BCSIR

Available online at www.banglajol.info

Bangladesh J. Sci. Ind. Res. **44(3)**, 281-288, 2009

BANGLADESH JOURNAL OF SCIENTIFIC AND INDUSTRIAL RESEARCH

E-mail: bjsir07@gmail.com

Creating Concise Summaries of Network Traffic Using Hierarchical Clustering

A. Mahmood*, C. Leckie, and P. Udaya

Department of Computer Science and Sofitware Engineering, University of Melbourne, Victoria-3010, Australia

Abstract

In today's high speed networks it is becoming increasingly challenging for network managers to understand the nature of the traffic that is carried in thier network. A major problem for traffic analysis in this context is how to extract a concise yet accurate summary of the relevant aggregate traffic flows that are present in network traces. In this paper we present two summarization techniques to minimize the size of the traffic flow report that is generated by a hierarchical cluster analysis tool. By analyzing the accuracy and compaction gain of our approach on a standard benchmark dataset, we demonstrate that our approach achieves more accurate summaries than those of an existing tool that is based on frequent itemset mining.

Key words : Cluster analysis, Internet management, Traffic analysis, Summarization.

Introduction

In today's high speed Internet it is becoming increasingly important for network managers and system administrators to understand the types of traffic flows on their network using traffic analysis and network monitoring techniques. These analysis and monitoring techniques need to be able to process huge volumes of network data that are expensive to store and difficult to analyze. Since such huge volumes of traffic are intractable for manual analysis, often network managers are only interested in a summary of the network events that are taking place at a certain time. For example, instead of looking at each of the possibly thousands of DDoS connections to a web server, we would like to summarize them into just one report entry explaining the attack.

Popular tools for network monitoring and analysis (Network Visualization Tools, Network Monitoring Tools, cflowd: Traffic Flow Analysis Tool and Flow-tools.) display the network traffic parameters visually, as graphs or in the form of "Top-*k*" lists. The methods are mostly ad-hoc, dependent on human expertise and can be inaccurate as the graphs are generated based on a predefined combination of features, which generally cannot sufficiently capture the multi-dimensional distribution of the network traffic.

Consequently, we need an efficient technique to summarize network data into compact reports so that the original network traffic is represented with as little error as possible. Unsupervised cluster analysis is a proven technique to capture unknown trends in data (Han and Kamber, 2006). In our previous work (Mahmood, *et al*, 2006) we have developed a hierarchical clustering technique, Echidna, to cluster network traffic. In this paper we propose two techniques to summarize data in hierarchical clusters: (1) summarization by size of clusters and (2) summarization by distance within clusters. Although the summarization techniques can be used with any hierarchical clustering algorithm, we have used them with our previously developed network traffic clustering algorithm Echidna to group similar network flows based on a combination of numerical, categorical and hierarchical attributes to describe network traffic.

The main contributions of our paper are as follows. We have evaluated our two summarization schemes in the context of identifying network attacks in a popular benchmark dataset. We have also presented our findings on the performance of the summarization techniques on various sizes of datasets and tested against two orthogonal objectives: the accuracy and the size of the report. Finally, we have evaluated our technique against a well known network traffic summarization algorithm, AutoFocus (Estan, *et al*, 2003), and have shown that our technique produces smaller and more accurate reports with less error than AutoFocus.

In Section 2 we discuss related work. In Section 3 we present a brief summary of our Echidna hierarchical clustering technique for network trtaffic data, and briefly discuss the

* Corresponding author: E-mail: {abdun, caleckie, udaya} @cssc.unimelb.edu.au

structure of its cluster tree. In Section 4 we present the summarization techniques we have developed, and compute the bounds on the size of the resulting report. In Section 5 we define our evaluation methodology, and in Section 6 we discuss our evaluation results.

II. Related Work

There are many tools (Network Visualization Tools, Network Monitoring Tools) that generate either text based or visual reports of network traffic. Some tools (Network Visualization Tools) graphically depict the variation of traffic volume, e.g, flow scan. Other tools provide "top K reports" of heaviest usage, such as cflowd and flow-tools. These tools provide visual clues of changes in user behavior at a very high level, for example, by providing a graphical report of IP addresses that are sending the most traffic. A problem with this approach to reporting is that it tells us nothing about sources that send only a small volume of traffic. If these small flows are combined, then they may form a large proportion of the overall traffic. Consequently, these trends may be overlooked unless we can identify patterns among traffic flows. Moreover, graphical tools generally cannot cope well with visualizing traffic using a large number of dimensions, and fail to generalize any underlying patterns.

