Traffic Classification Using Clustering Algorithms

Jeffrey Erman, Martin Arlitt, Anirban Mahanti

University of Calgary, 2500 University Drive NW, Calgary, AB, Canada
{erman, arlitt, mahanti}@cpsc.ucalgary.ca

ABSTRACT

Classification of network traffic using port-based or payload-based analysis is becoming increasingly difficult with many peer-to-peer (P2P) applications using dynamic port numbers, masquerading techniques, and encryption to avoid detection. An alternative approach is to classify traffic by exploiting the distinctive characteristics of applications when they communicate on a network. We pursue this latter approach and demonstrate how cluster analysis can be used to effectively identify groups of traffic that are similar using only transport layer statistics. Our work considers two unsupervised clustering algorithms, namely K-Means and DBSCAN, that have previously not been used for network traffic classification. We evaluate these two algorithms and compare them to the previously used AutoClass algorithm, using empirical Internet traces. The experimental results show that both K-Means and DBSCAN work very well and much more quickly then AutoClass. Our results indicate that although DBSCAN has lower accuracy compared to K-Means and AutoClass, DBSCAN produces better clusters.

Categories and Subject Descriptors

I.5.4 [**Computing Methodologies**]: Pattern Recognition—*Applications*

General Terms

Algorithms, classification

Keywords

machine learning, unsupervised clustering

1. INTRODUCTION

Accurate identification and categorization of network traffic according to application type is an important element of many network management tasks such as flow prioritization, traffic shaping/policing, and diagnostic monitoring. For example, a network operator may want to identify and throttle (or block) traffic from peer-to-peer (P2P) file sharing applications to manage its bandwidth budget and to ensure good performance of business critical applications. Similar to network management tasks, many network engineering problems such as workload characterization and modelling, capacity planning, and route provisioning also benefit from accurate identification of network traffic. In this paper, we present preliminary results from our experience with using a machine learning approach called *clustering* for the network traffic identification problem. In the remainder of this section, we motivate why clustering is useful, discuss the specific contributions of this paper, and outline our ongoing work.

The classical approach to traffic classification relies on mapping applications to well-known port numbers and has been very successful in the past. To avoid detection by this method, P2P applications began using dynamic port numbers, and also started disguising themselves by using port numbers for commonly used protocols such as HTTP and FTP. Many recent studies confirm that port-based identification of network traffic is ineffective [8, 15].

To address the aforementioned drawbacks of port-based classification, several payload-based analysis techniques have been proposed [3, 6, 9, 11, 15]. In this approach, packet payloads are analyzed to determine whether they contain characteristic signatures of known applications. Studies show that these approaches work very well for the current Internet traffic including P2P traffic. In fact, some commercial packet shaping tools have started using these techniques. However, P2P applications such as BitTorrent are beginning to elude this technique by using obfuscation methods such as plain-text ciphers, variable-length padding, and/or encryption. In addition, there are some other disadvantages. First, these techniques only identify traffic for which signatures are available and are unable to classify any other traffic. Second, these techniques typically require increased processing and storage capacity.

The limitations of port-based and payload-based analysis have motivated use of transport layer statistics for traffic classification [8, 10, 12, 14, 17]. These classification techniques rely on the fact that different applications typically have distinct behaviour patterns when communicating on a network. For instance, a large file transfer using FTP would have a longer connection duration and larger average packet size than an instant messaging client sending short occasional messages to other clients. Similarly, some P2P applications such as BitTorrent [1] can be distinguished from FTP data transfers because these P2P connections typically are persistent and send data bidirectionally; FTP data transfer connections are non-persistent and send data only unidirectionally. Transport layer statistics such as the total number of packets sent, the ratio of the bytes sent in each direction, the duration of the connection, and the average size of the packets characterize these behaviours.

In this paper, we explore the use of a machine learning approach called *clustering* for classifying traffic using only transport layer

[1] http://www.bittorrent.org/protocol.html

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

statistics. Cluster analysis is one of the most prominent methods for identifying classes amongst a group of objects, and has been used as a tool in many fields such as biology, finance, and computer science. Recent work by McGregor *et al.* [10] and Zander *et al.* [17] show that cluster analysis has the ability to group Internet traffic using only transport layer characteristics. In this paper, we confirm their observations by evaluating two clustering algorithms, namely K-Means [7] and DBSCAN [5], that to the best of our knowledge have not been previously applied to this problem. In addition, as a baseline, we present results from the previously considered AutoClass [1] algorithm [10, 17].

