Automated Traffic Classification and Application Identification using Machine Learning

Sebastian Zander, Thuy Nguyen, Grenville Armitage
Centre for Advanced Internet Architectures
Swinburne University of Technology, Melbourne, Australia
{szander,tnguyen,garmitage}@swin.edu.au

Abstract

The dynamic classification and identification of network applications responsible for network traffic flows offers substantial benefits to a number of key areas in IP network engineering, management and surveillance. Currently such classifications rely on selected packet header fields (e.g. port numbers) or application layer protocol decoding. These methods have a number of shortfalls e.g. many applications can use unpredictable port numbers and protocol decoding requires a high amount of computing resources or is simply infeasible in case protocols are unknown or encrypted. We propose a novel method for traffic classification and application identification using an unsupervised machine learning technique. Flows are automatically classified based on statistical flow characteristics. We evaluate the efficiency of our approach using data from several traffic traces collected at different locations of the Internet. We use feature selection to find an optimal feature set and determine the influence of different features.

1. Introduction

Recent years have seen a dramatic increase in the variety of applications using the Internet. In addition to 'traditional' applications (e.g. email, web or ftp) new applications have gained strong momentum (e.g. streaming, gaming or peer-to-peer (P2P)). The ability to dynamically identify and classify flows according to their network applications is highly beneficial for:

- Trend analyses (estimating the size and origins of capacity demand trends for network planning)
- Adaptive, network-based marking of traffic requiring specific QoS without direct client-application or end-host involvement
- Dynamic access control (adaptive firewalls that can detect forbidden applications, Denial of Service (DoS) attacks or other unwanted traffic)
- Lawful Interception (enabling minimally invasive warrants and wire-taps based on statistical summaries of traffic details)
- Intrusion detection (detect suspicious activities related to security breaches due to malicious users or worms)

The most common identification technique based on the inspection of 'known port numbers' is no longer accurate because many applications no longer use fixed, predictable port numbers. The Internet Assigned Numbers Authority (IANA) [1] assigns the well-known ports from 0-1023 and registers port numbers in the range from 1024-49151. But many applications have no IANA assigned or registered ports and only utilise 'well known' default ports. Often these ports overlap with IANA ports and an unambiguous identification is no longer possible [2]. Even applications with well-known or registered ports can end up using different port numbers because (i) non-privileged users often have to use ports above 1023, (ii) users may be deliberately trying to hide their existence or bypass port-based filters, or (iii) multiple servers are sharing a single IP address (host). Furthermore some applications (e.g. passive FTP or video/voice communication) use dynamic ports unknowable in advance.

A more reliable technique used in many current industry products involves stateful reconstruction of session and application information from packet content (e.g. [3]). Although this technique avoids reliance on fixed port numbers, it imposes significant complexity and processing load on the traffic identification device. It must be kept up-to-date with extensive knowledge of application semantics and network-level syntax, and must be powerful enough to perform concurrent analysis of a potentially large number of flows. This approach can be difficult or

The authors thank Cisco Systems, Inc. for supporting this work with a University Research Project grant.

impossible when dealing with proprietary protocols or encrypted traffic. Another problem is that direct analysis of session and application layer content may represent an explicit breach of organisational privacy policies or violation of relevant privacy legislation. The authors of [4] propose signature-based methods to classify P2P traffic. Although these approaches are more efficient than stateful reconstruction and provide better classification than the port-based approach they are still protocol dependent. The authors of [5] describe a method to bypass protocol-based detectors.

Previous work used a number of different parameters to describe network traffic (e.g. [4], [6], [7]) including the size and duration of flows, packet length and interarrival time distributions, flow idle times etc. We propose to use Machine Learning (ML) [8] to automatically classify and identify network applications based on these parameters. A ML algorithm automatically builds a classifier by learning the inherent structure of a dataset depending on the characteristics. Over the past decade, ML has evolved from a field of laboratory demonstrations to a field of significant commercial value [9]. Our approach includes the task of identifying the optimal set of flow attributes that minimizes the processing cost, while maximizing the classification accuracy. We evaluate the effectiveness of our approach using traffic traces collected at different locations of the Internet.