The concept of summarization is also common in frequent itemset analysis, where various techniques have been developed to generate a compact approximation to a collection of frequent itemsets (Xin, 2007 and Afrati *et al*, 2004) or other multidimensional summaries (Lenz and Shoshani, 1997). In the network traffic domain, Aiello *et al* (2005) proposed *sparse approximations* to port/protocol paris by applying signal processing techniques for time series data. Chandola and Kumar (2005) suggested a bottom-up summarization algorithm where the algorithm incrementally selects the best combination of candidate summaries that minimize the size of the report as well as the information loss. Estan *et al* (2003) developed a summarization technique to compress the size of the output report from their AutoFocus tool, which finds patterns of network usage based on frequent itemsets of network features. First a multi-dimensional tree of traffic clusters is created from a set of input records. This might result in a much larger report than the size of the input. Next, this cluster tree is pruned by selecting only those *significant* clusters whose support is above a given threshold. In the final step a bottom-up search is made to keep only those super-clusters whose support values are significantly more than their significant sub-clusters. A major challenge for both of these approaches is the computational overhead of finding all frequent patterns and their associated support, particularly in terms of the memory required to hold all the transactions and the derived frequent patterns. In this paper, we introduce a new method of traffic summarization based on hierarchical clustering, which is able to extract a very compact and accurate summary in a computationally efficient manner.

III. Cluster Formation

The problem that we address is how to generate a reasonably compact and accurate summary report from a given network traffic trace. The first step of our appproach is to apply hierarchical cluster formation to a traffic trace to identify a detailed set of aggregate traffic flows. The second step of our approach is to extract a compact summary report by applying a summarization algorithm to the clusters found by the first step. In this section we describe the first step of our approach and in the following section we describe the summarization step.

A. Representation of input data

The input data is extracted form the network traffic as 6-tuple records <*SrcIP, DstIP, Protocol, SrcPort, DstPort, bytes*>, where *SrcIP* and *DstIP* are *hierarchical* attributes, *bytes* is *numerical* and the rest are *categorical* attributes. Every record must have a value for each of these attributes.

B. Representation of clusters in the report

The report is created from the input data using a hierarchical clustering technique called Echidna (Mahmood *et al*, 2006). The algorithm takes each record and iteractively builds a hierarchical tree of clusters called a Cluster Feature Tree. Cluster Features (CF) are vectors consisting of the same 6-tuples as the input record, except that the attributes can take aggregated values in the cluster combined from multiple records in a cluster. An example of the aggregated values is shown Table I.

Table I. Example of flow records and the corresponding aggregated flow record in the cluster Feature

Type	Size	IP address	Port	Protocol	Bytes
Record	1	128.250.18.1	80	Tcp	50
Record	1	128.250.18.253	21	Tcp	75
Record	1	128.250.18.128	25	Tcp	100
Cluster	3	128.250.18.0/24	LOW	TCP	225

The key advantage of using Echidna for this problem is its ability to aggregate different types of attributes, such as IP addresses, port numbers, protocol types and byte counts. We now summarize how Echidna deals with each type of attribute. A more detailed description of Echidna's clustering algorithm can be found in (Mahmood *et al*, 2006).

IP address aggregates: We treat IP addresses as attributes with a hierarchical structure, which can be aggregated based on their common prefix as follwos.

Definition 1. CommonPrefix and AggregateIP

CommonPrefix: *The commonPrefix (IP_1, IP_2)* is the longest prefix *p* such that $IP_1/p=IP_2/p$.

AggregateIP: We define *AggregateIP(IP_1, IP_2)* of two IP addresses IP_1, and IP_2 as *IP/p*, such that *p = CommonPrefix (IP_1, IP_2)*, and the least significant (32-*p*) bits of IP are set to zero. Note that it is trivial to extend this definition to a set of IP addresses. For example. *AggregateIP* (203.190.32. 127,203.190.32.128) = 203.190.32.0/24. Using this definition, we can define a distance function for hierarchical attributes.