The algorithms evaluated in this paper use an *unsupervised* learning mechanism, wherein unlabelled training data is grouped based on similarity. This ability to group unlabelled training data is advantageous and offers some practical benefits over learning approaches that require labelled training data (discussed in Section 2). Although the selected algorithms use an *unsupervised* learning mechanism, each of these algorithms, however, is based on different clustering principles. The K-Means clustering algorithm is a partition-based algorithm [7], the DBSCAN algorithm is a density-based algorithm [5], and the AutoClass algorithm is a probabilistic model-based algorithm [1]. One reason in particular why K-Means and DBSCAN algorithms were chosen is that they are much faster at clustering data than the previously used AutoClass algorithm.

We evaluate the algorithms using two empirical traces: a well-known publicly available Internet traffic trace from the University of Auckland, and a recent trace we collected from the University of Calgary's Internet connection. The algorithms are compared based on their ability to generate clusters that have a high predictive power of a single application. We show that clustering works for a variety of different applications, including Web, P2P file-sharing, and file transfer with the AutoClass and K-Means algorithm's accuracy exceeding 85% in our results and DBSCAN achieving an accuracy of 75%. Furthermore, we analyze the number of clusters and the number of objects in each of the clusters produced by the different algorithms. In general, the ability of an algorithm to group objects into a few "good" clusters is particularly useful in reducing the amount of processing required to label the clusters. We show that while DBSCAN has a lower overall accuracy the clusters it forms are the most accurate. Additionally, we find that by looking at only a few of DBSCAN's clusters one could identify a significant portion of the connections.

Ours is a work-in-progress. Preliminary results indicate that clustering is indeed a useful technique for traffic identification. Our goal is to build an efficient and accurate classification tool using clustering techniques as the building block. Such a clustering tool would consist of two stages: a model building stage and a classification stage. In the first stage, an unsupervised clustering algorithm clusters training data. This produces a set of clusters that are then labelled to become our classification model. In the second stage, this model is used to develop a classifier that has the ability to label both online and offline network traffic. We note that offline classification is relatively easier compared to online classification, as flow statistics needed by the clustering algorithm may be easily obtained in the former case; the latter requires use of estimation techniques for flow statistics. We should also note that this approach is not a "panacea" for the traffic classification problem. While the model building phase does automatically generate clusters, we still need to use other techniques to label the clusters (e.g., payload analysis, manual classification, port-based analysis, or a combination thereof). This task is manageable because the model would typically be built using small data sets.

We believe that in order to build an accurate classifier, a good classification model must be used. In this paper, we focused on the model building step. Specifically, we investigate which clustering algorithm generates the best model. We are currently investigating building efficient classifiers for K-Means and DBSCAN and testing the classification accuracy of the algorithms. We are also investigating how often the models should be retrained (e.g., on a daily, weekly, or monthly basis).

The remainder of this paper is arranged as follows. The different Internet traffic classification methods including those using cluster analysis are reviewed in Section 2. Section 3 outlines the theory and methods employed by the clustering algorithms studied in this paper. Section 4 and Section 5 present our methodology and outline our experimental results, respectively. Section 6 discusses the experimental results. Section 7 presents our conclusions.

2. BACKGROUND

Several techniques use transport layer information to address the problems associated with payload-based analysis and the diminishing effectiveness of port-based identification. McGregor *et al.* hypothesize the ability of using cluster analysis to group flows using transport layer attributes [10]. The authors, however, do not evaluate the accuracy of the classification as well as which flow attributes produce the best results. Zander *et al.* extend this work by using another Expectation Maximization (EM) algorithm [2] called AutoClass [1] and analyze the best set of attributes to use [17]. Both [10] and [17] only test Bayesian clustering techniques implemented by an EM algorithm. The EM algorithm has a slow learning time. This paper evaluates clustering algorithms that are different and faster than the EM algorithm used in previous work.

