrics under different traffic and measurement sampling methodologies. A distributed Denial of Service attack (TCP SYN attack) against a single host campus is studied in details. For CPD the difference between SYN and FIN packets is observed as well as the ratio of new source IP addresses over the frequent source IP addresses that are observed in a specific time bin. For PCA-based approach he used seven metrics, as follows: number of flows, packets, TCP flows, TCP packets, short flows (with a small number of packets), SYN packets, and FIN packets. The impact of systematic, random n-out-of-N and uniform probabilistic sampling on the anomaly detection techniques is evaluated. Results revealed that systematic sampling does not perform well under low sampling rates when the detection of the attack depends on certain packet characteristics (e.g. TCP flags). The experimental results of the PCA-based method show, that the efficiency of the method is independent of the sampling method used and relies only on the sampling rate, for both packet-based and flow-based metrics.

Mai in [87] presents the impact of flow sampling techniques: random packet sampling (the sampled traffic is classified into flows), random flow sampling, smart sampling, and sample-and-hold on the volume anomaly detection and the port scan detection. He focuses on a discrete wavelet transform (DWT) based detection procedure [88] used against DoS attacks. Threshold Random Walk [89] and Time Access Pattern Scheme (TAPS) [90] is evaluated against portscan. Both packet and the flow-based sampling schemes degrade volume anomaly and portscan detection. However random flow sampling turns out to perform better for all detection methods. 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. 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. Random packet sampling used to build up flows causes significant deterioration in performance of both the volume anomaly and portscan detection algorithms. On the other hand, techniques such as TAPS that exploits access pattern of scanners, e.g., spread of destination IP addresses or ports are more robust with respect to sampling [91]

Kawahara in [92] also shows that packet sampling degrades detection of network anomalies. For example, portscan and SYN Flooding generates a number of small flows consisting of single packets. Such flows are more unlikely to be sampled than normal flow. As a solution to this problem, he shows that we can increase the detectability of such anomalies by spatially partitioning the monitored traffic into groups and analyzing the traffic of individual groups. He partitions the traffic into groups according to the source IP addresses of individual flows composing the traffic. This approach is also presented in [93] where Ishibashi divides network traffic according to hashed IP addresses and protocols.

Performance of the anomaly detection can be improved by using the appropriate metrics, such as entropy-based summary [94, 95]. This works well for the worm detection [96], unfortunately not for the portscan detection [91]. Worms expose periodic behavior and packets belonging to them are more likely to be sampled than packets from a single portscan.

Zhao in [97] shows that packet sampling can improve the efficiency of data collection and strengthen the processing performance of IDS for high-speed networks. Simple random sampling and stratified sampling is compared. He uses content depend sampling and compares the certain bits of IP header of every packet. The stratification scheme design is based on the packet type. Results show that stratified sampling performs better and gives higher detection rate.

Lakhina in [98] and [99] builds traffic flows from random sampled packets and then aggregates them at the Origin-Destination (OD) level. It applies the subspace method [99] to multivariate time series of OD traffic defined as number of bytes, packets and IP-level flows. He shows that each of these time series reveals a different class of anomalies. The anomalies span a remarkably wide spectrum of event types, including denial of service attacks (single-source and distributed), flash crowds, port scanning, downstream traffic engineering, high-rate flows, worm propagation, and network outage.

Huang in [100] developed a packet sampling and filtering mechanism as a countermeasure against flooding DoS attacks. It is integrated into the router congestion control mechanism. If the packet is dropped, then the packet is sampled, its header is extracted and forwarded to the anomaly detection system for further processing based on different thresholds. Also in [101] information about dropped packets is used to trigger an action against suspicious flows. A rate limit is applied against high bandwidth aggregates.

5 Conclusions

5.1 Adaptive sampling

Packet sampling has been available for many years. It has been successfully used in many applications, like accounting, billing, SLA, etc. with acceptable accuracy.

However, in order to preserve the accuracy and provide accurate estimations, the sampling technique should adapt to network state and traffic. We consider adaptive sampling as an interesting approach. It could help to build accurate time series of different parameters and improve the accuracy of classification of traffic into flows, groups, etc.

There are other reasons for using adaptive sampling. Network devices have certain limits in terms of resources available for sampling. Some network devices might even stop sampling during traffic bursts. So, the sampling mechanism should be dynamic in order not to lose any important information during these busy periods as well as to get enough information during idle states.

Unfortunately we do not have that many possibilities to change sampling techniques applied in network equipment. However we can change sampling rate, which means that we can control the accuracy by controlling the number of samples received in the defined time period. This decision has to be taken in advance, before network traffic will change. Fortunately, various prediction techniques could help to predict network traffic.