Port aggregates: Although there are 2^{16} ports, only a few of them are most frequently used and can help to identify which application is using those ports. Ports in the range of [0, 1023] are referred to as *Well Known* ports. Examples include port 21 used by the File Transfer Protocol (FTP), port 23 used by the Telnet protocol, and port 80 used by HTTP web browsers. These ports require special administrative privileges to be used by server applications and are sometimes known as *Low* ports. Ports in the range of [1024, 65535] are known as Low ports and are used by client side applications, different unregistered server applications and generally by applications which change their port addresses dynamically. As the number of Internet applications grow there is a need for more server ports beyond the first 1024. To this end ports in the range 1024-4951 are also known as *Registered* ports but such a distinction will not be made here. In the CF vector all Low Ports are represented by LOW and *high* ports are represented by HIGH. This is often useful to distinguish the server and client nature of the traffic requests. For example, most Peer-to-Peer (P2P) applications use High ports. Thus, a high number of flows belonging to TCP port 4662 may indicate the file sharing application *eMule*, and TCP port 1214 is used by the well known application *Kazaa*.

Protocol aggregates: Most applications on the Internet use the TCP protocol, followed by UDP and ICMP. TCP and UDP are transport protocols and ICMP is a network protocol. There are many protocols in each of these layers, for example, the Datagram Congestion Control Protocol (*DCCP*) and *NetBEUI* in the transport layer, and *IGMP* and *IPsec* in the network layer. However, they are very infrequent compared to TCP, UDP and ICMP and are therefore combined as OTHER in the aggregates.

Byte aggregates: The byte field shows the total number of bytes that are transferred in the packets belonging to the flow. A very high number may indicate a bandwidth intensive application and warn the network administrator of possible misuse.

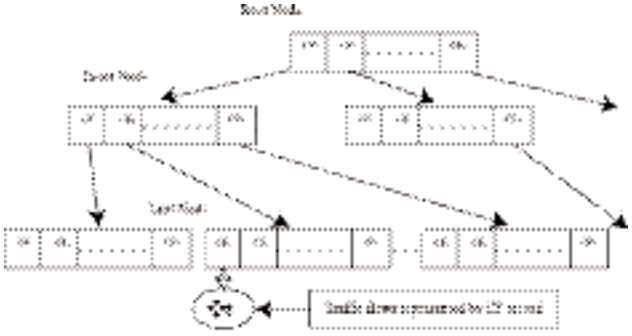

Fig. 1 Cluster Feature Tree

C. Building a cluster tree

Echidna constructs a cluster feature tree 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. Fig. 1 shows a CF-Tree with branching factor *B* and leaf node capacity *L*.

IV. Summarization Algorithms

In this section we present two techniques to generate summarized reports from a cluster feature tree (CF-Tree, see Fig. 1). 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 hierchical structure with clusters at the 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_1 of *C*. This hierarchical structure provides a natural framework to create a summary report of the hierarchical clusters.

Choosing significant clusters: Depending on the size of the tree there can be thousands of leaf level clusters, which very in the number of records they contain. In order to produce a manageable report, at first, only those clusters that contain records above a certain threshold are considered significant. We define *significant* nodes in terms of the number of records in a cluster as follows.

a) A leaf node is significant if the number of records in the leaf node C is above a certain threshold T_r.

b) An index node is considered significant if one of its descendants is significant or the sum of records in the index node is greater than T_r.

A cluster report consisting of all the significant nodes in a CF-Tree can be quite large. Consequently, we require a technique to further reduce the number of clusters that are included in the final report. There are two ways that a report containing significant clusters can be reduced - summarizing by the size of clusters, and summarizing by the homogeneity of clusters. We describe cach of these methods in turn.

A. Summarization by the size of clusters

A significant ancestor cluster is reported during summarizatin only when it contains a significant number of additional records beyond the sum of its significant children. This method was proposed by Estan *et al* in their work on AutoFocus, which is different from our implementation of a balanced CF-tree. We show in Lemma 1 that by using this technique the size of the summarized report depends on the tree height and total traffic.

Lemma 1: For a cluster tree with τ traffic records and threshold T_r, the size of the report p is bounded by

$h \frac{\tau}{T_r} \geq \rho \geq 2 \frac{\tau}{T_r}$, $h \geq 2$, where $h = 1 + \log_B (M/P)$ is

the height of the height-balanced CF-tree, M is the fixed amount of available memory and P is the node size.