The rest of the paper is organized as follows. Section 2 presents an overview about related work. Section 3 describes our approach for ML-based application identification. Section 4 evaluates our approach using traffic traces. Section 5 concludes and outlines future work.

2. Related Work

The idea of using ML techniques for flow classification was first introduced in the context of intrusion detection [10].

The authors of [11] use principal component analysis (PCA) and density estimation to classify traffic into different applications. They use the distributions of two flow attributes from a fairly small dataset and study a few well-known ports.

In [12] the authors use nearest neighbour (NN) and linear discriminate analysis (LDA) to successfully map applications to different QoS classes using up to four attributes. With this approach, the number of classes is small and known a-priori. The attributes used for the classification have been aggregated over 24 hour periods.

The Expectation Maximization (EM) algorithm is used in [13] to cluster flows into different application types using a fixed set of attributes. The authors find that the algorithm separates the traffic into few basic classes, but from their evaluation it is not clear what influence different attributes and EM parameters have and how good the clustering actually is.

In [14] the authors use a simulated annealing EM algorithm to classify traffic flows based on their size (e.g. mice and elephants). The authors conclude that their approach produces more meaningful results than previous threshold-based methods.

We have proposed an ML-based approach for identifying different network applications in [15]. In this paper we evaluate the approach using a number of traffic traces collected at different locations in the Internet.

The authors of [16] are using a similar approach based on the Naïve Bayes classifier and a large number of flow attributes. They only use one data set but the flows in this set have been hand-classified allowing a very accurate evaluation.

Research on combining different non-ML techniques to identify network applications is presented in [17].

3. ML-based Application Identification

First we classify packets into bidirectional flows and compute the flow characteristics using NetMate [18]. Sampling can be used to only select a subset of the flow data to improve the performance of the learning process (see [15] for more details). The flow characteristics and a model of the flow attributes are then used to learn the classes (1). Once the classes have been learned new flows can be classified (2).

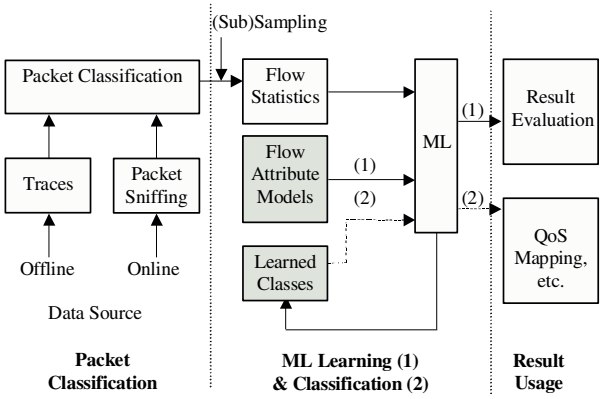

Figure 1: ML-based flow classification

Note that we use the term learning for the initial process of constructing the classifier to differentiate it from the latter process of classification. However, in other work this is sometimes called classification or clustering. The results of the learning and classification can be exported for evaluation. The results of the classification would be used for e.g. QoS mapping, trend analysis etc.

3.1. ML Algorithm

For the machine learning we use the autoclass approach [19]. autoclass is an unsupervised Bayesian classifier that automatically learns the 'natural' classes (also called clustering) inherent in a training dataset with unclassified instances based on some attributes of the instances. The resulting classifier can then be used to classify new unseen instances. In this section we only provide an overview and the reader is encouraged to read [19] for a detailed description of the algorithm.

In the autoclass model all instances must be conditionally independent and therefore any similarity between two instances is accounted for by their class membership only. The class an instance is a member of is an unknown or hidden attribute of each instance.