We would like to investigate some prediction techniques and their applications in adaptive sampling. This study will also apply to our network data collector. On one hand we do not want to increase control traffic between the collector and network devices, on the other hand we do not want to delegate the sampling-rate adaptation to these devices and overload them. The two-level adaptation could be a good compromise. For a short interval, the adaptation could be performed by a network device on the basis of the parameters provided and calculated by the collector over the longer time period. This would not increase significantly control traffic, however this approach requires some changes in devices.

5.2 Sampled statistics

As it was pointed out in previous sections, packet sampling may skew various estimates of network traffic properties. This is particularly visible in case of flow building (4.1) and estimating the first order statistics for the traffic (3.1.2).

We would like to conduct several offline experiments on dumps originating both from CERN site, public repositories and `ns2` generator. In these experiments we would like to compare how the sampling method utilized in sFlow compares to other methods when it comes to estimating different traffic parameters. Furthermore we want to see how the sampling influences direct anomaly detection techniques by comparing the output from IDS (like Snort) when fed with full traffic dump and sampled headers respectively.

The goal of these tests is to determine weak and strong points of sFlow sampling. It would also help us to asses whether direct anomaly detection techniques are of any use when working with sampled packet headers. In addition to that we would like to identify the aggregates which could be useful input for various data mining techniques.

References

[1] Joel Apisdorf, K. Claffy, Kevin Thompson, and Rick Wilder. Oc3mon: Flexible, affordable, high performance staistics collection. In *LISA 96: Proceedings of the 10th USENIX conference on System administration*, pages 97–112, Berkeley, CA, USA, 1996. USENIX Association.

[2] R. Caceres, N. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P. Mishra, K. Ramakrishnan, J. Rexford, F. True, and J. van der Merwe. Measurement and analysis of ip network usage and behaviour, 2000.

[3] Chuck Cranor, Yuan Gao, Theodore Johnson, Vlaidslav Shkapenyuk, and Oliver Spatscheck. Gigascope: high performance network monitoring with an sql interface. In *SIGMOD 02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 623–623, New York, NY, USA, 2002. ACM.

[4] Ken Keys, David Moore, Ryan Koga, Edouard Lagache, Michael Tesch, and k claffy. The architecture of CoralReef: an Internet traffic monit oring software suite. In *PAM2001 — A workshop on Passive and Active Measurements*. CAIDA, RIPE NCC, April 2001. http://www.caida.org/tools/measurement/coralreef/.

[5] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi. Design and deployment of a passive monitoring infrastructure. *Lecture Notes in Computer Science*, 2170:556+, 2001.

[6] N.G. Duffield. Sampling for passive internet measurement: A review. *Statistical Science*, 19(3):472–498, 2004.

[7] P.D. Amer and L.N. Cassel. Management of sampled real-time network measurements. In *Proceedings 14th Conference on Local Computer Networks*, pages 62–68, October 1989.

[8] J. Micheel, H. Braun, and I. Graham. Storage and bandwidth requirements for passive internet header traces, 2001.

[9] Peter Phaal. Network monitoring device and system (us patent 5,315,580), August 1991. US Patent 5,315,580.

[10] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of ethernet traffic. In *SIGCOMM*, pages 183–193, 1993.

[11] J. Beran, R. Sherman, M.S. Taqqu, and W. Willinger. Long-range dependence in variable-bit-rate video traffic. *IEEE Transactions on Communications*, 43(234):1566–1579, Feb/Mar/Apr 1995.

[12] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[13] Yuan Gao, Guanghui He, and Jennifer C. Hou. On exploiting traffic predictability in active queue management. In *INFOCOM*, 2002.

[14] S.A.M. Ostring and H. Sirisena. The influence of long-range dependence on traffic prediction. In *IEEE International Conference on Communications*, volume 4, 2001.

[15] Kimberly C. Claffy, George C. Polyzos, and Hans-Werner Braun. Application of sampling methodologies to network traffic characterization. In *SIGCOMM*, pages 194–203, 1993.

[16] Cisco Systems. *Sampled NetFlow*, 2003. http://www.cisco.com/.

[17] Juniper packet sampling. http://www.juniper.net.

[18] Baek-Young Choi and Supratik Bhattacharrya. On the accuracy and overhead of cisco sampled netflow. In *ACM Sigmetrics Workshop on Large-Scale Network Inference (LSNI)*, Banff, Canada, June 2005.

[19] S. Goldberg and J. Rexford. Security vulnerabilities and solutions for packet sampling. 2007.

[20] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for ip performance metrics, 1998.