Proof: Since a significant node cannot have less than T_r records, there can be a maximum of τ/T_r significant nodes at the leaf level of a CF-Tree having traffic τ. However, since every parent of a significant node in a path from the leaf to the root is also a significant node, it follows there can be a maximum of $h(\tau/T_r)$ significant nodes in the tree. On the other hand, if τ/T_r significant leaf nodes are distributed in a way to minimize the number of significant index nodes, then there will be a minimum of $\tau/T_r + \tau/T_r B + r/T_r B^2 + ... + \tau/T_r B^{h-l} < 2\tau/T_r$ significant nodes in the tree. Hence, the upper limit is $h(\tau/T_r)$ and the lower limit is $2\tau/T_r$.

If we use a lower value of T_r to ensure that clusters with less traffic are reported, then the report size can grow significantly. We can report fewer clusters by observing that not all parent nodes of a significant child node convey additinol information beyond what is contained in the child node. First we formalize our observations about significant cluster nodes and their relationship with significant children.

Let $C = \{C_1, C_2 ....., C_h\}$ be the set of clusters in a path P from the root to a node at level h of the CF-tree. Since a cluster C_i, is represented as a node in the tree, then C_i, consists of a set of l sub-clusters (l CF entries) at the same level i of the tree $C_i = \{C_{l,i}, C_{2,i}, ..., C_{l,i}\}$. It follows that C_i, is significant if there exists a C_j in the path ρ , such that C_j is significant, where $i<j$, i.e., C_i in an ancestor of C_j.

Let τ_i and τ_j denote the traffic of C_i and C_j, then $\tau_i \geq \tau_j$, if $i < j$. In other words, the size of cluster C_i is greater than or equal to the size of cluster C_j.

Let p be the summarized report, and C_i, and C_j be significant clusters. C_i is included in p if $\tau_i - \tau_j > T_r$, where T_r is the theshold of records. T_r can be expressed as a proportion of the total traffic size, $T_r = r\tau$ were $r = [0, 1]$ and τ 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.

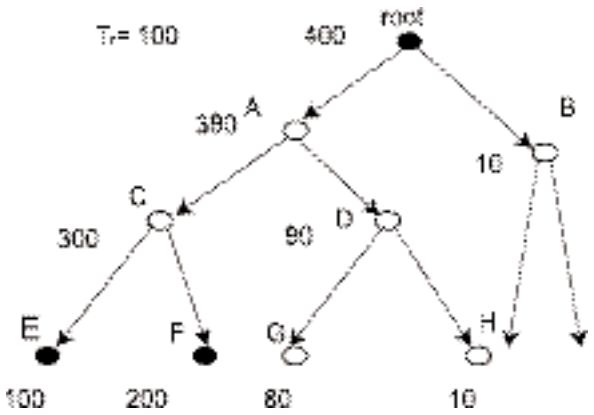

Fig. 2. Example of a CF-tree having Significant clusters

Fig. 2 shows an example of a CF-tree, where each node is annotated with the number of records it contains. Clusters E, F and the root are retained in the summarized report for T_r = 100. We refer to E and F as reported clusters. Clusters B, D, G and H are not significant (<100). Among the significant

non-leaf nodes (root, A, and C) only the root captures significantly more (>100) traffic records than the sum of its descendent nodes that have been reported, i.e, E and F. Thus only the shaded nodes (root, E and F) would be included as reported nodes in the summarized report.

B. Summarization by the homogeneity of clusters

In a hierarchical representation, a parent cluster is actually a combination of its children's clusters, While traversing the tree, we may find certain parent clusters to contain children clusters whose centroids are the same or lie very close to each other. These may be clusters of a similar nature or class and we call them *homogeneous* clusters. In this case there is little benefit to traverse even further below these children because the top level class will remain the same. Therefore, only the parent cluster is picked during summarization and the children are ignored. On the other hand, some parent clusters may have two centroids which might reflect bimodality of a cluster, as illustrated in Fig. 3.

If the children clusters represent distinct information by themselves, rather than when combined at their parent level, it is preferable to report those children during summarization instead of their parents.