Some non-clustering techniques also use transport layer statistics to classify traffic [8, 9, 12, 14]. Roughan *et al.* use nearest neighbor and linear discriminate analysis [14]. The connection durations and average packet size are used for classifying traffic into four distinct classes. This approach has some limitations in that the analysis from these two statistics may not be enough to classify all applications classes.

Karagiannis *et al.* propose a technique that uses the unique behaviors of P2P applications when they are transferring data or making connections to identify this traffic [8]. Their results show that this approach is comparable with that of payload-based identification in terms of accuracy. More recently, Karagiannis *et al.* developed another method that uses the social, functional, and application behaviors to identify all types of traffic [9]. These approaches focus on higher level behaviours such as the number of concurrent connections to an IP address and does not use the transport layer characteristics of single connection that we utilize in this paper.

In [12], Moore *et al.* use a *supervised* machine learning algorithm called Naïve Bayes as a classifier. Moore *et al.* show that the Naïve Bayes approach has a high accuracy classifying traffic. Supervised learning requires the training data to be labelled before the model is built. We believe that an unsupervised clustering approach offers some advantages over supervised learning approaches. One of the main benefits is that new applications can be identified by examining the connections that are grouped to form a new cluster. The supervised approach can not discover new applications and can only classify traffic for which it has labelled training data. Another advantage occurs when the connections are being labelled. Due to the high accuracy of our clusters, only a few of the connections need to be identified in order to label the cluster with a high degree of confidence. Also consider the case where the data set being clustered contains encrypted P2P connections or other types of encrypted traffic. These connections would not be labelled using payload-based classification. These connections would, therefore,

be excluded from the supervised learning approach which can only use labelled training data as input. This could reduce the supervised approach's accuracy. However, the unsupervised clustering approach does not have this limitation. It might place the encrypted P2P traffic into a cluster with other unencrypted P2P traffic. By looking at the connections in the cluster, an analyst may be able to see similarities between unencrypted P2P traffic and the encrypted traffic and conclude that it may be P2P traffic.

3. CLUSTERING ALGORITHMS

This section reviews the clustering algorithms, namely K-Means, DBSCAN, and AutoClass, considered in this work. The K-Means algorithm produces clusters that are spherical in shape whereas the DBSCAN algorithm has the ability to produce clusters that are non-spherical. The different cluster shapes that DBSCAN is capable of finding may allow for a better set of clusters to be found that minimize the amount of analysis required. The AutoClass algorithm uses a Bayesian approach and can automatically determine the number of clusters. Additionally, it performs soft clustering wherein objects are assigned to multiple clusters fractionally.

The Cluster 3.0 [4] software suite is used to obtain the results for K-Means clustering. The DBSCAN results are obtained the WEKA software suite [16]. The AutoClass results are obtained using an implementation provided by [1].

In order for the clustering of the connections to occur, a similarity (or distance) measurement must be established first. While various similarity measurements exist, Euclidean distance is one of the most commonly used metrics for clustering problems [7, 16]. With Euclidean distance, a small distance between two objects implies a strong similarity whereas a large distance implies a low similarity. In an n-dimensional space of features, Euclidean distance can be calculated between objects x and y as follows:

$$dist\,(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{1}$$

with n being the number of features in each object. The algorithms in this paper all use the Euclidean distance as their similarity measurement. The objects in this case will be connections and the features are the connection's transport layer statistics.

3.1 K-Means Clustering

There are a variety of partition-based clustering algorithms available [7]. The K-Means algorithm is selected because it is one of the quickest and most simple. The K-Means algorithm partitions objects in a data set into a fixed number of K disjoint subsets. For each cluster, the partitioning algorithm maximizes the homogeneity within the cluster by minimizing the square-error. The formula for the square error is:

$$E = \sum_{i=1}^{K}\sum_{j=1}^{n} |dist(x_j, c_i)|^2 . \tag{2}$$

The square error is calculated as the distance squared between each object x and the centre (or mean) of its cluster. Object c represents the respective centre of each cluster.

The square error is minimized by K-Means using the following algorithm. The centers of the K clusters are initially chosen randomly from within the subspace. The objects in the data set are then partitioned into the nearest cluster. K-Means iteratively computes the new centers of the clusters that are formed and then repartitions them based on the new centers. The K-Means algorithm continues this process until the membership within the clusters stabilizes, thus producing the final partitioning. The algorithm converges within a small number of iterations for the data sets tested in this paper.