[21] Baek-Young Choi, Jaesung Park, and Zhi-Li Zhang. Adaptive packet sampling for flow volume measurement. *Computer Communication Review*, 32(3):9, 2002.

[22] Banchong Harangsri, John Shepherd, and Anne H. H. Ngu. Selectivity estimation for joins using systematic sampling. In *DEXA Workshop*, pages 384–389, 1997.

[23] Guanghui He and Jennifer C. Hou. An in-depth, analytical study of sampling techniques for self-similar internet traffic. In *25th IEEE International Conference on Distributed Computing Systems (25th ICDCS2005)*, pages 404–413, Columbius, OH, 2005. IEEE. UIUC.

[24] Guanghui He and Jennifer C. Hou. On sampling self-similar internet traffic. *Computer Networks*, 50(16):2919–2936, 2006.

[25] Vern Paxson. End-to-end routing behavior in the internet. *IEEE/ACM Trans. Netw.*, 5(5):601–615, 1997.

[26] A. Adams. The use of end-to-end multicast measurements for characterizing internal network behavior.

[27] Feng Zhang and Zhenming Lei. The evaluation of poisson packet sampling measurement techniques. In *Joint Conference of the 10th Asia-Pacific Conference on Communications*, volume 1, pages 239–243, September 2004.

[28] Guang Cheng and Jian Gong. Traffic behavior analysis with poisson sampling on high-speed network. In *Proceedings.of International Conferences on Info-tech and Info-net, 2001.*, volume 5, pages 158–163, 2001.

[29] sflow home page. http://sflow.org.

[30] Random sampled netflow. http://www.cisco.com/.

[31] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall. Sampling and filtering techniques for ip packet selection.

[32] Jonathan Jedwab and Peter Phaal. Traffic estimation for the largest sources on a network, using packet sampling with limited storage. Technical Report HPL-92-35, Hewlett Packard Laboratories, March 05 1992.

[33] Baek-Young Choi, Jaesung Park, and Zhi-Li Zhang. Adaptive random sampling for load change detection, July 11 2002.

[34] Baek-Young Choi and Zhi-Li Zhang. Adaptive random sampling for traffic volume measurement. *Telecommunication Systems*, 34(1-2):71–80, 2007.

[35] J. Drobisz and Kenneth J. Christensen. Adaptive sampling methods to determine network traffic statistics including the hurst parameter. In *LCN*, pages 238–, 1998.

[36] Edwin A. Hernandez, Matthew C. Chidester, and Alan D. George. Adaptive sampling for network management. *J. Netw. Syst. Manage.*, 9(4):409–434, 2001.

[37] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. On random sampling over joins. In *SIGMOD 99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 263–274, New York, NY, USA, 1999. ACM.

[38] Duffield, Lund, and Thorup. Learn more, sample less: Control of volume and variance in network measurement. *IEEETIT: IEEE Transactions on Information Theory*, 51, 2005.

[39] Nick G. Duffield, Carsten Lund, and Mikkel Thorup. Flow sampling under hard resource constraints. In Edward G. Coffman Jr., Zhen Liu, and Arif Merchant, editors, *SIGMETRICS*, pages 85–96. ACM, 2004.

[40] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw*, 11(1):37–57, 1985.

[41] Claudia Ulbricht. Multi-recurrent networks for traffic forecasting. In *National Conference on Artificial Intelligence*, pages 883–888, 1994.

[42] J. T. Connor, R. D. Martin, and L. E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, 1994.

[43] Cheng Guang, Gong Jian, and Ding Wei. A traffic sampling model for measurement using packet identification. In *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, pages 409–413, 27-30 Aug. 2002.

[44] Cheng Guang, Jian Gong, and Wei Ding. Network traffic sampling model on packet identification. In *ICN (1)*, pages 758–765, 2005.

[45] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In *SIGCOMM 00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 271–282, New York, NY, USA, 2000. ACM.

[46] N. Duffield, A. Gerber, and M. Grossglauser. Trajectory engine: A backend for trajectory sampling, 2002.

[47] N. DUFFIELD and M. GROSSGLAUSER. Trajectory sampling with unreliable reporting, 2007.

[48] Young H. Cho and William H. Mangione-Smith. Deep network packet filter design for reconfigurable devices. *ACM Transactions on Embedded Computing Systems (ACM TECS)*, 2007.

[49] Fang Yu, Zhifeng Chen, Yanlei Diao, T. V. Lakshman, and Randy H. Katz. Fast and memory-efficient regular expression matching for deep packet inspection. In Laxmi N. Bhuyan, Michel Dubois, and Will Eatherton, editors, *ANCS*, pages 93–102. ACM, 2006.

