Echidna: Efficient Clustering of Hierarchical Data for Network Traffic Analysis

Abdun Mahmood, Christopher Leckie, Parampalli Udaya

Department of Computer Science and Software Engineering
University of Melbourne, Australia
Email: {abdun,caleckie,udaya}@csse.unimelb.edu.au

Abstract. There is significant interest in the network management community about the need to improve existing techniques for clustering multi-variate network traffic flow records so that we can quickly infer underlying traffic patterns. In this paper we investigate the use of clustering techniques to identify interesting traffic patterns in an efficient manner. We develop a framework to deal with mixed type attributes including numerical, categorical and hierarchical attributes for a one-pass hierarchical clustering algorithm. We demonstrate the improved accuracy and efficiency of our approach in comparison to previous work on clustering network traffic.

1 Introduction

There is a growing need for efficient algorithms to detect important trends and anomalies in network traffic data. 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 scalability problems in previous approaches [5-9] of network traffic analysis as it is a one-pass, fixed memory algorithm.

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. For example, byte counts are numerical, protocols are categorical and IP addresses have hierarchical structure. We have proposed a hierarchical approach to clustering that exploits the hierarchical structure present in network traffic data. In network traffic a hierarchical relation between two IP addresses can reflect traffic flow to or from a common sub-network. We propose a common framework to incorporate such hierarchical attributes in the distance function of our clustering algorithm.

The second contribution of this paper is the use of a single-pass hierarchical clustering technique to address the problems suffered by existing algorithms in terms of their need to make multiple passes through the dataset. In order to keep the size of the reports small we present a number of summarization techniques over the cluster tree.

In the next section we briefly summarize existing research on identifying trends in network traffic. In Section 3 we present our clustering and summarization algorithm

called Echidna. We demonstrate the effectiveness of our approach using an empirical evaluation in Section 4.

2 Related Work

The problem of identifying network trends has been studied by [5-9]. In [2], the authors address the problem of finding patterns in network traffic by proposing a frequent itemset mining algorithm. Their tool, called AutoFocus [1] 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. AutoFocus requires multiple passes through the network traffic dataset in order to generate *significant* multidimensional clusters. To address this inefficiency, we consider the use of a hierarchical clustering algorithm.

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 Fig. 1). 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 1 shows a CF-Tree in fixed memory M with branching factor *B* and leaf node capacity *L*. 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 [3].

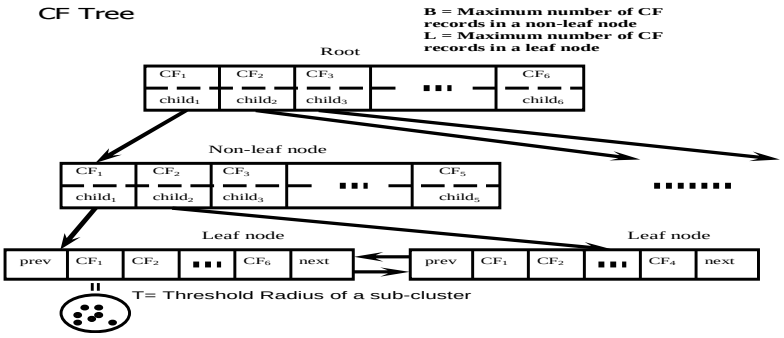

Fig. 1. A Cluster Feature Tree

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. 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.

3 Our Approach to Clustering Network Traffic: Echidna

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. We now describe the distance functions used for each attribute type.

Distance functions: When clustering network traffic records we need to consider three kinds of attributes: *numerical*, *categorical* and *hierarchical*.

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 the mean of the N points. 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.

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 $\mathbf{x}[i] = \{a_1, a_2, \ldots, a_c\}$. The centroid $\bar{\mathbf{c}}[i]$ of a *categorical* attribute *i* in cluster **C** having *N* points is represented by a histogram of the frequencies of the attribute values. The distance between clusters $\mathbf{C}_1$ and $\mathbf{C}_2$ in terms of a single *categorical* attribute is given by the Euclidean distance between the frequency vectors of each attribute.

c) Hierarchical Attributes: A *hierarchical* attribute represents a generalization hierarchy in the form of an *L*-level tree applied to a domain of values at the leaves of the tree. A non-leaf node in the hierarchy is a generalization of the leaf nodes in the subtree rooted at that node.

In a cluster **C**, the centroid for a *hierarchical* attribute that corresponds to an IP address is represented by an IP prefix $\overline{IP} / p$, which is an aggregate of the IP addresses [10] in that cluster. We calculate the distance between two clusters C_1 and C_2 with centroids $\overline{IP_1} / p_1$ and $\overline{IP_2} / p_2$ as $d_h(C_1, C_2) = 32 - p$ if $p > 8$, or 24 if $p \leq 8$, where $p = CommonPrefix(\overline{IP_1} / p_1, \overline{IP_2} / p_2)$. The definitions of *CommonPrefix* and *IP aggregation* can be found in [10].