3.2 DBSCAN Clustering

Density-based algorithms regard clusters as dense areas of objects that are separated by less dense areas. These clustering algorithms have an advantage over partition-based algorithms because they are not limited to finding spherical shaped clusters but can find clusters of arbitrary shapes. In this paper, the DBSCAN (Density Based Spatial Clustering of Applications with Noise) algorithm was chosen as a representative of density-based algorithms [5].

The DBSCAN algorithm is based on the concepts of density-reachability and density-connectivity. These concepts depend on two input parameters: epsilon (eps) and minimum number of points (minPts). Epsilon is the distance around an object that defines its eps-neighborhood. For a given object *q*, when the number of objects within the eps-neighborhood is at least minPts, then *q* is defined as a core object. All objects within its eps-neighborhood are said to be directly density-reachable from *q*. In addition, an object *p* is said to be density-reachable if it is within the eps-neighborhood of an object that is directly density-reachable or density-reachable from *q*. Furthermore, objects *p* and *q* are said to be density-connected if an object *o* exists that both *p* and *q* are density-reachable from.

These notions of density-reachability and density-connectivity are used to define what the DBSCAN algorithm considers as a cluster. A cluster is defined as the set of objects in a data set that are density-connected to a particular core object. Any object that is not part of a cluster is categorized as noise. This is in contrast to K-Means and AutoClass, which assign every object to a cluster.

The DBSCAN algorithm works as follows. Initially, all objects in the data set are assumed to be unassigned. DBSCAN then chooses an arbitrary unassigned object *p* from the data set. If DBSCAN finds *p* is a core object, it finds all the density-connected objects based on eps and minPts. It assigns all these objects to a new cluster. If DBSCAN finds *p* is not a core object, then *p* is considered to be noise and DBSCAN moves onto the next unassigned object. Once every object is assigned, the algorithm stops.

3.3 AutoClass

Probabilistic model-based clustering, previously considered in [10, 17], is another powerful clustering technique. We use an implementation of a probabilistic model-based clustering technique called AutoClass [1]. This algorithm allows for the automatic selection of the number of clusters and the soft clustering of the data. Soft clusters allow the data objects to be fractionally assigned to more than one cluster. For our work, we use the most probable assignment as the object's assignment.

To build the probabilistic model, the clustering algorithm determines the number of clusters and the parameters that govern the distinct probability distributions of each cluster. To accomplish this, AutoClass uses the Expectation Maximization (EM) algorithm [2]. The EM algorithm has two steps: an expectation step and a maximization step. The initial expectation step guesses what the parameters are using pseudo-random numbers. In the maximization step, the mean and variance are used to re-estimate the parameters continually until they converge to a local maximum. These local maxima are recorded and the EM process is repeated. This process continues until enough samples of the parameters have been found (we use 200 cycles in our results). AutoClass uses a Bayesian score to determine the best set of parameters to use for the probabilistic model. The Bayesian score is based on intra-cluster similarity and inter-cluster dissimilarity. Also, the Bayesian score penalizes models with more clusters to minimize potential over-fitting.

4. METHODOLOGY

4.1 Empirical Traces

To analyze the algorithms, we used data from two empirical packet traces. One is a publicly available packet trace called Auckland IV[2], the other is a full packet trace that we collected ourselves at the University of Calgary.

Auckland IV: The Auckland IV trace contains only TCP/IP headers of the traffic going through the University of Auckland's link to the Internet. We used a subset of the Auckland IV trace from March 16, 2001 at 06:00:00 to March 19, 2001 at 05:59:59. This subset provided sufficient connection samples to build our model (see Section 4.4).

Calgary: This trace was collected from a traffic monitor attached to the University of Calgary's Internet link. We collected this trace on March 10, 2006 from 1 to 2pm. This trace is a full packet trace with the entire payloads of all the packets captured. Due to the amount of data generated when capturing full payloads, the disk capacity (60 GB) of our traffic monitor was filled after one hour of collection, thus, limiting the duration of the trace.

