A Scalable Sampling Scheme for Clustering in Network Traffic Analysis

Abdun Mahmood
Department of Computer Science and Software Engineering
University of Melbourne
Melbourne, Australia
abdun@csse.unimelb.edu.au

Christopher Leckie
Department of Computer Science and Software Engineering
University of Melbourne
Melbourne, Australia
caleckie@csse.unimelb.edu.au

Parampalli Udaya
Department of Computer Science and Software Engineering
University of Melbourne
Melbourne, Australia
udaya@csse.unimelb.edu.au

ABSTRACT

Sampling is a popular method for improving the scalability of analyzing massive datasets such as network traffic traces, web-click traffic and other forms of transaction data. However, in some cases, existing simple sampling strategies fail to capture the underlying distribution of the data. In particular, for network traffic, sampling is influenced by heavy traffic from flash crowds and Denial of Service (DoS) attacks. In such cases, it reveals little information about the other smaller traffic patterns which may contain interesting yet important information about the traffic. We propose an adaptive sampling technique that utilizes a buffer of frequently seen patterns and a combination of sampling steps to build a hierarchical tree of traffic clusters. We show that this sampling technique ensures that smaller and newer patterns are represented in the cluster tree while satisfying the maximum sampling rate imposed by the resource constraints. This technique has two benefits: it preserves the underlying patterns of the data, and improves efficiency by reducing the sampling of records from known patterns. Through an empirical evaluation on a benchmark dataset, we demonstrate the accuracy of our system in detecting certain types of rare attacks that are otherwise not detected by systematic sampling. We also demonstrate the efficiency of our system in terms of reducing the number of sampled records in detecting frequent patterns.

Categories and Subject Descriptors

D.3.3 [**Computer-Communication Networks**]: Network Operations-Network monitoring

General Terms

Algorithms, **Management**, **Measurement**, **Performance**

Keywords

adaptive sampling; network traffic analysis; clustering

INFOSCALE 2007, June 6-8, Suzhou, China
Copyright © 2007 ICST 978-1-59593-757-5
DOI 10.4108/infoscale.2007.930

1. INTRODUCTION

There is a growing need to characterize network traffic data for a number of network management services. These include analysis of traffic volume, traffic dynamics, traffic mixture, and internet security. Often these characterization tasks require using some form of data mining technique such as frequent itemset mining [1] and clustering [2, 3]. However, most of these techniques perform poorly when large amounts of data are required to be analyzed at or near network transmission speeds. Sampling is a popular technique for data reduction. An open problem in analyzing network traffic data is how to combine sampling with selection so that rare patterns in the traffic can be recognized. In this paper, we present a two-stage adaptive sampling scheme to address these problems.

Traditional approaches to sampling are often inadequate to capture the underlying distribution of the data. In particular, for network traffic, sampling is influenced by high volume traffic from flash crowds and Denial of Service (DoS) attacks. When a DoS attack is active, a naïve sampling technique would be biased towards the distribution of the DoS traffic. When used with a clustering algorithm, a naïve sampling scheme would be unable to capture smaller clusters since they will be sampled less than the DoS attack traffic. Our goal is to avoid wasting resources on clustering traffic patterns that have been already represented by large clusters.

Our approach is a two-stage sampling scheme. In the first stage, traffic is sampled at a higher rate relative to the overall sampling rate required by the user. This is followed by a selection stage where sampled traffic is systematically matched against a buffer of previously observed traffic patterns. Sampled traffic that matches a pattern in the buffer is then filtered through a second stage of sampling, so that only a subset of traffic of known patterns is passed to the clustering system. All traffic that did not match the buffer is passed to the clustering system as well.

A key advantage of this approach is that we can increase the proportion of computational resources that are spent on new or unusual traffic patterns. Note that we do not eliminate previously seen traffic, but only sample them at a lower rate, thus making our sampling more efficient in terms of the number of different patterns captured. Consequently, this approach enables us to identify smaller but still significant clusters more accurately than if traditional sampling approaches were employed. The key contributions of this paper are: i) a novel two stage sampling scheme for use in resource-constrained traffic characterization, ii) a strategy for selecting the sampling rates in each sampling stage, and iii) an evaluation on a standard benchmark dataset which demonstrates that our scheme can achieve greater accuracy for smaller traffic clusters in comparison to traditional systematic sampling, without significantly degrading the identification of larger traffic patterns.

The organization of the rest of this paper is as follows. Section 2 motivates the use of adaptive sampling in resource constrained environments where there is a need to identify smaller but potentially significant patterns and discusses related work. Section 3 gives the background of an efficient hierarchical clustering algorithm that we have used in our experiments. Section 4 formally describes the problem and establishes its scope. Section 5 describes our adaptive sampling scheme and Section 6 analyzes of our evaluation.

2. MOTIVATION AND RELATED WORK: FROM ELEPHANTS TO MICE

Due to the growth in the bandwidth of networks, the volume of network traffic data is often too large for analysis using traditional data mining techniques [4]. In particular, it is difficult to collect, store and analyze huge amounts of network traffic data. Consequently, there is increasing interest in scalable solutions for mining network traffic data. For example, system administrators need to identify significant categories of traffic that are consuming resources in the network, such as DoS attacks, flash crowd events or peer-to-peer traffic.