Intuitively, d_h corresponds to the hierarchical distance from the leaf level of the tree to the most specific generalization of the two centroids. In the case of IP addresses, this corresponds to the size in logarithm (base 2) of the smallest subset that would be required to contain these two clusters. For example, the distance between 128.0.0.252/32 and 128.0.0.254/31 is 2. The distance between two centroids is the squared sum of the distances of each attribute using the appropriate distance function. Note that each attribute is scaled into the range [0,1] so that no single attribute dominates.

Radius Calculation: In order to control the variance of data records within a cluster, we need some measure of the *radius* of a cluster. 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 propose that the *radius* is proportional to the size of the subtree in the *generalization hierarchy* covering the values that appear in the cluster. Consider the case of IP addresses. We keep two variables *minIP* and the *maxIP*,

which correspond to the smallest and the largest IP values present in the cluster. Let *C*[i].*range*=(*minIP*, *maxIP*) denote the range of IP addresses present in attribute *i* of cluster *C*. We can measure this *radius* in terms of the height of the smallest subtree in the *generalization hierarchy* that covers *minIP* and *maxIP*, which can be calculated using *CommonPrefix* as R_h = *(32-CommonPrefix(minIP, maxIP))/32*. The final radius value of the cluster is simply a linear combination of the individual radius values of different attributes types.

Cluster Formation: Following the general approach of BIRCH [3], each cluster C_l is represented by a cluster feature vector that contains sufficient statistics to calculate the centroid $\overline{C_l}$ and *radius* R_l of the cluster. Each data record *X*, corresponding to a 6-tuple traffic flow record, is inserted by comparing *X* to the closest cluster starting from the root along a path *P* to a leaf node. 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. 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.

Summarization: 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 number of records, *Average Intra-Cluster* distance and *Maximum Intra-cluster* distance measures that intuitively pick those nodes that contain a heterogeneous set of clusters.

An index node is considered significant if one of its descendants is significant. A leaf node is significant if it has the following properties:

a) The number of records in the leaf node *C* is above a certain threshold T_r.

b) The *Average Intra-cluster* (AI) distance of the leaf node *C* is above a threshold T_{ai}, where the AI distance of cluster *C* with respect to its *l* sub-clusters $C_1,\ldots,C_l$ is

$$AI(C) = \sqrt{\left(2\sum_{i=1}^{l}\sum_{j=2}^{l} (C_i - C_j)^2 / l(l-1) \right)}$$, where C_i and C_j are sub-clusters of *C*

c) The *Maximum Intra-cluster* (MI) distance of the leaf node *C* is greater than or equal to the AI distance. The MI distance is given by MI(C) = $max\{d(C_i, C_j)\}$, *i*=1,..,*l* and *j*=2,..,*l*

Compression: We require a technique to further compress the number of significant clusters that are included in the final report. We consider a node to be significant if the number of traffic records it contains is greater than T_r. Lemma 1 then gives the upper and lower bound on the number of significant nodes in the tree. The proof of Lemma 1 can be found in [10].

Lemma 1: For a cluster tree of height *h* with τ traffic records and threshold T_r, the size of the report ρ is bounded by $h\frac{\tau}{T_r} \geq \rho \geq 2\frac{\tau}{T_r}, h \geq 2$.

Compression of cluster report: Let $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2,\ldots,\mathbf{C}_h\}$ be the set of clusters in a path *P* from the root to a node in level *h* of the CF-tree. Since a cluster $\mathbf{C}_i$ is represented as

a node in the tree, then $\mathbf{C}_i$ consists of a set of l sub-clusters (l CF entries) at the same level i of the tree $\mathbf{C}_i=\{C_{1,i},C_{2,i},..,C_{l,i}\}$. It follows that

a) $\mathbf{C}_i$ is significant if there exists a $\mathbf{C}_j$ in the path P, such that $\mathbf{C}_j$ is significant, where $i < j$, i.e., $\mathbf{C}_i$ is an ancestor of $\mathbf{C}_j$.
b) Let τ_i and τ_j denote the traffic of $\mathbf{C}_i$ and $\mathbf{C}_j$, then $\tau_i \geq \tau_j$, if $i < j$. In other words, the size of cluster $\mathbf{C}_i$ is greater than or equal to the size of cluster $\mathbf{C}_j$.