4.2 Connection Identification

To collect the statistical flow information necessary for the clustering evaluations, the flows must be identified within the traces. These flows, also known as connections, are a bidirectional exchange of packets between two nodes.

In the traces, the data is not exclusively from connection-based transport layer protocols such as TCP. While this study focused solely on the TCP-based applications it should be noted that statistical flow information could be calculated for UDP traffic also. We identified the start of a connection using TCP's 3-way handshake and terminated a connection when FIN/RST packets were received. In addition, we assumed that a flow is terminated if the connection was idle for over 90 seconds.

The statistical flow characteristics considered include: total number of packets, mean packet size, mean payload size excluding headers, number of bytes transfered (in each direction and combined), and mean inter-arrival time of packets. Our decision to use these characteristics was based primarily on the previous work done by Zander *et al.* [17]. Due the heavy-tail distribution of many of the characteristics and our use of Euclidean distance as our similarity metric, we found that the logarithms of the characteristics gives much better results for all the clustering algorithms [13, 16].

4.3 Classification of the Data Sets

The publicly available Auckland IV traces include no payload information. Thus, to determine the connections "true" classifications port numbers are used. For this trace, we believe that a port-based classification will be largely accurate, as this archived trace predates the widespread use of dynamic port numbers. The classes considered for the Auckland IV datasets are DNS, FTP (control), FTP (data), HTTP, IRC, LIMEWIRE, NNTP, POP3, and SOCKS. LimeWire is a P2P application that uses the Gnutella protocol.

In the Calgary trace, we were able to capture the full payloads of the packets, and therefore, were able to use an automated payload-based classification to determine the "true" classes. The payload-based classification algorithm and signatures we used is very similar to those described by Karagiannis *et al.* [9]. We augmented their signatures to classify some newer P2P applications and instant messaging programs. The traffic classes considered for the Calgary trace are HTTP, P2P, SMTP, and POP3. The application breakdown of the Calgary trace is presented in Table 1. The breakdown of the Auckland IV trace has been omitted due to space limitations. However, HTTP is also the most dominant application accounting for over 76% of the bytes and connections.

[2]Available at: http://www.wand.net.nz/wand/wits/auck/

Table 1: Application breakdown of the Calgary trace

Application	Connections	Bytes	% Bytes
HTTP	1,132,920	23,693,723,103	47.3%
P2P	41,478	17,578,995,934	35.1%
SMTP	46,882	2,997,244,939	6.0%
IMAP	2,955	228,156,060	0.5%
POP3	3,674	72,274,560	0.1%
MSSQL	8,105	23,824,936	0.0%
OTHER	41,239	658,046,156	1.3%
UNKNOWN	354,798	4,811,332,761	9.6%

4.4 Testing Methodology

The majority of the connections in both traces carry HTTP traffic. This unequal distribution does not allow for equal testing of the different classes. To address this problem, the Auckland data sets used for the clustering consist of 1000 random samples of each traffic class, and the Calgary data sets use 2000 random sample of each traffic category. This allows the test results to fairly judge the ability on all traffic and not just HTTP. The size of the data sets were limited to 8000 connections because this was the upper bound that the AutoClass algorithm could cluster within a reasonable amount of time (4-10 hours). In addition, to achieve a greater confidence in the results we generated 10 different data sets for each trace. Each of these data sets was then, in turn, used to evaluate the clustering algorithms. We report the minimum, maximum, and average results from the data sets of each trace.

In the future, we plan on examining the practical issue of what is the best way to pick the connections used as samples to build the model. Some ways that we think this could be accomplished is by random selection or a weighted selection using different criteria such as bytes transfered or duration. Also, in order to get a reasonable representative model of the traffic, one would need to select a fairly large yet manageable number of samples. We found that K-Means and DBSCAN algorithms are able to cluster much larger data sets (greater than 100,000) within 4-10 hours.

5. EXPERIMENTAL RESULTS

In this section, the overall effectiveness of each clustering algorithm is evaluated first. Next, the number of objects in each cluster produced by the algorithms are analyzed.

5.1 Algorithm Effectiveness

The overall effectiveness of the clustering algorithms is calculated using *overall accuracy*. This overall accuracy measurement determines how well the clustering algorithm is able to create clusters that contain only a single traffic category.