Numerous approaches have been proposed for the problem of clustering large traffic flows. Cormode et. al [5, 6] proposed solutions to deal with *heavy-hitters* by using sketches and summaries of network traffic. Estan et. al [1] developed a traffic summarization technique called AutoFocus, which applies frequent itemset mining to network flows. In our previous work [2], we have developed a clustering technique to speed up the summarization process of network data under constrained memory. However, the common problem in all these techniques is the lack of scalability as network bandwidth increases.

Although sampling is a popular solution for data mining and statistical analysis of large datasets, such as network traffic flows, traditional sampling techniques are often inadequate to capture the underlying distributions [7]. For example, systematic sampling or uniform random sampling is heavily biased by the density of the dataset. Sampling a typical network trace containing a large number of probes or DoS attack packets will dwarf the other smaller attacks or interesting patterns. Uniform random sampling has been previously used in the context of clustering techniques [8] [9]. In applications demanding a very large amount of data, sampling is an important and necessary technique to reduce the volume of data [2]. Xu et. al [10] used random sampling to create a "Relative Uncertainty" profile of network flows, which is then used to characterize new flows using a frequent itemset mining technique similar to AutoFocus [1]. The authors then use this technique to create a behavioral pattern and block potential "exploit traffic" [11] by constructing the Access Control Lists for routers. Gonzalez and Paxson [12], also inspired by AutoFocus, proposed packet level random sampling to detect heavy-hitters in the network traffic.

It has been observed [7, 13] that a small number of heavy flows account for a large amount of traffic. Similarly, Estan et. al [14] claimed that it is infeasible to accurately measure all flows on high speed links, however, keeping track of only a few large flows (called "Elephants") may be sufficient. They proposed a *sample and hold* algorithm, which shares the common principle of counting samples with Gibbons and Matias [15] but identifies large flows using less memory. In [16, 17], sampling techniques are proposed for clustering based on the density of the clusters. Palmer et. al [16] developed an algorithm to find clusters under the assumption of a Zipf distribution for the sizes of clusters. Kollios et. al [17] proposed a variation of this technique using kernel based density estimation. Unlike these previous works, our emphasis is on the smaller flows that are often dominated by the larger flows. For example, in uniform random sampling, every record has an equal probability of being sampled [18]. In the case of finding clusters for rare or infrequent classes of traffic, it is often desirable that rare records are sampled with a higher sampling rate than records belonging to frequent classes [19].

In summary, the main focus of traffic analysis techniques such as [3, 13, 20] have been to focus on identifying the large traffic flows ("Elephants"), while ignoring the large number of smaller flows ("Mice"). However, many smaller clusters of flows can still contain relevant information. For example, many types of large traffic problems start out small, such as worm spread or distributed DoS attacks. The open issue that we address is how to develop a scalable data mining technique that can help to identify smaller traffic patterns in a computationally efficient manner.

2.1 A Motivating Example - The 1998 DARPA dataset

In order to demonstrate the distribution of the sizes of traffic patterns in networks, it is useful to examine a packet trace with labeled patterns. Publicly available labeled traffic data are very rare because of security and privacy concerns [20]. The Lincoln Laboratory DARPA intrusion detection data repository [21] is one of the largest publicly available traffic traces. For example, the 1998 DARPA traffic traces include 25 days of traffic data with labeled attack information, collected from a purpose-built network, following the behavior model of both normal users and malicious users. In the absence of real life labeled data, these traces provide us with a classified set of traffic patterns, which can be used as the basis for evaluating methods to characterize network traffic.

It is interesting to note that although there may be many different types of attacks present in a dataset, the overwhelming majority of attacks (by the number of flows involved) are caused by Denial of Service (*Neptune* 8%, *smurf* 13%) and Probe (*satan* 2%, *portsweep* 1%). The other attacks types are User-to-Local (U2R) and Root-to-Local (R2L). More information about these categories and the attacks they represent can be found in [22]. Table 1 shows the top 10 attacks and their categories. This gives us a picture of the relative frequency of different attacks in terms of the number of flows involved in the attack and their categories. Clearly, the most flow intensive attack categories are DoS and Probe.

TABLE 1 TOP 10 ATTACKS WITH FREQUENCY FROM DAY 1-25 OF THE 1998 DARPA TRACES

Attack Category	Flows	Attacks
DoS	1526628	Neptune
DoS	249609	Smurf
Probe	32632	Satan
Probe	15406	Ipsweep
Probe	10504	Portsweep
DoS	10045	Pod
Probe	2356	Nmap
DoS	2172	Teardrop
DoS	1766	Warezclient
DoS	1281	Back

Any attempt to cluster flow records from traces such as these will be hampered by the volume of data to be analyzed. Moreover, if traditional sampling is used, most of the sampled records will still come from the top 5 to 10 traffic patterns as shown in Table 1. Many of the smaller, but still significant patterns will be overlooked at low sampling rates. To illustrate this problem, consider the distribution of the sizes of clusters found as a result of clustering network traffic flows using an existing clustering tool [2], which we describe in more detail later. We can see from Figure 1, that there is a noticeable degree of separation between the sizes of the larger clusters and the sizes of the smaller clusters, indicating the nature of the network traffic representing a heavy-tailed distribution. Thus, low rate sampling schemes will miss a significant proportion of the traffic patterns on this network. Our goal is to improve the scalability of clustering techniques for characterizing network traffic patterns, so that we have a better chance of identifying these smaller traffic patterns with constrained computational resources.