The probability that an instance X_i of a set of I instances (the database) with attributes $\vec{X}_i$ is a member of a particular class C_j of a set of J classes consists of two parts: the interclass probability and the probability density function of each class (intraclass probability). Because the classes constitute a discrete partitioning of the data, the appropriate interclass probability density function is a Bernoulli distribution characterised by a set $\vec{V}_j$ of probabilities $\{\pi_1,\ldots,\pi_j\}$ constrained that $0 \le \pi_j \le 1$ and $\sum_j \pi_j = 1$. Thus

$$P(X_i \in C_j \mid \vec{V}_c) \equiv \pi_j \quad (1)$$

The interclass probability does only depend on J and the known number of instances assigned to C_j but not on $\vec{X}_i$. The intraclass probability is the product of the conditionally independent probability distributions of the k attributes:

$$P(\vec{X}_i \mid X_i \in C_j, \vec{V}_j) = \prod_k P(X_{ik} \mid X_i \in C_j, \vec{V}_{jk}) \quad (2)$$

autoclass supports discrete and real valued attribute models for the individual $P(X_{ik})$. However, we only use real attributes, which are modelled with lognormal distributions. Therefore we assume there is only one functional form for the attribute probability density functions and have omitted this parameter from all the equations (see [19] for the more general approach). Combining the interclass and intraclass probabilities we get the direct probability that an instance X_i with attribute values $\vec{X}_i$ is a member of class C_j:

$$P(\vec{X}_i, X_i \in C_j \mid \vec{V}_c, \vec{V}_j) = \pi_j \prod_k P(X_{ik} \mid X_i \in C_j, \vec{V}_{jk}) \quad (3)$$

By introducing priors on the parameter set this can be converted into a Bayesian model obtaining the joint probability of the parameter set $\vec{V}$ and the current database X:

$$P(X\vec{V}) = P(\vec{V})P(X \mid \vec{V}) \quad (4)$$

The goal is to find maximum posterior parameter values obtained from the parameters posterior probability density function:

$$P(\vec{V} \mid X) = \frac{P(X,\vec{V})}{P(X)} = \frac{P(X,\vec{V})}{\int d\vec{V} P(X,\vec{V})} \quad (5)$$

One could use this equation directly, computing the posterior probabilities for every partitioning of the data into J non-empty subsets. But the number of possible partitions approach J^I for small J making the approach computationally infeasible for large sets of instances and/or classes. Therefore autoclass uses approximation based on the EM algorithm [20]. autoclass cycles between estimating the class assignments for all instances and estimating the parameters $\vec{V}$. The EM algorithm is guaranteed to converge to a local maximum. In an attempt to find the global maximum autoclass performs repeated EM searches starting from pseudo-random points in the parameter space. The model with the parameter set scoring the highest probability given the current database is chosen as the best.

autoclass can be preconfigured with the number of classes (if known) or it can try to estimate the number of classes itself. For our problem the exact number of classes is unknown in advance. One could argue that there should be exactly one class per application. However, we found the distributions of flow attributes – even for a single application – can be quite complex. Because we are using simple attribute models (lognormal distributions) it is not possible to model each application with a single class. When the number of classes is unknown autoclass is configured with a start list of class numbers $\vec{J}_{start}$. Then for each EM search the initial number of classes is taken from the next entry in $\vec{J}_{start}$ as long as there are entries left. The number of classes at the end of an EM search can be smaller if some of the classes drop out of convergence. For all further iterations after the start

list is exhausted autoclass randomly chooses J from a lognormal distribution fitted to the number of classes of the best 10 classifications found so far.

Having multiple classes per application provides the advantage of a more fine-grained view into an application. For instance, web traffic is used for different purposes (e.g. bulk transfers, interactive interfaces, streaming, etc.) and for detailed analysis it would be beneficial to differentiate between them. On the other hand an increasing number of classes decreases the performance of the approach in terms of runtime and memory.

3.2. Feature Selection

Our feature selection technique is based on the actual performance of the learning algorithm. This method generally achieves the highest accuracy because it 'tailors' the feature set to the algorithm. On the downside it is much more computationally expensive (especially when the learning algorithm is not very fast) than algorithm-independent methods such as correlation-based feature selection (CFS) [21].

