Hierarchical Clustering and Sampling Techniques for Network Monitoring

S. Sindhuja ME

ABSTRACT: Network monitoring applications are used to monitor network traffic flows. Clustering techniques are used to extract network traffic patterns. Anomaly detection schemes are used to detect network attacks. Hierarchical and partitional clustering schemes are used to analyze network traffic data values. The hierarchical data analysis uses the structure and data values for the clustering process. The patterns are minimized using the summarization techniques.

Keywords: Categorical, cluster feature, itemset, sampling.

I. INTRODUCTION

The hierarchical clustering based network monitoring system uses a distance measure to calculate distance between the hierarchical structures. Numerical, categorical and hierarchical data values are compared using the behavior in order to plan network capacity. As network capacities increase, traffic analysis tools face the problem of scalability due to high packet arrival rates and limited memory. In this paper, we present a hierarchical clustering technique for identifying significant traffic flow patterns. In particular, we present a novel way of exploiting the hierarchical structure of traffic attributes, such as IP addresses, in combination with categorical and numerical attributes. This algorithm addresses the above problems in previous approaches [3] of network traffic analysis as it is a one-pass, fixed memory algorithm. We demonstrate the advantages of our clustering in terms of improved accuracy and significantly reduced computation time in comparison to an earlier approach, on a standard benchmark dataset.
distance measure. The network pattern analysis system uses the data values from the database. All the network traffic information are collected and updated into the database. The traffic analysis is carried out on the stored data values.

The proposed system is designed to perform instant network traffic pattern analysis process. All the network traffic information are captured and transferred to the data analysis tool. The data values are transferred through data streams. TCP data streams are used to transfer network traffic information. The pattern extraction tool analyzes the data values and assigns the relevant data into the hierarchy. Traffic flow, access pattern and service information are extracted from the system. Data sampling techniques are used in high-speed data communications and list of

S.Sindhuja is working as Asst Professor in SNS College of Engineering, Tamilnadu. Emails: sindhugubi@gmail.com

There is a growing need for efficient algorithms to detect important trends and anomalies in network traffic data. For example, network managers need to understand user. A key challenge in clustering multi-dimensional network traffic data is the need to deal with various types of attributes: numerical attributes with real values, categorical attributes with unranked nominal values and attributes with hierarchical structure [6]. For example, byte counts are numerical, protocols are categorical and IP addresses have hierarchical structure. A key issue for these schemes is how to represent a distance function incorporating hierarchical attributes to help find meaningful clusters. We have proposed a hierarchical approach to clustering network traffic data that exploits the hierarchical structure present in real life data. In network traffic a hierarchical relation between two IP addresses can reflect traffic flow to or from a common sub-network. The hierarchical representation of such attributes thus gives more meaning to a general cluster, which can reflect a trend in traffic flows. We propose a common framework to incorporate such hierarchical attributes in the distance function of our clustering algorithm

II RELATED WORK

The problem of identifying network trends has been studied by [3]. One of the earlier works is which studied several sampling Analysis. In [9], a probabilistic model of network traffic is estimated using the Expectation Maximization (EM) technique.

There are several existing tools for network traffic analysis [1]. Some graphically [2] depict the traffic like flow-scan [3], while others provide 'top K reports' like cflowd [3] and flow-tools [4]. A typical graphical report might look at the source IP addresses that are sending the most traffic, or the ports that receive the most connection attempts. A problem with this approach to reporting is that it tells us nothing about sources that only send a small volume of traffic. If these small flows are combined, then they may form a large portion of the overall traffic. Consequently, these trends maybe overlooked, unless we can identify patterns among traffic flows. The graphical tools cannot in general cope well with high dimensions and fail to generalize patterns.

In [6], the authors address the problem of finding patterns in network traffic by proposing a frequent itemset mining algorithm. Their tool, called AutoFocus [5], describes the traffic mix on a network link by using textual reports as well as time series plots. It also produces concise reports that can show general trends in the data. In Section 2.1 we describe frequent itemset mining of network data in more detail. In [1], the authors build on the work of [6] and suggest a technique for traffic anomaly detection based on analyzing correlations of destination IP addresses in outgoing traffic. This address correlation data is modeled

using discrete wavelet transformation for detection of anomalies. In [12], the authors use the combination of a rule based flow header detection and a traffic aggregation based pattern detection algorithm. Although our work shares a common philosophy of traffic aggregation using traffic headers with the techniques mentioned above, our approach differs in the efficiency and expressiveness of our clustering technique.