3. BACKGROUND: THE ECHIDNA ALGORITHM FOR CLUSTERING NETWORK TRAFFIC DATA

The scalability problem we address in this paper is relevant to any clustering algorithm, such as frequent itemset and partition-based clustering algorithms. However, in this paper we focus on using a hierarchical clustering algorithm called Echidna [2], which we have previously developed for clustering network traffic data. In this section we briefly describe the basic functionality of Echidna.

Our approach to finding multidimensional clusters of network data builds on the BIRCH framework [3], which is a clustering algorithm that uses a *Cluster Feature* (CF) to represent a cluster of records in the form of a vector <*n, LS, SS*>, where *n* is the number of records in the cluster, *LS* is the linear sum and *SS* is the square sum of the attributes of the records. Clusters are built using a hierarchical tree called a *Cluster Feature* Tree (CF-Tree) to summarize the input records.

The tree is built in an agglomerative hierarchical manner (see Figure 2). Each leaf node consists of *L* clusters, where each cluster is represented by its CF record. These CF records can themselves be clustered at the non-leaf nodes. Figure 2 shows a CF-Tree.

Echidna modifies the BIRCH framework for clustering in order to exploit the hierarchical structure of network traffic attributes, such as IP addresses.

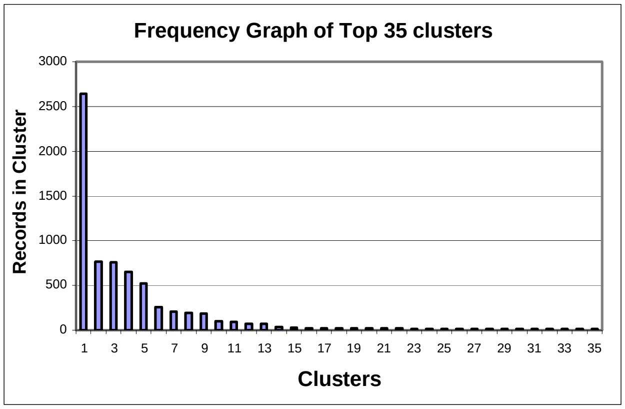

Figure 1 Distribution of cluster sizes as a result of clustering Dataset 6 (Week 4, Day 1) of the 1998 DARPA traces using the Echidna clustering tool [2].

The input data is extracted from network traffic as 6-tuple records <*SrcIP, DstIP, Protocol, SrcPort, DstPort, bytes*>, where *SrcIP, DstIP* are *hierarchical* attributes, *bytes* is numerical and the rest are *categorical* attributes.

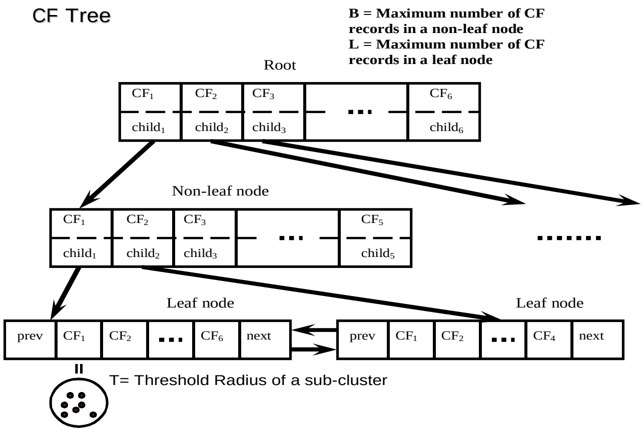

Figure 2 Cluster Feature Tree

Echidna takes each record and iteratively builds a hierarchical tree of clusters called a CF-Tree. In order to support these different types of attributes, Echidna provides an integrated approach to distance calculations, which incorporate distance functions for numerical, hierarchical and categorical attributes. In particular, Echidna can find clusters that represent a generalization of attribute values, such as a subnet that corresponds to a set of IP addresses. This provides a natural representation for describing a generalized pattern of network flows as a cluster.

Each cluster C_m is represented by a cluster feature vector that contains sufficient statistics to calculate the centroid $\overline{c}_m$ and *radius* ρ_m of the cluster. Each data record *R*, corresponding to a 6-tuple traffic flow record, is inserted by comparing *R* to the closest cluster starting from the root along a path *P* to a leaf node. At the leaf node, the data record *R* is inserted into the closest C_m and the *radius* ρ_m of the updated cluster is calculated. If $\rho_m > T$, where *T* is a threshold value in the range [0,1], and if the number of CF entries in the node is less than a maximum value, then *R* is inserted into the node as a new cluster. If a node has no more space for a new CF entry, then the node is split to create a new node and the path to the root is updated recursively.

The clusters at each level represent a generalized set of traffic flows, which can be used to describe the traffic flows in the network. Since there is redundant information between different levels, the summary report should contain only those nodes of any level having significant additional information compared to their descendant levels. We define significant nodes in terms of the number of records, and the ratio of *Average Intra-Cluster* distance and *Maximum Intra-cluster* distance measures that intuitively pick those nodes that contain a heterogeneous set of clusters. Details of this can be found in [2].