Let ρ be the compressed report, and $\mathbf{C}_i$ and $\mathbf{C}_j$ are significant clusters. $\mathbf{C}_i$ is included in ρ if $\tau_i - \tau_j > T_r$, where T_r is the threshold of records. T_r can be expressed as a proportion of the total traffic size, $T_r = r\tau$, where $r = [0,1]$ and T is the total traffic. In other words, a higher level cluster is only included if it reports some traffic not mentioned by its more specific significant sub-clusters.

Complexity: 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 $O(N * B(1+\log_B m))$.

4 Evaluation

Our aim was to test the accuracy and scalability of our hierarchical traffic summarization algorithm. We have compared the accuracy and run-time performance of our algorithm to AutoFocus [2] using 1998 DARPA dataset [4]. Note that the attack/normal labels in this dataset are used for evaluation purposes only, and are not used as part of the cluster formation process.

Detection Accuracy: Our aim is to generate a summary traffic report that identifies important flows in network traffic. In this case, we use the DARPA traces (weeks 3-5) to test whether the reports generated by Echidna or AutoFocus identify specific attacks that appear in the traces. For each file, we identified the number and type of attacks, reported as clusters in the summary reports from Echidna and AutoFocus, and identified the total number of occurrences of these attack types in the traces (see Table 1).

Echidna was able to detect 7 different types of attacks compared to 4 attack types detected by AutoFocus. Moreover, in the case of the ipsweep attack, Echidna detected 3 instances compared to 1 instance detected by AutoFocus. In most cases, the attacks that were detected can be characterized by their influence on the network bandwidth during the time of the attack.

Run-time performance: In order to test the scalability of our algorithm in comparison to AutoFocus, we measured the execution time required by Echidna and AutoFocus for different traffic samples on a time shared dual 2.8GHz Xeon processor machine with 4 GB RAM running SunOS 5.9 (see Fig. 2).

As predicted by the complexity analysis in Section 3, the computational complexity of Echidna is linear with respect to the number of input traffic flow records. Furthermore, Echidna shows a significant reduction in computation time and variance in comparison to AutoFocus.

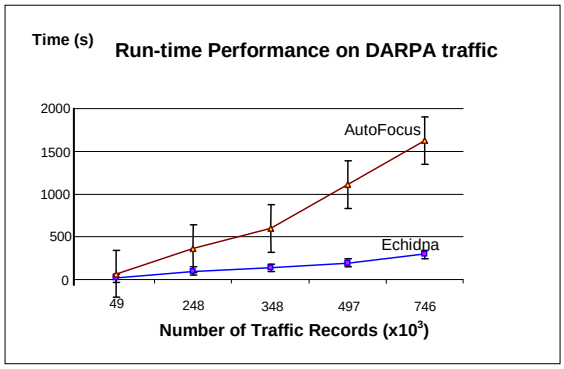

Fig.2. Comparison of Run-time Performances

	Number of Detected Attacks		
Attack	Total	AF	Echidna
ipsweep	5	1	**3**
Neptune	5	4	4
Nmap	2	0	**1**
Pod	7	0	**2**
Satan	4	2	2
Syslog	2	0	**1**
Smurf	7	3	3

Table 1. Detection Accuracy

Conclusion

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. Using standard benchmark traffic traces, we have demonstrated that our clustering scheme achieves greater accuracy and efficiency in comparison to previous work.

References

1. http://www.caida.org/tools/measurement/autofocus/
2. C. Estan, S. Savage. and G. Varghese. Automatically Inferring Patterns of Resource Consumption in Network Traffic problem. In Proceedings of SIGCOMM 2003
3. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pages 103-114, 1996
4. http://www.ll.mit.edu/IST/ideval/data/1998/1998_data_index.html
5. A. Medina, K. Salamatian, N. Taft, I. Matta, and C. Diot. A Two-step Statistical Approach for Inferring Network Traffic Demands (Revises Technical Report BUCS-TR-2003-003).
6. A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural analysis of network traffic flows, In Proceedings of ACM SIGMETRICS, June 2004.
7. A. Lakhina, M. Crovella, and C. Diot. Characterization of Network-Wide Anomalies in Traffic Flows. Technical Report BUCS-2004-020, Boston University, 2004.
8. K. Lan, and J. Heidemann. On the correlation of Internet flow characteristics. Technical Report ISI-TR-574, USC/Information Sciences Institute, July, 2003.
9. K. C. Claffy, G. C. Pluyzos, and H. W. Braun. Applications of Sampling Methodologies to Network Traffic Characterization. In Proceeding of ACM SIGCOMM, 1993.
10. A. Mahmood, C. Leckie, P. Udaya. Echidna: Efficient Clustering of Hierarchical Data for Network Analysis. (http://www.cs.mu.oz.au/~abdun/TR01112005.pdf)