A. Frequent Itemset Clustering using AutoFocus

AutoFocus identifies significant patterns in traffic flows by using frequent itemset mining. It first creates a report based on unidimensional clusters of network flows and then combines these unidimensional clusters in a lattice to create a traffic report based on multidimensional clusters.

Unidimensional clustering: For each attribute, AutoFocus builds a one-dimension tree by counting frequent itemsets in the network traffic data [6]. This is straightforward for Protocols and Ports. It requires only those values with more than a certain frequency to be tallied. For IP addresses, it builds a tree of counters to reflect the structure of the IP address space. Counters at the leaves of the tree correspond to the original IP addresses that appeared in the traffic. Higher level nodes in the tree correspond to clusters of addresses that have the same common prefix, i.e., addresses with the first l bits in common, where l is the level of the node in the tree. In order to prune the tree, only those nodes having traffic volumes above a threshold are retained.

Multidimensional clustering: For multidimensional clustering, AutoFocus uses unidimensional cluster trees to create an m-dimensional lattice structure. Building the complete lattice would be expensive since it involves all possible combinations among the values of different attributes. Instead, AutoFocus uses certain properties of the lattice structure to avoid brute force enumeration. Nevertheless, AutoFocus still requires multiple passes through the network traffic dataset in order to generate significant multidimensional clusters. An open issue for research is how to find multidimensional clusters in network traffic in a computationally efficient manner. To address this problem, we consider the use of a hierarchical clustering algorithm.

B. Hierarchical Clustering

Our approach to finding multidimensional clusters of network data builds on the BIRCH framework [7], which is a clustering algorithm that uses a *Cluster Feature* (CF) to represent a cluster of records. Instead of individual records, a CF only keeps their sufficient statistics as 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 (CFTree) to summarize the input records.

The tree is built in an agglomerative hierarchical manner. 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 in a recursive manner, due to the additive property of the statistics in the record. Consequently, the clusters in the root of the tree represent the most abstract summary of the dataset. As the CF-tree grows, more nodes are allocated to the tree. An important advantage of the CF-tree structure is that the cluster hierarchy can be maintained in a fixed memory size M by re-clustering the leaf-level CF records if necessary. If P denotes the size of a node in the tree, then it takes only $O\ (B*(1+\log_B M/P))$ comparisons to find the closest leaf node in the tree for a given record [7].

An open issue for using the BIRCH approach to cluster network traffic records is how to cope with numerical, categorical and hierarchical attributes which are used to describe the network traffic. For example, in traffic records, the number of bytes in a flow is represented as an integer, the type of network protocol is represented using a categorical attribute and IP addresses are attributes with a hierarchical structure. This requires a method for representing the summary statistics and calculating distances for clusters based on these types of attributes. We also require a method for extracting significant clusters from the CF-tree in order to generate a concise and informative report on the given network traffic data. In the next section we propose several modifications to the BIRCH clustering approach to address these problems.

III CLUSTERING NETWORK TRAFFIC USING ECHIDNA

The key problem in clustering network traffic data is the representation of various distances for different data types in order to accurately describe the relationships among different records and clusters. We propose a framework that exploits the natural hierarchy of an attribute and combines different distance functions for our clustering algorithm. 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. Our algorithm takes each record and iteratively builds a hierarchical tree of clusters called a CF-Tree. As the tree is built, each record is inserted into the closest cluster using a combined distance function for all attributes (Section 3.1). The radius of a cluster determines if a record should be absorbed into the cluster or if the cluster should be split (Section 3.2). In Section 3.3, the cluster formation process is explained in detail. Once the cluster tree is created, significant nodes are identified in the summarization process (Section 3.4). Next, the cluster tree is further compressed (Section 3.5) to create a concise and meaningful report. All of these require one additional pass over the cluster tree.

A. Distance Functions

When clustering network traffic records we need to consider three kinds of attributes: *numerical*, *categorical* and *hierarchical*. For each type of attribute, we define the representation that we have used for records, clusters and distance functions in each case.

a) Numerical Attributes: A *numerical* attribute is represented by a scalar $x[i] \in R$.