Since the total number of attributes and their range of values are fixed, we can consider that the cost of distance calculation between a record and a cluster is also constant. In a height-balanced CF-Tree with branching factor B and m nodes, $log_B\ m$ comparisons are required for each record to be inserted into the closest leaf cluster. For N records the insertion time is bounded by $O(N * B(1+\log_B m))$.

In the rest this paper, we use Echidna as the underlying clustering tool for finding network traffic patterns, and we examine the problem of how to improve the scalability of this tool by using adaptive sampling. Note that we focus on Echidna for testing purposes; our proposed sampling scheme is applicable to a wide range of clustering techniques.

3.1 Problem Statement

We are given as input a sequence of flow records that have been extracted from a network traffic stream, where each flow record is a 5-tuple <SrcIP, DstIP, Protocol, SrcPort, DstPort>.

Our goal is to cluster these flow records into a set of generalized flows which represent significant patterns in the underlying traffic stream. Note that clustering is a resource constrained problem if the network traffic stream to be clustered is from a high speed network. Clustering is both CPU bounded, in terms of the maximum rate at which we can update clusters, and memory bounded, in terms of the maximum number of flows that can be kept in memory. In this context, if the arrival rate of new flow records exceeds the maximum rate at which we can update clusters with a new record, then we need to use some form of sampling in order to satisfy these resource constrains. Please see [7] for more discussion on the necessity of sampling for analyzing high speed network traffic.

Traditional approaches to sampling network traffic include systematic, simple random and stratified random sampling [4]. These approaches have the effect of sampling flow records from each underlying cluster at the same rate.

In practice, however, the distribution of the sizes of the underlying clusters corresponding to the significant flows in the traffic is heavy-tailed [7], i.e., many clusters contain only a small number of flows, while a few clusters contain many flows.

If traditional sampling techniques are applied in this case, then there will be a penalty in terms of the ability to accurately discover the smaller clusters. This is due to an observation by Guha et al. [8] and Kollios et al. [17], who noted that a minimum number of points (i.e., flows) need to be sampled from each cluster in order for those clusters to be recognized by the clustering algorithm. Consequently, if the sampling rate is too low, then there is a high probability that an insufficient number of flows will be sampled from a small cluster.

Suppose there are M classes of data among N records, where, $M<N$. For systematic sampling or random sampling, the probability of sampling an individual record for cluster C_m is

$$P_m = \frac{N_m}{N} S \qquad (1)$$

where, N_m is the number of records from class C_m and S is the Sampling Rate, $0 < S \leq 1$.

Our goal is to develop a sampling scheme such that frequent clusters are sampled at a lower rate S_L, while rare clusters are sampled at a higher rate $S_H > S_L$. In this way, we aim to increase the probability that a sufficient number of records are sampled from smaller clusters.

The key problem that needs to be addressed in this context is how to recognize, during sampling, whether an input record belongs to a rare or frequent cluster, given that the clusters are not all known a priori. In the next section, we describe the two-stage adaptive sampling scheme that we have proposed to address this problem.

4. A TWO-STAGE ADAPTIVE SAMPLING SCHEME FOR CLUSTERING NETWORK TRAFFIC

The architecture of our two-stage sampling scheme is shown in Figure 3. In the first stage, input records are sampled using uniform random sampling, with a sampling probability P_1. Once a record has been selected as a result of the first sampling stage, we need to identify whether it belongs to one of the frequently occurring traffic patterns that have been already been identified in the traffic trace. This is achieved by matching the record R against a buffer B of frequently occurring traffic patterns. If the record R does not match any frequent pattern in B, then R is considered as a new or less frequent pattern, and is passed directly to the Echidna clustering algorithm for inclusion into the cluster tree. In contrast, if R matches a frequent pattern in the buffer B, then R is considered to be less informative, since it already matches a known frequent pattern. Consequently, in that case, R would be passed to a second sampling stage where it would be sampled using uniform random sampling with sampling probability P_2. If the record R passes this second stage of sampling, it is passed to Echidna, otherwise it is discarded. The effect of the second sampling stage is to reduce the rate at which known, frequent patterns are clustered, so that computational resources can be focused on characterizing the new or less frequent patterns.

There are two key research challenges in the design of this two stage sampling scheme. The first challenge is how to select the sampling probabilities P_1 and P_2 given the constraint on the maximum throughput or overall sampling rate of the clustering algorithm, assigned by the user. The second challenge is how to populate and match entries in the buffer which describe known frequent patterns in the traffic. We describe our approach to each of these problems in the subsections that follow.

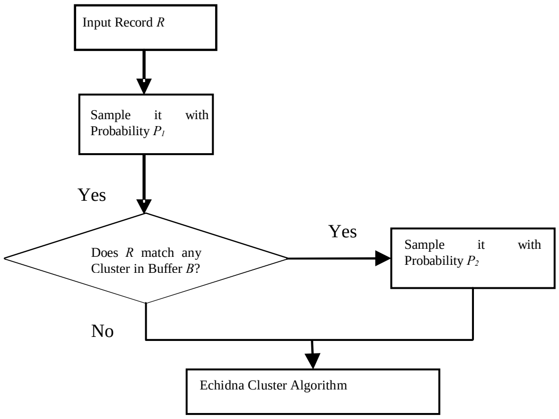
Input Record *R*
Sample it with Probability *P₁*
Yes
Does *R* match any Cluster in Buffer *B*?
Yes
Sample it with Probability *P₂*
No
Echidna Cluster Algorithm