The traffic class that makes up the majority of the connections in a cluster is used to label the cluster. The number of correctly classified connections in a cluster is referred to as the True Positives (TP). Any connections that are not correctly classified are considered False Positives (FP). Any connection that has not been assigned to a cluster is labelled as noise. The overall accuracy is thus calculated as follows:

$$overall\ accuracy = \frac{\sum TP\ for\ all\ clusters}{total\ number\ of\ connections}. \quad (3)$$

In the following subsections, the effectiveness of the K-Means, DBSCAN, and AutoClass algorithms are presented.

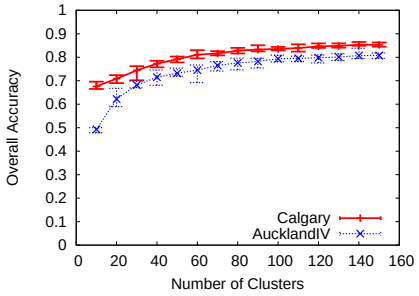

Figure 1: Accuracy using K-Means

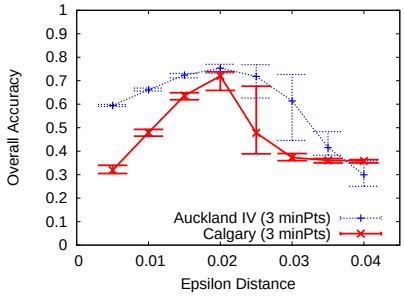

Figure 2: Accuracy using DBSCAN

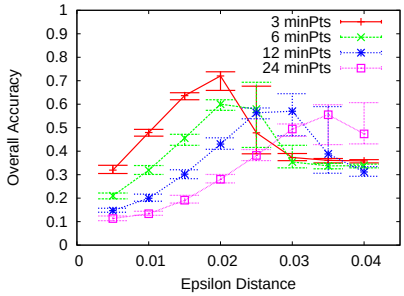

Figure 3: Parametrization of DBSCAN

Table 2: Accuracy using AutoClass

Data Set	Average	Minimum	Maximum
Auckland IV	92.4%	91.5%	93.5%
Calgary	88.7%	86.6%	90.0%

5.1.1 *K-Means Clustering*

The K-Means algorithm has an input parameter of K. This input parameter as mentioned in Section 3.1, is the number of disjoint partitions used by K-Means. In our data sets, we would expect there would be at least one cluster for each traffic class. In addition, due to the diversity of the traffic in some classes such as HTTP (e.g., browsing, bulk download, streaming) we would expect even more clusters to be formed. Therefore, based on this, the K-Means algorithm was evaluated with K initially being 10 and K being incremented by 10 for each subsequent clustering. The minimum, maximum, and average results for the K-Means clustering algorithm are shown in Figure 1.

Initially, when the number of clusters is small the overall accuracy of K-Means is approximately 49% for the Auckland IV data sets and 67% for the Calgary data sets. The overall accuracy steadily improves as the number of clusters increases. This continues until K is around 100 with the overall accuracy being 79% and 84% on average, for the Auckland IV and Calgary data sets, respectively. At this point, the improvement is much more gradual with the overall accuracy only improving by an additional 1.0% when K is 150 in both data sets. When K is greater than 150, the improvement is further diminished with the overall accuracy improving to the high 80% range when K is 500. However, large values of K increase the likelihood of over-fitting.

5.1.2 *DBSCAN Clustering*

The accuracy results for the DBSCAN algorithm are presented in Figure 2. Recall that DBSCAN has two input parameters (minPts, eps). We varied these parameters, and in Figure 2 report results for the combination that produce the best clustering results. The values used for minPts were tested between 3 and 24. The eps distance was tested from 0.005 to 0.040. Figure 3 presents results for different combinations of (minPts, eps) values for the Calgary data sets. As may be expected, when the minPts was 3 better results were produced than when the minPts was 24 because smaller clusters are formed. The additional clusters found using three minPts were typically small clusters containing only 3 to 5 connections.