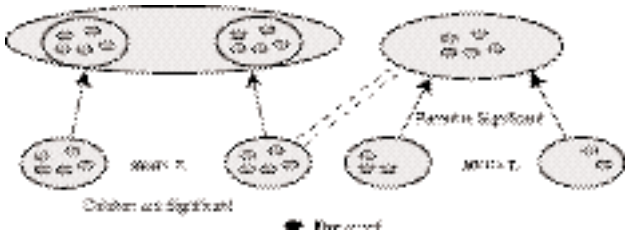

Fig. 3. Example of significant and not significant clusters based on MI/AI distance

In order to test if a group of clusters is homogeneous, during summarization, we check if the ratio $MI/AI > T_c$, where T_c is a threshold value, and MI and AI are defined as follows.

Definition 3: The *Average Intra-cluster* (AI) distance of the node *C*, where the AI distance of cluster *C* is defined with respect to its *l* sub-clusters C_1,.....,C_1, is

$$AI(C) = \sqrt{\left| 2\sum_{i=1}^{l}\sum_{j=i+1}^{l} D(C_i, C_j)^2 / l(l-1) \right|} \quad (5)$$

where *D* is a distance measure between clusters, as defined in detail in (Mahmood, 2006). A cluster with a small AI distance implies that elements within the cluster are homogeneous, e.g., in Euclidean space they lie close to each other.

Definition 4: The *Maximum Intra cluster* (MI) distance of a node *C* is given by MI(C) = *max* $\{D(C_i, C_j)\}$, $i = 1,...,l$ and $j = l, ...l$. The MI distance gives a measure of the maximum width of the cluster. This is useful in addition to AI distance in understanding the homogeneity of a cluster.

In general, the MI distance is always greater than or equal to the AI distance for a cluster. However, using Fig. 3 as an example, a hierarchical cluster will tend to have a smaller MI/AI ratio than would be the case for a homogeneous cluster. Intuitively, if there is a homogeneous cluster that has several significant sub-clusters, then we can remove the sub-clusters without any significant loss of information. in practice we have found through empirical testing that a threshold $T_c = 1.3$ provides a reasonable boundary between the MI/AI ratios of homogeneous and non-homogeneous clusters during the summarization process of the network traffic we have studied. In the next section we discuss ways of validating the usefulness of the summarized report in terms of conciseness and correctness with respect to the input data.

V. Experimental Evaluation Methodology

There are two important objectives of our exprimental evaluation. We want our summary report to be concise, yet we would like it to be an accurate representation of the input. It is obvious that there is a tradeoff between conciseness and correctness of a report. For example, a report consisting of all the input records is correct, i.e, there is no information loss in the report. However, since the size of the report is equal to the size of the input there is no gain in compaction. The report can be made smaller by aggregating some of the entries together into a single entry. However, this can introduce a loss of information. In our evaluation we compare our summarization technique against AutoFocus (Estan, *et al*, 2003) a tool (described in Section II) to create summaries of network traffic.

A. Compaction Gain (CG)

A report is concise if it has fewer entries than the original flow traffic. More precisely, we can measure the *compaction gain* (CG) of a report as the ratio τ/r, where *r* is the number of entries in the report and τ is the total number of entries or records in the input flow traffic. For example in Table 1, $r = 1$ and $\tau = 3$, so $CG = 3$.

B. Information Loss (IL)

Although a report may be concise it may not be completely correct. Consider a report consisting of single aggregated entry that represents all possible values in each of its fields (IP address = , * Protocol = * and Port =*, were* represents all possible values). This report is trivially most accurate but it has zero information content. The correctness of a report can be measured by the error introduced by each entry in the report because it is not represented correctly by the input records. Because of the representation of a report entry, an entry may match partially with an input record, match completely or may not match at all.

We define the *information loss* (IL) of a report as the normalized sum of the distances from each input record to its closest report entry. This penalizes entries that are more general in representation. We can formulate IL as follows.

$$IL = \sum_{i=1}^{r} D(r_i, closest(r_i, S))$$

where r_i, is the i^{th} input record, S is the summary report to be evaluated, and *closest* (r_i, S) is the closest entry to r_i, in S.

C. Key parameters

Report size: The summary is represented as a report of size r containing the summarized clusters. By traversing down the tree we pick significant clusters based on the techniques described in Section IV. The report contains the top r significant clusters. Note that we refer to compaction gain τ/r rather than report size in our results.