Figure 3 Two-stage Adaptive Sampling Scheme

4.1 Assignment of Sampling Probabilities

Given our two-stage sampling scheme as described above, we require a strategy for selecting the sampling probabilities P_1 and P_2. Let *SR* denote overall sampling ratio of the two-stage sampling scheme, i.e., the proportion of the original input records that are passed to the Echidna clustering subsystem. We can derive an expression for *SR* in terms of the initial sampling probability P_1, the sampling probability of the second stage P_2, and the proportion of records that match an entry in the buffer, which we denote as *X:*

$$SR = P_1[XP_2 + (1-X)] \qquad (1)$$

Note that *SR* is the bound on the throughput of the clustering subsystem, and the match ratio *X* can be measured based on the observed frequency with which records match the buffer contents. Consequently, the two probability values P_1 and P_2 need to be further constrained.

In principle, if the match ratio *X* is large, then a large proportion of the traffic contains known, frequently occurring patterns. In that situation, we can afford to discard a large proportion of these recurring records, i.e., P_2 should be small. Conversely, if few records match the buffer, *X* is small, hence we need to retain a larger proportion of these records, i.e., P_2 should be large. We can formulate this intuition using the constraint

$$P_2 = 1 - X \qquad (3)$$

From the above two equations we can derive an expression for P_1 as

$$P_1 = \frac{SR}{1-X^2} \qquad (4)$$

If we wish to be more aggressive in filtering records that match the buffer, we can constrain P_2 as $P_2 = (1-X)^i, i > 1$ and derive P_1 accordingly. In that case, high values of the match ratio *X* would result in a lower sampling rate P_2 for the known, frequent patterns. The following Figure 4 shows different options for setting P_2 using three different functions

4.2 Management of the Buffer of Known Frequent Patterns

The role of the buffer in our two-stage sampling scheme is to provide a cache of known, frequent patterns in the network traffic. If a sampled input record *R* matches an entry B_i *(i=1,…,m)* in *B*, then it is of less interest to the clustering process. Key challenges in the design of the buffer are how to represent entries in the buffer and how to populate these entries in the buffer.

Each entry in the buffer corresponds to a leaf level node from the hierarchical cluster tree maintained by Echidna. Such a cluster corresponds to a generalized flow record, which may correspond to a source or destination IP sub-network, or a range of source or destination ports.

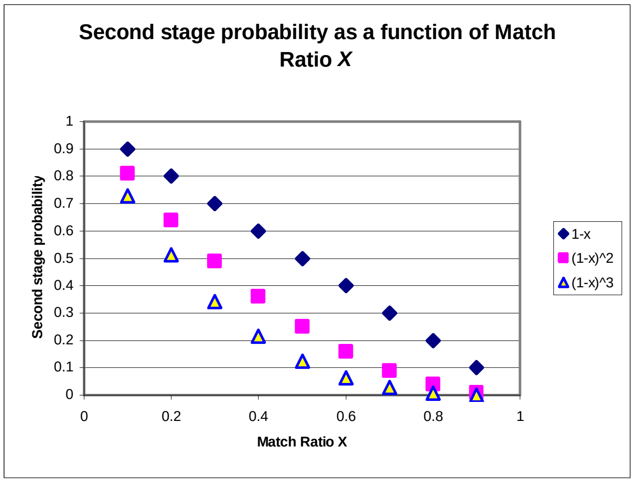

Figure 4 Constraint on second stage sampling probability P2 as a function of the match ratio X

We consider that a sampled input record *R* matches a buffer entry B_i if each attribute of *R* belongs to the range of values represented by the corresponding attribute in B_i. In order to find the similarity between *R* and B_i, the Euclidean distance metric d(*R*, B_i) is used. As explained in Section 3, a traffic record is represented by a vector containing several types of variables or attributes. In the case of matching an IP address against a cluster entry, this only requires a match against the prefix of the two IP addresses. Similarly, in the case of a port this only requires testing whether both the ports are in the "low" or "high" ranges. Thus, the computational overhead of matching records against entries in the buffer is low and bounded by $O(dmP_1T)$, where *d* is the number of dimensions of *R* (in our case it is 5), *m* is the number of entries in the buffer, P_1 is the sampling rate of the first stage, and *T* is the total traffic.

The entries in *B* are populated by extracting the *m* largest leaf nodes from the hierarchical cluster tree. While this is a straightforward process, we still have to address the issue of bootstrapping the system, given that the cluster tree needs to be generated before we can populate the buffer. We solve this by first clustering a sample of the traffic, without using the second sampling stage, i.e., this is equivalent to using our two-stage scheme with an empty buffer. The clustering process allows us to identify dominant flows in the traffic automatically. These dominant flows can then be extracted from the leaves of the cluster tree and used to populate the buffer *B*. In the current system, this is done once per packet trace. An issue for further research is how to incrementally update the buffer in non-stationary environments.

Recalling our observation in Figure 1, there are a few big clusters containing a large proportion of records compared to many small clusters containing small proportions of records. Thus, we expect that a relatively small buffer will be able to match a large proportion of the records sampled by the first stage of our scheme. This means that P_2 will be small, which reduces the proportion of computational resources that are applied to clustering known, frequent patterns in the data.