[50] Sailesh Kumar, Jonathan S. Turner, and John Williams. Advanced algorithms for fast and scalable deep packet inspection. In Laxmi N. Bhuyan, Michel Dubois, and Will Eatherton, editors, *ANCS*, pages 81–92. ACM, 2006.

[51] Sunil Kim and Jun-Yong Lee. A system architecture for high-speed deep packet inspection in signature-based network intrusion prevention. *Journal of Systems Architecture*, 53(5-6):310–320, 2007.

[52] Young H. Cho and William H. Mangione-smith. Programmable hardware for deep packet filtering on a, July 27 2004.

[53] Jia Ni, Chuang Lin, Zhen Chen, and Peter D. Ungsunan. A fast multi-pattern matching algorithm for deep packet inspection on a network processor. In *ICPP*, page 16. IEEE Computer Society, 2007.

[54] Kocak and Kaya. Low-power bloom filter architecture for deep packet inspection. *IEEECLETS: IEEE Communications Letters*, 10, 2006.

[55] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd S. Sproull, and John W. Lockwood. Deep packet inspection using parallel bloom filters. *IEEE Micro*, 24(1):52–61, 2004.

[56] Broder and Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics, A K Peters, Ltd.*, volume 1. 2004.

[57] M. Schöller, T. Gamer, R. Bless, and M. Zitterbart. An Extension to Packet Filtering of Programmable Networks. Sophia Antipolis, France, November 2005. 7th International Working Conference on Active Networking (IWAN) 2005.

[58] Tanja Zseby. Stratification strategies for sampling-based non-intrusive measurements of one-way delay.

[59] Tanja Zseby. Deployment of sampling methods for sla validation with non-intrusive measurements. In *Passive and Active Measurement Workshop, Fort Collins, CO, April 2002*, 2002.

[60] T. Zseby. Sampling techniques for non-intrusive qos measurements: Challenges and strategies. *Computer Communications Special Issue on Monitoring and Measurement*, 2005.

[61] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In Roch Guerin, Ramesh Govindan, and Greg Minshall, editors, *SIGCOMM*, pages 169–180. ACM, 2005.

[62] Ken Keys, David Moore, and Cristian Estan. A robust system for accurate real-time summaries of internet traffic. In Derek L. Eager, Carey L. Williamson, Sem C. Borst, and John C. S. Lui, editors, *SIGMETRICS*, pages 85–96. ACM, 2005.

[63] Kuai Xu, Feng Wang, Supratik Bhattacharyya, and Zhi-Li Zhang. A real-time network traffic profiling system. In *DSN*, pages 595–605. IEEE Computer Society, 2007.

[64] Richard G. Clegg, Hamed Haddadi, Raul Landa, and Miguel Rio. Towards informative statistical flow inversion. *CoRR*, abs/0705.1939, 2007.

[65] C. Estan and G. Varghese. New directions in traffic measurement and accounting, 2001.

[66] Rong Pan, Lee Breslau, Balaji Prabhakar, and Scott Shenker. Approximate fairness through differential dropping. *Computer Communication Review*, 33(2):23–39, 2003.

[67] N. Duffield, C. Lund, and M. Thorup. Charging from sampled network usage, 2001.

[68] Wenjia Fang and Larry Peterson. Inter-AS traffic patterns and their implications. Technical Report TR-598-99, Princeton University, Computer Science Department, March 1999.

[69] Anja Feldmann, Albert G. Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True. Deriving traffic demands for operational IP networks: methodology and experience. In *SIGCOMM*, pages 257–270, 2000.

[70] Chadi Barakat, Gianluca Iannaccone, and Christophe Diot. Ranking flows from sampled traffic. In *CoNEXT 05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 188–199, New York, NY, USA, 2005. ACM.

[71] Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. pages 331–342, 1998.

[72] G. S. Manku and R. Motwani. Approximate frequency counts over data streams, 2002.

[73] N. Duffield and C. Lund. Predicting resource usage and estimation accuracy in an ip flow measurement collection infrastructure, 2003.

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

[75] Tatsuya Mori, Masato Uchida, Ryoichi Kawahara, Jianping Pan 0002, and Shigeki Goto. Identifying elephant flows through periodically sampled packets. In Alfio Lombardo and James F. Kurose, editors, *Internet Measurment Conference*, pages 115–120. ACM, 2004.

[76] Rate Estimation Murali. Runs based traffic estimator (rate): A simple, memory efficient scheme for per-flow.

[77] Gion Reto Cantieni, Gianluca Iannaccone, Chadi Barakat, Christophe Diot, and Patrick Thiran. Reformulating the monitor placement problem: Optimal network-wide sampling.