Tolerance: There is a possibility that there could be significant clusters that are very similar to each other, for example, varying only by one or two least significant bits of the IP address. In order to avoid creating duplicate or very similar report entries we allow a small tolerance $\delta \in [0,1]$ that represents the difference in the Euclidean distance between a record and a cluster in the report that we are willing to tolerate. This helps us to reduce the size of the summary reprot, when the report contains many similar entries. Based on empirical testing, we found that tolerance values in the range of [0, 0.05] gave the best results.

VI. Evaluation Results

In this section we discuss the results obtained in our evaluation of the summarization approaches of Echidna in comparison to AutoFocus. We have used the MIT Lincoln Laboratory 1998 DARPA Intrusion Detection dataset as a basis of our evaluation. This dataset contains 5-tuples: source IP, destination IP, protocol, source port and destination port and also has labels indicating whether a given flow record is an attack or normal, which is useful for finding the accuracy of a summary report. We have used these labels to test the accuracy of the summarization schemes in Echidna in terms of their ability to distinguish normal or attack traffic (Section VI.A).

A. Classification accuracy

In this experiment we have trained the Echidna clustering algorithm with data from the 1998 DARPA dataset to build a cluster tree from the network traffic records. Then the summarization techniques are used to create a summary report from the cluster tree. Note that a summarized entry is in fact a cluster centroid that contains important statistics of the individual records that belong to the cluster. Included in the statistics is the number of attack and normal records represented by the cluster. For each input record that matches this cluster entry in the summary, the class label of the record is matched against the majority label of the cluster and a confusion matrix is calculated according to Table II.

Table II. Standard confusion metrics for evaluation of attack classification

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

By varying parameters of the clustering algorithm we are able to obtain a Receiver Operating Characteristics (ROC) curve (Recall vs. False Positive rate) for each of the summarization techniques. The area under the ROC curves indicates how well each technique has been able to identify the attack classes for various report sizes, Fig. 4 illustrates how the area under the curve varies with increasing report sizes. As expected, the classification accuracy increases as the report includes more entries. We can see that summarization by homogeneity performs better than summarization by cluster size across all report sizes. As a consequence, in the following subsections, when we show the results for Echidna, we are referring to Echidna using summarization by homogeneity.

B. Information loss vs. compaction gain

Fig. 5 shows how information loss varies with compaction gain for various sizes of input data files. The legend shows the size of the input file in terms of the number of records it contains.

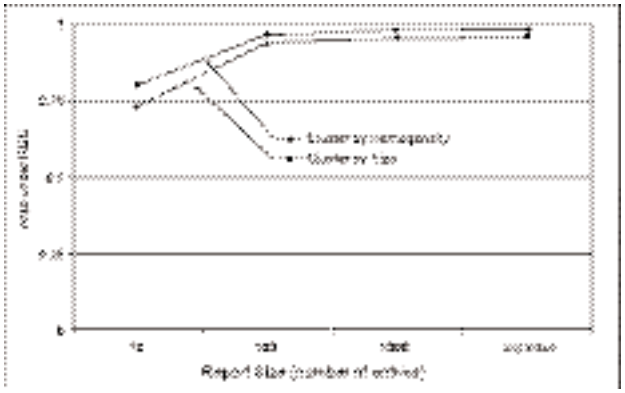

Fig. 4. Summarization accuracy for identifying attacks in summarized reports

Fig 5 shows that, in general for a report of fixed size, incresing the input size results in reduced information loss but higher compaction gain. It appears that for short reports (i.e., high compaction gain), the summaries generated on the larger files produced better generalizations that the summaries generated on the shorter files. An issue for further research is to clarify the reasons for this.