5. EVALUATION

There is a tradeoff between sampling and accuracy. In applications such as networking it is not possible to accumulate, as well as analyze, the gigabytes of network traffic data generated everyday. Instead, our aim is to filter out some of the repetitive patterns of flows while focusing on the less frequent patterns. In order to evaluate our two-stage sampling scheme, we have conducted an empirical evaluation in terms of a) the effect of sampling on overall accuracy, b) the effect of adaptive sampling on the detection of low volume traffic patterns, and c) the computational efficiency of adaptive sampling compared to traditional systematic sampling.

As the basis for our evaluation, we required a dataset containing known traffic patterns. We have used the 1998 DARPA Intrusion Detection dataset [23], as discussed in Section 2.1, to provide a set of packet traces containing known labeled patterns. From the dataset 25 days of traffic traces from Week 3 to Week 7 were used in the experiments. As a basis for comparison with our two-stage adaptive sampling scheme we use systematic sampling [16, 19, 24, 25], which chooses records at an equal interval $I = SR \times T$ where *SR* is the sampling ratio and *T* is the total number of records. In the following subsections, all evaluations were performed on a time shared dual 2.8GHz Xeon processor machine with 4 GB RAM running SunOS 5.9. The implementation of our algorithm was in Java version 1.5.

5.1 Effect of Sampling Scheme on Accuracy

While sampling saves computational resources, it has the potential to reduce the accuracy of clustering, since many examples or patterns are excluded in the learning process. In this section, we study the effect of reducing the sampling rate on the overall accuracy of our adaptive sampling scheme.

From the labeled traffic traces we identify the records as either belonging to an attack instance or as an instance of normal traffic. We measure accuracy by clustering the DARPA packet trace files, and measuring the number of sampled attack records that map into a cluster containing a majority of attack records. In this context, a True Positive (TP) is an attack record in the trace that maps into a cluster containing a majority of attack records. A False Negative (FN) is a known attack in the trace that, when inserted into the cluster tree, maps into a cluster that contains a majority of normal records. Similarly, a false positive is a normal record that maps into a majority attack cluster. A confusion matrix describing these parameters is shown in Table 2.

Table 2 Confusion metrics to evaluate attack classification

	Predicted connection label (majority class of matching cluster)	
Actual connection label of record	***Normal***	***Attack***
Normal	True Negative (TN)	False Positive (FP)
Attack	False Negative (FN)	True Positive (TP)

We can summarize the overall accuracy in terms of Precision and Recall as follows.

$$\text{Precision} = \frac{TP}{TP + FP}$$

reflects the number of true attacks detected by Echidna as a proportion of the total number of attacks reported. Similarly,

$$\text{Recall} = \frac{TP}{TP + FN}$$

reflects the number of true attack records detected by Echidna as a proportion of the total number of attack records present in the dataset used for clustering.

The effect on Precision and Recall of varying the sampling ratio *SR* in our two-stage adaptive sampling scheme is shown in Figure 7 and Figure 8. The results show the mean Precision and Recall values across the 25 trace files, along with error bars corresponding to ±1 standard deviations. We show how Precision and Recall vary as the sampling ratio *SR* varies from 0.05 to 0.5. We can see that both Precision and Recall are around 80% and above. Moreover, there was no significant effect on the overall Precision and Recall as the sampling ratio was decreased. Thus, using our two-stage sampling scheme, we can achieve the same overall accuracy for lower *SR* settings using fewer computational resources.

5.2 Effect of Adaptive Sampling on Accuracy of Detecting Low Frequency Patterns

While the previous subsection demonstrates that there is no overall degradation in accuracy as a result of our adaptive sampling scheme, we also wanted to examine whether our scheme could improve the accuracy of clustering *smaller* flow patterns present in the packet traces. We analyzed the number of rare attack instances that were detected using our sampling scheme in comparison to systematic sampling.

In particular we studied the number of instances of these rare attacks that were detected as the sampling rate *SR* decreased from 0.25 to 0.12. Figure 7 and Figure 8 show the number of rare attack instances detected using each sampling technique for sampling rates *SR=0.25* and *SR=0.12* respectively. We compare the number of different attacks detected using adaptive and systematic sampling for sampling rates 0.12 and 0.25.

In the case of *SR=0.25*, our two-stage adaptive sampling scheme performed as well or better than systematic sampling in 17 of the 18 attack types detected. In particular, there were three types of attacks (*land*, *loadmodule*, and *warez*) detected using our sampling scheme, which were not detected at all using systematic sampling.