The centroid $\bar{c}[i]$ of a numerical attribute i in cluster **C** having N points is given by

$$\overline{C}[i] = \frac{1}{N}\sum_{j=1}^{N} X_j[i] \qquad - \qquad - \qquad (1)$$

We calculate the distance d_n between the centroids of two clusters $\mathbf{C}_1$ and $\mathbf{C}_2$ by using the Euclidean distance metric ||.||

$$d_n(C_1, C_2) = \left\| \overline{c_1} - \overline{c_2} \right\| = \left[(\overline{c_1} - \overline{c_2})^2 \right]^{\frac{1}{2}} \quad (2)$$

b) Categorical Attributes: In the case of a *categorical* attribute **x**[i] is a vector $\in Z^c$, where *c* is the number of possible values that the *categorical* attribute *i* can take. For a *d*-dimensional *categorical* attribute vector **X**, let the i^{th} attribute **x**[i] be represented as **x**[i] ={a_1, a_2,…,a_c}, where $a_k \in \{0,1\}a$. Any *categorical* attribute of a record can only take one value at a time. If a_k= 1, then a_j = 0 for $\forall j \neq I$ so that, the sum of *ak* in a single record is always 1, i.e.,

$$\sum_{k=1}^{c} a_k = 1, \text{ where } a_k \in x[i]$$

The centroid $\overline{c}$ [i]of a *categorical* attribute *i* in cluster **C** having *N* points is represented by a histogram of the frequencies of the attribute values:

$$\overline{c}[i] = \frac{1}{N} \sum_{j=1}^{N} X_j[i] = \left\{ \frac{A_1}{N}, \frac{A_2}{N}, ..., \frac{A_c}{N} \right\} \quad (3)$$

where, $A_k = \sum_{j=1}^{N} a_{k,j}, x_j[i] = \{a_{1,j}, a_{2,j}, ..., a_{c,j}\}$ and

$$\sum_{k=1}^{c} A_k = N$$

$$d_c(C_1, C_2) = \left\| \overline{c}_1 - \overline{c}_2 \right\| = [\sum_{k=1}^{c} (A_{k,1} - A_{k,2})^2] \quad (4)$$

For example, consider an attribute **x**[i] that represents Color, which has values {Red, Green, Blue}. A value X[i]={1,0,0}represents the color Red. In addition, consider a cluster **C**1 that contains 5 records described by the Color attribute. If there are 3 instances of Red, 2 instances of Green and 0 instances of Blue, then $\overline{c}[i] = \{3,2,0\}$.

c) Hierarchical Attributes: We define a *hierarchical* attribute **H** as a generalization hierarchy in the form of an *L*-level tree applied to a domain of values **H**L which form the leaves of the tree at level *L*.A non-leaf node $h_{l,i} \in H_l$, in the hierarchy represents a generalization of the leaf nodes in the subtree rooted at $h_{l,i}$.

Similarly, IP addresses can be represented as a hierarchical attribute using a 32-level generalization hierarchy (i.e. *L*=32) on the domain of integer IP address values in the range [0,232 -1]. Before describing the IP address hierarchy in detail we need to define the following:

B. Radius Calculation

In order to control the variance of data records within a cluster, we need some measure of the *radius* of a cluster. This gives us a sense of how compact the cluster is. The *radius* for *numerical* and *categorical* attributes can be represented in a straightforward manner as the standard deviation of the attribute values of the records in the cluster. In the case of hierarchical attributes, we need to define a new measure of *radius*. The final radius value of the cluster is simply a linear combination of the individual radius values of different attributes types.

C. Cluster Formation

We follow the general approach of BIRCH [5] for cluster formation, which results in a hierarchical set of clusters in the form of a CF-tree. The leaves of the CF-tree represent the most detailed set of clusters, and the root of the CF-tree represents the most general set of clusters describing the input records.

Cluster formation proceeds by adding each input data record *X* to a cluster C_l in a leaf node of the CF-tree, such that C_l is closest to *X* using the distance measure $D(X,C_l)$, given in Equation 7. If the addition of *X* to C_l would cause the *radius* R_l of the cluster to grow above a threshold *T*, then a new cluster is created. We now summarize the main steps in this process.

Each data record *X*, corresponding to a 6-tuple traffic flow record is compared to the closest cluster starting from the root along a path *P* to a leaf node, where the closest $C = \arg\min_1 \{X, \overline{C_l}\}$ At the leaf node, the data record *X* is inserted into the closest C_l and the *radius* R_l of the updated cluster is calculated. If $R_l > T$, 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 minimum *m*, then *X* is inserted into the node as a new cluster

D. Summarization

Because of the hierarchical nature of the CF-Tree, we can think of the clusters at the index nodes as holding summaries of the leaf nodes. Thus the levels of the CF-Tree form a hierarchical structure with clusters at leaf level, and super-clusters at the index levels. Note that each node corresponds to a cluster *C*, and the CF-entries in the node correspond to the sub-clusters C_1,…,C_l of *C*. This hierarchical structure provides a natural framework to create a summary report of the hierarchical clusters.

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. In order to keep the report concise but meaningful, the summary contains only significant nodes.