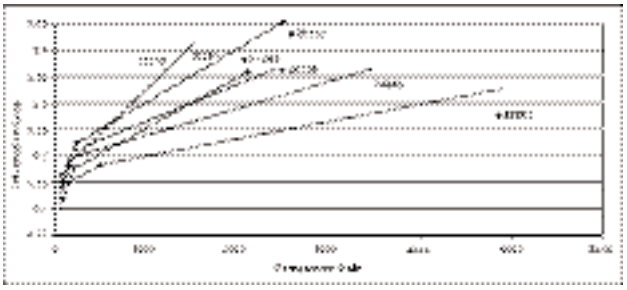

Fig. 5. Information Loss vs. Compaction Gain for varying input sizes of input files

C. Comparison between Echidna and AutoFocus

In order to compare our summarization technique against Auto Focus, we ran AutoFocus for various threshold values to obtain different sizes of reports (note that the size of the report cannot be controlled directly in AutoFocus). Fig. 6 shows that the information loss of the summary produced by Echidna is consistently less than that of AutoFocus for similar compaction gains.

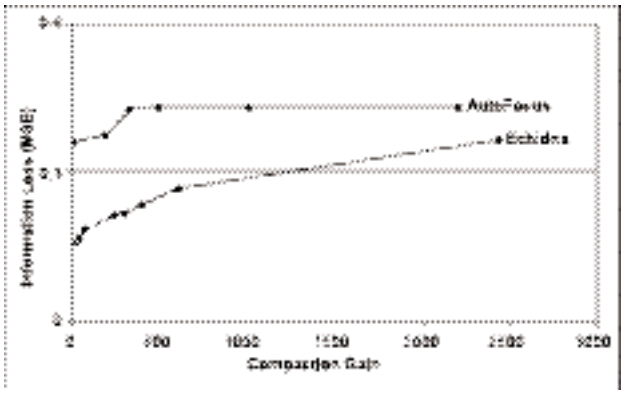

Fig. 6 Comparison of Echidna and AutoFocus summarization techniques

VII. Conclusion

In this paper, we have presented two summarization strategies for generating compact yet accurate summaries of the traffic patterns that are observed in a network traffic trace. Our summarization strategies are able to eliminate redundancies in the hierarchical cluster tree that is generated by our Echidna hierarchical clustering scheme. Based on an evaluation using a standard benchmark dataset, we have found that Echidna can create summaries with lower information loss for the same level of compaction gain when compared to an existing approach that is based on frequent itemset mining.

These results demonstrate that the Echidna summarization scheme can help network managers to address the challenge of how to extract and report meaningful and concise summaries of the activity of traffic in high capacity networks.

Acknowledgement

We take this opportunity to recognize the effort of DARPA and the MIT Lincoln Lab for disseminating, probably the only publicly available, labeled intrusion detection dataset. We also tank Christian Estan for making available his implementation of the AutoFocus system.

References

Afrati F., Gionis A. and Mannila H. (2004) Approximating a collection of frequent sets. In Proceedings of the 2004 ACM SIGKDD International conference on

Knowledge discovery and data mining, ACM Press New Yourk. NY. USA.

Aiello W. (2005) Sparse Approximations for High Fidelity Compression of Network Traffic Data. In Proceedings of ACM/USENIX Internet Masurement Conference (IMC). 2005.

Cflowd: Traffic Flow Analysis Tool.http://www.caida.org/tools/measurement/cflowd/

Chandola V. and Kumar V. (2005) Summarization-Compressing Data into an Informative Representation. In 5th International Conference on Data Mining (ICDM).

Estan C., Savage S. and Varghese G. (2003)Automatically inferring pattens of resource consumption in network traffic. In Proceedings of the 2003 conference on Applications, technologies, architectures. and protocols for computer communictions, 2003: p. 137-148.

Flow-tools. http://www.splintered. net/sw/flow-tools/

Han J. and Kamber M. (2006) Data Mining: Concepts and Techniques, Morgan Kaufmann.

Lenz H. and Shoshani A. (1997) Summarizability in OLAP and Statistical Data Bases. In Proceedings of the Ninth International Conference on Scientific and Statistical Database Management. IEEE Computer Society Washington, DC, USA.

Mahmood A., Leckie C. and Udaya. P. (2006) Echidna: Efficient Clustering of Hierarchical Data for network Traffic Analysis. In Procedings of Networking Springer.

Network Visulization Tools.http://www.caida.org/funding/internetatlas/viz/viztools. html

Network Monitoring Tools. http:www.slac.stanford.edu/xorg/nmtf/rmf-nools.html

Xin D. (2007) On compressing frequent patterns. *Data & Knwledge Engineering*, **60**(1): p.29.

Received : August 13, 2008;
Accepted : Febuary 11, 2009