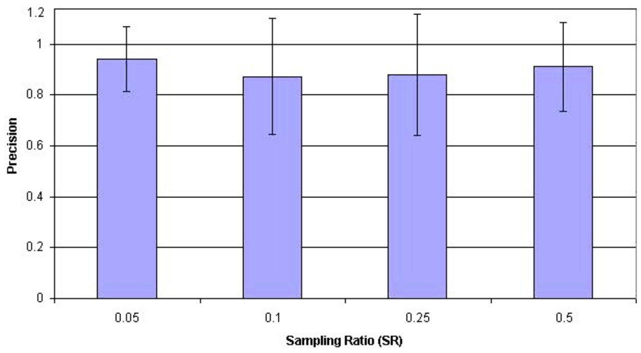

Figure 5 Precision values of Echidna using Adaptive Sampling

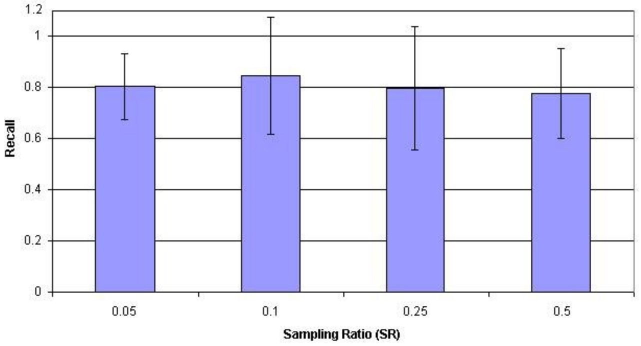

Figure 6 Recall values of Echidna using Adaptive Sampling

Moreover, as the sampling ratio *SR* decreased to *SR=0.12*, the difference in performance between two-stage adaptive sampling and systematic sampling was greater, with a larger number of attack instances detected by our scheme in comparison to systematic sampling. In both cases, adaptive sampling was able to detect some rare attack instances that were not detected by systematic sampling, i.e., *eject*, *land*, *loadmodule*, *multihop*, and *warez* (see Figure 8).

This provides evidence to demonstrate that at low sampling rates, our adaptive sampling scheme can achieve greater accuracy in detecting rare traffic patterns compared to traditional systematic sampling. This is consistent with our expectation that the adaptive scheme should achieve greater accuracy for rare patterns by diverting resources from known, high frequency traffic patterns.

5.3 Computational Efficiency

The introduction of the buffer matching process adds additional computational overhead to the two-stage adaptive sampling. In this section, our aim is to measure the scale of this overhead.

In order to measure this overhead, we have applied both our two-stage adaptive sampling and systematic sampling to packet trace files of different lengths. As a basis for comparison, both schemes have the same overall sampling rate, and the buffer is already populated with records. In this way, we can isolate the overhead due to the buffer lookup process. The computation time required by each scheme is shown in Figure 9.

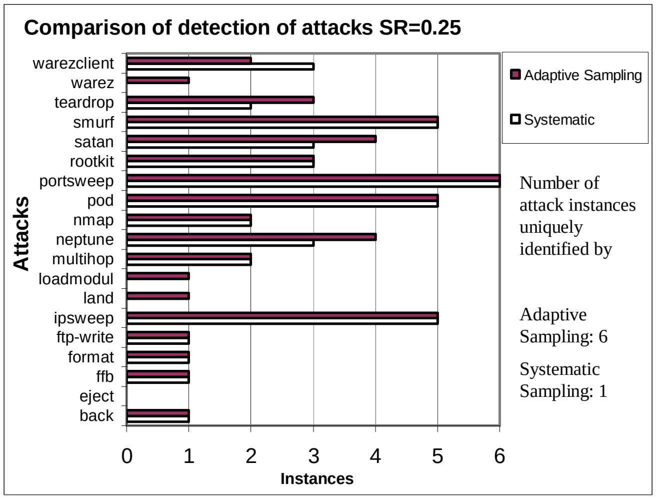

Figure 7 Comparison of attack types for Sampling Rate 0.25

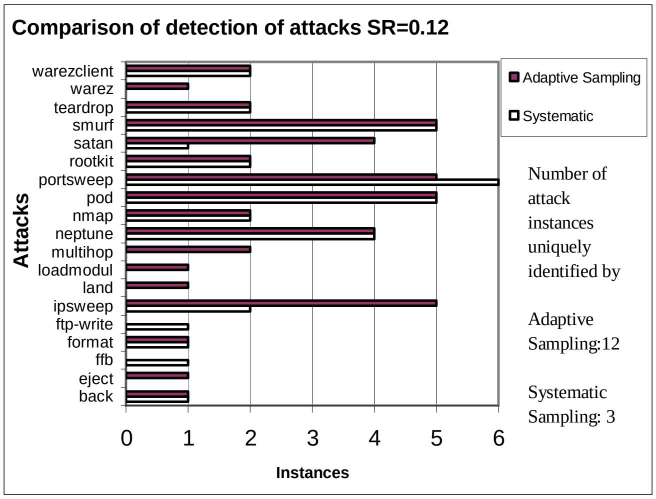

Figure 8 Comparison of attack types for Sampling Rate 0.12

In all but six of the cases, the overhead caused by buffer matching is no more than 30% and for 15 out of 25 instances the overhead is less than 20%.