IV ONLINE NETWORK MONITORING WITH SAMPLING TECHNIQUE

The proposed system is designed to monitor stream based network traffic analysis. All network traffic information are collected and transferred to the monitoring application. The clustering technique is used to extract network traffic patterns. The anomalies are identified with respect to the cluster information. The system uses a new hierarchical distance measure to compare network transactions. The distance measure compares numerical, categorical and hierarchical data values. The clustering results are hierarchically assigned in a tree view. All the traffic flow information are compared with the distance measure. Traffic information's are assigned in the cluster

based on the distance value. Patterns are extracted with reference to the cluster intervals.

Network monitoring system is divided into four modules. Transaction observer module is designed to collect network traffic information. Clustering module is used to group up the network traffic information. Data sampling module is designed to perform the pattern extraction using the sample data value. Summary analysis module is used to extract summary information about the network.

V EVALUATION

Our aim was to test the accuracy and scalability of our hierarchical traffic summarization algorithm. As a basis for comparison, we have compared the performance of our algorithm to AutoFocus [6]. In order to provide a quantitative comparison of the algorithms, and to test their ability to generalize patterns of usage, we require network data containing known traffic patterns. In the absence of such publicly available data with known patterns, we decided to use a widely accepted synthetic dataset for intrusion detection from the MIT Lincoln Lab. The 1998 DARPA dataset [6] provides traffic files that contain raw traffic flow records with an associated label to indicate whether the record was part of a normal or attack flow.

Testing methodology

We have conducted tests to compare our algorithm with AutoFocus: cluster accuracy, detection accuracy and run-time performance.

Cluster Accuracy: The aim of this test is to check if the clusters contain records that reflect known patterns present in the traffic files. The CF-Tree represents a hierarchical model of the traffic data built without supervision. The objective of this test is to determine how accurately the clustering algorithm can group traffic records from two different populations,

1) $DR = TP/(TP+FN)$.

2) $FPR = FP/(TP+FP)$.

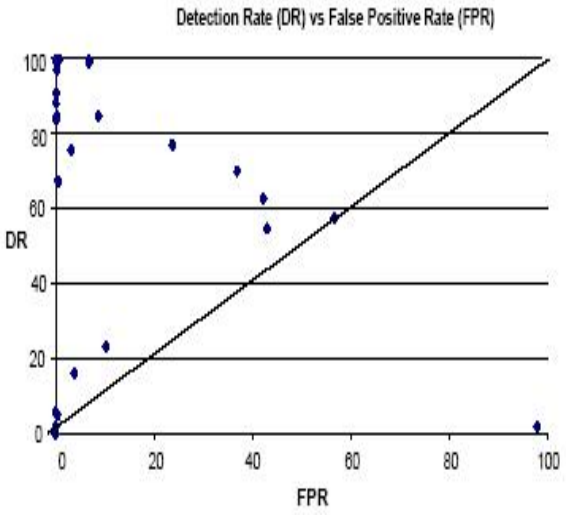

Figure: 1. Cluster accuracy of Echidna in terms of Detection Rate and False Positive Rate on 24 files from weeks 3 through 7 of the 1998 DARPA traces.

Figure 1 shows the cluster accuracy for each network traffic file in terms of the *Detection Rate* and *False Positive Rate*. The diagonal line corresponds to a classifier that chooses classes at random. It can be seen that 23 out of 24 traffic files achieve a cluster accuracy that is above the diagonal. In 16 out of 24 cases, the *DR* exceeds 50%, while the FPR is less than 50% in 24 out 26 cases. Given that we are clustering on a very limited 6-tuple of traffic features, this result supports the reliability of our algorithm in terms of cluster accuracy.

VI PERFORMANCE ANALYSIS

Our aim was to test the accuracy and scalability of our hierarchical traffic summarization algorithm. As a basis for comparison, we have compared the performance of our algorithm to Echidna. In order to provide a quantitative comparison of the algorithms, and to test their ability to generalize patterns of usage, we require network data containing known traffic patterns.

Testing methodology

We have conducted two types of tests to compare our algorithm with Echidna: Detection accuracy and run-time performance.

Detection Accuracy: In particular, our aim is to generate a summary traffic report that identifies important flows in network traffic. In this case, we use the DARPA traces to test whether the reports generated by Echidna or Hierarchical stream clustering (HSC) identify specific attacks that appear in the traces. We assess the comparative utility of each approach in finding significant traffic patterns by comparing the number and type of attacks reported by each algorithm.