When using minPts equal to 3 while varying the eps distance between 0.005 and 0.020 (see Figure 2), the DBSCAN algorithm improved its overall accuracy from 59.5% to 75.6% for the Auckland IV data sets. For the Calgary data sets, the DBSCAN algorithm improved its overall accuracy from 32.0% to 72.0% as the eps distance was varied with these same values. The overall accuracy for eps distances greater than 0.020 decreased significantly as the distance increased. Our analysis indicates that this large decrease occurs because the clusters of different traffic classes merge into a single large cluster. We found that this larger cluster was for connections with few packets, few bytes transfered, and short durations. This cluster contained typically equal amounts of P2P, POP3, and SMTP connections. Many of the SMTP connections were for emails with rejected recipient addresses and connections immediately closed after connecting to the SMTP server. For POP3, many of the connections contained instances where no email was in the users mailbox. Gnutella clients attempting to connect to a remote node and having its "GNUTELLA CONNECT" packets rejected accounted for most of the P2P connections.

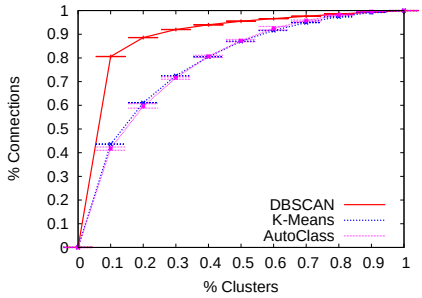

Figure 4: CDF of cluster weights

5.1.3 *AutoClass Clustering*

The results for the AutoClass algorithm are shown in Table 2. For this algorithm, the number of clusters and the cluster parameters are automatically determined. Overall, the AutoClass algorithm has the highest accuracy. On average, AutoClass is 92.4% and 88.7% accurate in the Auckland IV and Calgary data sets, respectively. AutoClass produces an average of 167 clusters for the Auckland IV data sets, and 247 clusters for the Calgary data sets.

5.2 Cluster Weights

For the traffic classification problem, the number of clusters produced by a clustering algorithm is an important consideration. The reason being that once the clustering is complete, each of the clusters must be labelled. Minimizing the number of clusters is also cost effective during the classification stage.

One way of reducing the number of clusters to label is by evaluating the clusters with many connections in them. For example, if a clustering algorithm with high accuracy places the majority of the connections in a small subset of the clusters, then by analyzing only this subset a majority of the connections can be classified. Figure 4 shows the percentage of connections represented as the percentage of clusters increases, using the Auckland IV data sets. In this evaluation, the K-Means algorithm had 100 for K. For the DBSCAN and AutoClass algorithms, the number of clusters can not be set.

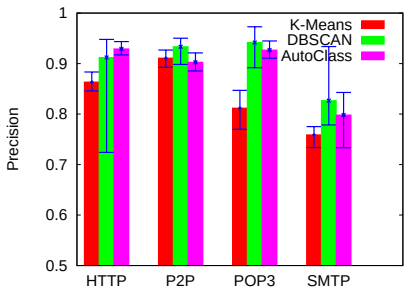

Figure 5: Precision using DBSCAN, K-Means, and AutoClass

DBSCAN uses 0.03 for eps, 3 for minPts, and has, on average, 190 clusters. We selected this point because it gave the best overall accuracy for DBSCAN. AutoClass has, on average, 167 clusters.

As seen in Figure 4, both K-Means and AutoClass have more evenly distributed clusters than DBSCAN. The 15 largest clusters produced by K-Means only contain 50% of the connections. In contrast, for the DBSCAN algorithm the five largest clusters contain over 50% of the connections in the data sets. These five clusters identified 75.4% of the NNTP, POP3, SOCKS, DNS, and IRC connections with a 97.6% overall accuracy. These results are unexpected when considering that by only looking at five of the 190 clusters, one can identify a significant portion of traffic. Qualitatively similar results were obtained for the Calgary data sets.

6. DISCUSSION

The DBSCAN algorithm is the only algorithm considered in this paper that can label connections as noise. The K-Means and AutoClass algorithms place every connection into a cluster. The connections that are labelled as noise reduce the overall accuracy of the DBSCAN algorithm because they are regarded as misclassified. We have found some interesting results by excluding the connections labelled as noise and just examining the clusters produced by DBSCAN. Figure 5 shows the *precision* values for the DBSCAN (eps=0.02, minPts=3), the K-Means (K=190), and the AutoClass algorithms using the Calgary data sets. Precision is the ratio of TP to FP for a traffic class. Precision measures the accuracy of the clusters to classify a particular category of traffic.