Overall, both techniques scale linearly as the size of the packet trace to be clustered increases. Thus, there is only a small penalty for introducing the buffer into our sampling scheme, while it provides an advantage in terms of improving the detection accuracy of the clustering algorithm for less frequent traffic patterns. Since, our current implementation of the buffer management scheme has not been optimized for efficiency, we expect that this overhead can be reduced further by using more efficient indexing schemes

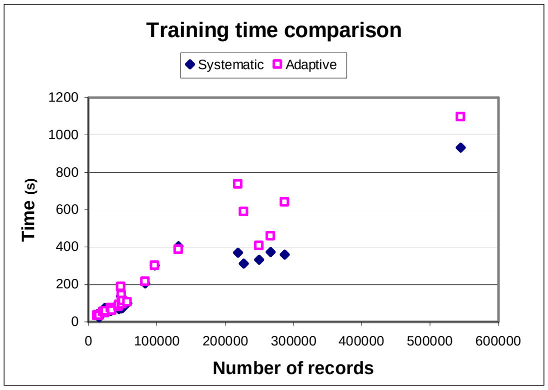

Figure 9 Run time of Adaptive Vs Systematic sampling

6. CONCLUSION

In this paper we have presented a scalable two-stage adaptive sampling scheme to characterize network traffic flows using a clustering algorithm. The sampling technique can be used to identify smaller and rare traffic patterns due to a heavy-tailed distribution of pattern sizes. Our main contributions are how to use and manage a buffer of known frequent patterns to prioritize sampled traffic, and how to adjust the sampling probabilities according to a user provided sampling rate.

We have shown experimentally on a standard benchmark dataset that the accuracy of the underlying clustering algorithm does not deteriorate significantly when the sampling rate is reduced using our sampling scheme. Furthermore, experiments with attack detection have demonstrated that our two-stage adaptive sampling scheme identified rare patterns and attacks that were not identified using systematic sampling. For future research, we are interested in studying the effect of varying the buffer size on training time, as well as developing strategies for how to dynamically update the buffer as changes are observed in the underlying traffic patterns.

7. ACKNOWLEDGMENTS

We thank the MIT Lincoln Laboratory for providing the DARPA Intrusion Detection Evaluation traces.

8. REFERENCES

[1] Estan, C., S. Savage, and G. Varghese. Automatically Inferring Patterns of Resource Consumption in Network Traffic. In *Proceedings of the ACM SIGCOMM Conference*. 2003. pp. 137-148.

[2] Mahmood, A.N., C. Leckie, and P. Udaya. Echidna: Efficient Clustering of Hierarchical Data for Network Traffic Analysis. In *Networking*. 2006. pp. 1092-1098.

[3] Zhang, T., R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of 1996 ACM SIGMOD*. 1996. pp. 103-114.

[4] Claffy, K., G. Polyzos, and H. Braun, Application of Sampling Methodologies to Network Traffic Characterization. *ACM SIGCOMM Computer Communication Review*, 1993. 23(4): pp. 194-203.

[5] Cormode, G., et al. Diamond in the Rough: Finding Hierarchical Heavy Hitters in Multi-Dimensional Data. In *Proceedings of ACM SIGMOD*. 2004. pp. 155-166.

[6] Cormode, G., et al. Finding Hierarchical Heavy Hitters in Data Streams. In *Proceedings of VLDB*. 2003. pp. 464-475.

[7] Duffield, N., C. Lund, and M. Thorup. Charging From Sampled Network Usage. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. 2001. pp. 245-256.

[8] Guha, S., R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proceedings of 1998 ACM SIGMOD*. 1998. pp. 73-84.

[9] Ng, R. and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of VLDB*. 1994. pp. 144-155.

[10] Xu, K., Z. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Cmputer Communications*. 2005. pp. 169-180.

[11] Xu, K., Z. Zhang, and S. Bhattacharyya. Reducing Unwanted Traffic in a Backbone Network. In *Proceedings of Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*. 2005. pp. 9-15.

[12] Gonzalez, J. and V. Paxson. Enhancing Network Intrusion Detection with Integrated Sampling and Filtering. In *Proceedings of RAID*. 2006. pp. 272-289.

[13] Feldmann, A., et al., Deriving Traffic Demands for Operational IP Networks: Methodology and Experience. *IEEE/ACM Transactions on Networking*, 2001. 9(3): pp. 265-279.

[14] Estan, C. and G. Varghese, New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice. *ACM Transactions on Computer Systems*, 2003. 21(3): pp. 270-313.

[15] Gibbons, P. and Y. Matias. New Sampling-based Summary Statistics for Improving Approximate Query Answers. In *Proceedings of 1998 ACM SIGMOD*. 1998. pp. 331-342.

[16] Palmer, C. and C. Faloutsos. Density Biased Sampling: an Improved Method for Data Mining and Clustering. In *Proceedings of 2000 ACM SIGMOD*. 2000. pp. 82-92.

[17] Kollios, G., et al. An Efficient Approximation Scheme for Data Mining Tasks. In *Proceedings of IEEE Int. Conf. on Data Engineering (ICDE'01)*. 2001. pp. 453-462.

[18] Thompson, S., *Sampling*. New York. 1992: John Wiley and Sons.

[19] Thompson, S. and G. Seber, *Adaptive Sampling*. 1996, New York: John Wiley and Sons.

[20] Mahoney, M. and P. Chan. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In *Proceedings of RAID*. 2003. pp. 220-237.

[21] MIT Lincoln Lab DARPA Intrusion Detection Datasets. http://www.ll.mit.edu/IST/ideval/data/data_index.htm

[22] Kendall, K., *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*. 1999, Massachusetts Institute Of Technology.

[23] MIT Lincoln Lab 1998 DARPA Intrusion Detection Dataset. http://www.ll.mit.edu/IST/ideval/data/1998/1998_data_index.htm

[24] Cochran, W., Sampling techniques. *New York*, 1977.

[25] Krishnaiah, P. and C. Rao, *Handbook of Statistics 6: Sampling*. 1988: North-Holland.