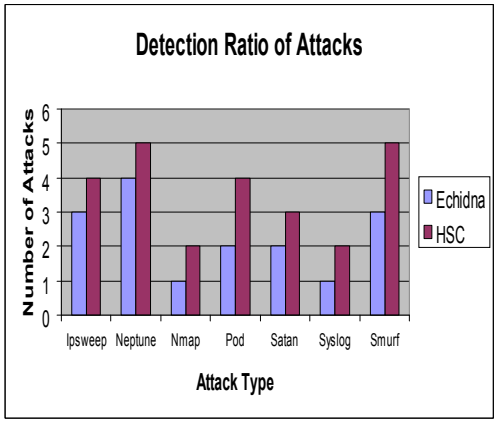

Figure: 2. Detection accuracy of Echidna and HSC by attack types

Table No: 1. Detection accuracy of Echidna and HSC by attack types

	Number of Detected Attacks		
Attack	Total	Echidna	HSC
Ipsweep	5	3	4
Neptune	6	4	5
Nmap	3	1	2
Pod	7	2	4
Satan	4	2	3
Syslog	3	1	2
Smurf	7	3	5

In order to evaluate detection accuracy, we applied Echidna and Hierarchical stream clustering (HSC) to 15 daily network traffic files, which correspond to weeks 3-5 of the DARPA traces. For each file, we identified the number and type of attacks, reported as clusters in the summary reports from Echidna and Hierarchical stream clustering (HSC). We then identified the total number of occurrences of these attack types in the traces. An attack occurrence was detected by Echidna or Hierarchical stream clustering (HSC) if the description of a cluster generated by the system matched the description of the attack in the DARPA attack signature database.

Figure 2 and Table 1 show the number of times each type of attack was detected by Echidna and Hierarchical stream clustering (HSC). Note that HSC was able to produce accurate results in detection process than Echidna. *Run-time performance*: A key motivation for using a hierarchical clustering scheme such as Echidna is to reduce the amount of time required to find the closest cluster for a traffic record. Hierarchical stream clustering (HSC) reduces mush more amount of time to detect traffic flows. This is particularly important when the number of traffic flows is large.

Table No: 2: Comparison of Run-time Performance on DARPA traffic

Echidna	HSC
40	35
90	80
145	125
220	185
410	340

Figure 2 shows the run-time performance of Echidna and HSC with traffic samples varying from approximately $50x10^3$ to $750x10^3$ traffic records. The input file is combined from all days of week 6 of the 1998 DARPA traffic traces. For each sample size, the tests were repeated 3 times to give the average execution time and error bars. The computational complexity of HSC is linear with respect to the number of input traffic flow records.

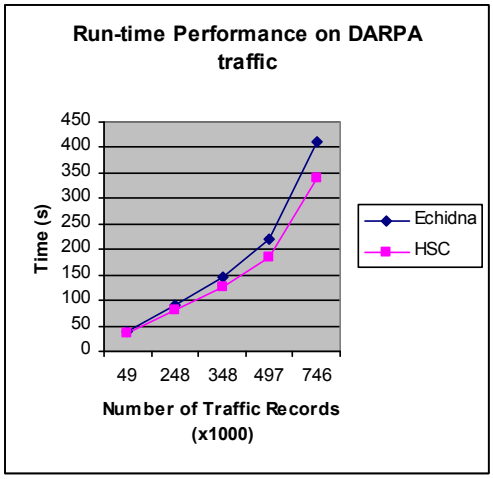

Figure 3 Comparison of Run-time Performance on DARPA traffic

VII CONCLUSION

In this paper, we have presented a clustering scheme called Echidna for generating summary reports of significant traffic flows in network traces. The key contributions of our scheme are the introduction of a new distance measure for hierarchically-structured attributes, such as IP addresses, and a set of heuristics to summarize and compress reports of significant traffic clusters from a hierarchical clustering algorithm. Based on an evaluation using standard benchmark traffic traces, we have demonstrated that our clustering scheme achieves greater cluster accuracy and computational efficiency in comparison to previous work.

REFERENCES

[1] http://www.slac.stanford.edu/xorg/nmtf/nmtf tools.html
[2]http://www.caida.org/projects/internetatlas/viz/viztools.html
[3] http://www.caida.org/tools/measurement/cflow
[4] http://www.splintered.net/sw/flow-tools/
[5] http://www.caida.org/tools/measurement/autofocus
[6] C. Estan, S. Savage and G. Varghese. Automatically Inferring Patterns of Resource Consumption in Network Traffic problem. In Proceedings of SIGCOMM 2003
[7] Abdun Mahmood, Christopher Leckie, Parampalli Udaya "Echidna: Efficient Clustering of Hierarchical Data for Network Traffic Analysis"2008.

S.Sindhuja is working as a Assistant Professor in Dept of CSE, SNSCE. She published 1 international journal and 2 international conferences.