[78] M. Basseville and A. Benveniste. *Detection of Abrupt Changes in Signals and Dynamical Systems.* Springer, Berlin, 1986.

[79] Benefits of flow analysis using sflow: Network visibility, security and integrity. http://www.foundrynet.com/pdf/wp-lancope-sflow.pdf.

[80] Peter Phaal. Detecting nat devices using sflow. http://www.sflow.org/detectNAT/.

[81] Michal Zalewski. p0f. http://lcamtuf.coredump.cx/p0f.shtml.

[82] Maarten Van Horenbeeck. Dns tunneling. http://www.daemon.be/maarten/dnstunnel.html.

[83] G. Androulidakis, V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and V. Maglaris. Understanding and evaluating the impact of sampling on anomaly detection techniques. In *Military Communications Conference, 2006. MILCOM 2006*, pages 1–7, October 2006.

[84] D. Shin K. G. Wang, H. Zhang. Change-point monitoring for the detection of dos attacks. *IEEE Transactions on Dependable and Secure Computing*, 1(4):193–208, 2004. Member-Haining Wang and Member-Danlu Zhang and Fellow-Kang G. Shin.

[85] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Proactively detecting distributed denial of service attacks using source IP address monitoring. In Nikolas Mitrou, Kimon P. Kontovasilis, George N. Rouskas, Ilias Iliadis, and Lazaros F. Merakos, editors, *NETWORKING*, volume 3042 of *Lecture Notes in Computer Science*, pages 771–782. Springer, 2004.

[86] V. Chatzigiannakis, S. Papavassiliou, G. Androulidakis, and B. Maglaris. On the realization of a generalized data fusion and network anomaly detection framework. In *Fifth International symposium on Communication System*, 2006.

[87] Jianning Mai, Chen-Nee Chuah, Ashwin Sridharan, Tao Ye, and Hui Zang. Is sampled data sufficient for anomaly detection? In Jussara M. Almeida, Virgılio A. F. Almeida, and Paul Barford, editors, *Internet Measurement Conference*, pages 165–176. ACM, 2006.

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

[89] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy*, pages 211–225. IEEE Computer Society, 2004.

[90] Avinash Sridharan Ye and Supratik Bhattacharyya. Connectionless port scan detection on the backbone. In *25th IEEE International Performance, Computing, and Communications Conference*, pages 10–, April 2006.

[91] J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye. Impact of packet sampling on portscan detection. *IEEE Journal on Selected Areas in Communications*, 24(12):2285–2298, December 2006.

[92] Ryoichi Kawahara, Tatsuya Mori, Noriaki Kamiyama, Shigeaki Harada, and Shoichiro Asano. A study on detecting network anomalies using sampled flow statistics. In *SAINT Workshops*, page 81. IEEE Computer Society, 2007.

[93] Ishibashi Keisuke, Kawahara Ryoichi, Tatsuya Mori, Kondoh Tsuyoshi, and Asano Shoichiro. Effect of sampling rate and monitoring granularity on anomaly detectability. In *IEEE Global Internet Symposium, 2007*, pages 25–30, May 2007.

[94] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic distributions. Technical Report 2005-002, CS Department, Boston University, February 10 2005. Fri, 12 Oct 2007 09:56:20 GMT.

[95] Arno Wagner and Bernhard Plattner. Entropy based worm and anomaly detection in fast IP networks. In *WETICE*, pages 172–177. IEEE Computer Society, 2005.

[96] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In Jussara M. Almeida, Virgılio A. F. Almeida, and Paul Barford, editors, *Internet Measurement Conference*, pages 159–164. ACM, 2006.

[97] Kuo Zhao, Liang Hu, Guannan Gong, Meng Zhang, and Kexin Yang. Comparison of two sampling-based data collection mechanisms for intrusion detection system. In Hamid R. Arabnia and Selim Aissi, editors, *Security and Management*, pages 261–265. CSREA Press, 2006.

[98] Anukool Lakhina, Mark Crovella, and Christophe Diot. Characterization of network-wide anomalies in traffic flows. Technical Report 2004-020, CS Department, Boston University, May 14 2004. Fri, 12 Oct 2007 09:56:20 GMT.

[99] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. Technical Report 2004-008, CS Department, Boston University, February 24 2004. Fri, 12 Oct 2007 09:56:20 GMT.

[100] Yih Huang and J.M. Pullen. Countering denial-of-service attacks using congestion triggered-packet sampling and filtering. In *Proceedings of Tenth International Conference on Computer Communications and Networks, 2001.*, pages 490–494, 2001.

[101] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3):62–73, 2002.