Figure 5 shows that for the Calgary data sets, the DBSCAN algorithm has the highest precision values for three of the four classes of traffic. While not shown for the Auckland IV data sets, seven of the nine traffic classes have average precision values over 95%. This shows that while DBSCAN's overall accuracy is lower than K-Means and AutoClass it produces highly accurate clusters.

Another noteworthy difference among the clustering algorithms is the time required to build the models. On average to build the models, the K-Means algorithm took 1 minute, the DBSCAN algorithm took 3 minutes, and the AutoClass algorithm took 4.5 hours. Clearly, the model building phase of AutoClass is time consuming. We believe this may deter systems developers from using this algorithm even if the frequency of retraining the model is low.

7. CONCLUSIONS

In this paper, we evaluated three different clustering algorithms, namely K-Means, DBSCAN, and AutoClass, for the network traffic classification problem. Our analysis is based on each algorithm's ability to produce clusters that have a high predictive power of a single traffic class, and each algorithm's ability to generate a minimal number of clusters that contain the majority of the connections. The results showed that the AutoClass algorithm produces the best overall accuracy. However, the DBSCAN algorithm has great potential because it places the majority of the connections in a small subset of the clusters. This is very useful because these clusters have a high predictive power of a single category of traffic. The overall accuracy of the K-Means algorithm is only marginally lower than that of the AutoClass algorithm, but is more suitable for this problem due to its much faster model building time. Ours in a work-in-progress and we continue to investigate these and other clustering algorithms for use as an efficient classification tool.

8. ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada and Informatics Circle of Research Excellence (iCORE) of the province of Alberta.

We thank Carey Williamson for his comments and suggestions which helped improve this paper.

9. REFERENCES

[1] P. Cheeseman and J. Strutz. Bayesian Classification (AutoClass): Theory and Results. *In Advances in Knowledge Discovery and Data Mining, AAI/MIT Press, USA*, 1996.

[2] A. P. Dempster, N. M. Paird, and D. B. Rubin. Maximum likelihood from incomeplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[3] C. Dews, A. Wichmann, and A. Feldmann. An analysis of internet chat systems. In *IMC'03*, Miami Beach, USA, Oct 27-29, 2003.

[4] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster Analysis and Display of Genome-wide Expression Patterns. *Genetics*, 95(1):14863–15868, 1998.

[5] M. Ester, H. Kriegel, J. Sander, and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD 96)*, Portland, USA, 1996.

[6] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: Automated Construction of Application Signatures. In *SIGCOMM'05 MineNet Workshop*, Philadelphia, USA, August 22-26, 2005.

[7] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, USA, 1988.

[8] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy. Transport Layer Identification of P2P Traffic. In *IMC'04*, Taormina, Italy, October 25-27, 2004.

[9] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINK: Multilevel Traffic Classification in the Dark. In *SIGCOMM'05*, Philadelphia, USA, August 21-26, 2005.

[10] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow Clustering Using Machine Learning Techniques. In *PAM 2004*, Antibes Juan-les-Pins, France, April 19-20, 2004.

[11] A. W. Moore and K. Papagiannaki. Toward the Accurate Identification of Network Applications. In *PAM 2005*, Boston, USA, March 31-April 1, 2005.

[12] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *SIGMETRIC'05*, Banff, Canada, June 6-10, 2005.

[13] V. Paxson. Empirically-Derived Analytic Models of Wide-Area TCP Connections. *IEEE/ACM Transactions on Networking*, 2(4):316–336, August 1998.

[14] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification. In *IMC'04*, Taormina, Italy, October 25-27, 2004.

[15] S. Sen, O. Spatscheck, and D. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In *WWW2005*, New York, USA, May 17-22, 2004.

[16] I. H. Witten and E. Frank. *(2005) Data Mining: Pratical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[17] S. Zander, T. Nguyen, and G. Armitage. Automated Traffic Classification and Application Identification using Machine Learning. In *LCN'05*, Sydney, Australia, Nov 15-17, 2005.