Finding the combination of attributes that provides the most contrasting application classes is a repeated process of (i) selecting a subset of attributes, (ii) learning the classes and (iii) evaluating the class structure. We implemented sequential forward selection (SFS) to find the best attribute set because an exhaustive search is not feasible. The algorithm starts with every single attribute. The attribute that produces the best result is placed in a list of selected attributes SEL(1). Then all combinations of SEL(1) and a second attribute not in SEL(1) are tried. The combination that produces the best result becomes SEL(2). The process is repeated until no further improvement is achieved. SFS is only one (simple) approach to identify the most useful feature set and there are other approaches such as sequential backward elimination (see [22]).

To assess the quality of the resulting classes we have developed a metric termed intra-class homogeneity H. We define A and C as sets of applications and classes found during the learning respectively. We also define a function *count(a,c)* that counts the number of flows that application $a \in A$ has in class $c \in C$. Then the homogeneity $H(c)$ of a class c is defined as the largest fraction of flows of one application in the class:

$$H(c) = \frac{\max(count(a,c) \mid a \in A)}{\sum_a count(a,c)} \qquad (6)$$

The overall homogeneity H of a set of classes is the mean of the class homogeneities:

$$H = \frac{\sum_c H(c)}{\|C\|} \quad \text{and} \quad (0 \le H \le 1) \qquad (7)$$

The goal is to maximize H to achieve a good separation between different applications. The reason why we use H as an evaluation metric, instead of standard metrics like accuracy, precision and recall, is that we are using an unsupervised learning technique. The number of classes and the class to application mapping is not known before the learning. Afterwards each class can be assigned to the application that has the most flows in it. However, if more than one application contributes a significant number of flows to a class the mapping can be difficult. High homogeneity values are required in order to unambiguously map a class to an application.

4. Evaluation

4.1. Trace Files

For the evaluation we use the Auckland-VI, NZIX-II and Leipzig-II traces from NLANR [23] captured in different years at different locations in the Internet. Because we are limited to use public available anonymised traces we are unable to verify the true applications that created the flows. In our evaluation we therefore assume a flow's IANA defined or registered server port identifies the application. In our case the server port is usually the destination port of the bidirectional flows. In rare cases where the source port was the IANA defined port we have swapped both directions of the flow (including IP addresses, ports and flow attributes).

We admit that assuming the server port always identifies an application is not correct. However, we assume that for the ports we use in this study the majority of the traffic is from the expected application. Then it is most likely that few 'wrong' flows would decrease the homogeneity of the learned classes. Therefore our evaluation results can be treated as lower bound of the effectiveness. We also do not consider traffic of the selected applications on other than the standard server ports e.g. we do only consider web traffic on port 80 but not on port 81. Assuming there is no strong correlation between the used server port and the application characteristics this does not introduce any additional bias because it can be viewed as random sampling.

4.2. Flow Attributes

Our attribute set includes packet inter-arrival time and packet length mean and variance, flow size (bytes) and duration. Aside from duration all attributes are bidirectional meaning they are separately computed for both directions of a flow. Our goal is to minimise the number of attributes and we only use 'basic' attributes that can be easily computed. We are not using the server port as an attribute because 'wrong' ports could introduce an unknown bias.

In our analysis we exclude flows that have less than three packets in each direction because for very short flows only some attributes could be computed e.g. flows containing only a single packet would provide no inter-arrival time statistics and packet length statistics can only be computed in the forward direction. This would more than halve the number of available attributes making it difficult to separate different applications and would most likely bias the attribute influence results. Furthermore any valid TCP flows should have at least six packets. However, valid UDP flows can consist of just two packets and therefore excluding small UDP flows may have biased the results for DNS and Half-Life traffic.

This strategy clearly leaves aside a substantial number of flows but in this work we aim to separate different applications and we are not interested in 'strange' flows or anomalies. In fact using these flows would be dangerous because we rely on the server port for validating our approach. For instance, if we would use one-packet flows we might confuse port scans with the real application. However, in general small flows can provide interesting insights and should not be ignored especially from a security viewpoint.

4.3. Identifying Network Applications

For performance reasons we use a subset of 8,000 flows from each trace file. For each application (FTP data, Telnet, SMTP, DNS, HTTP, AOL Messenger, Napster, Half-Life) we randomly sample 1,000 flows out of all flows of that particular application. We derive the flow samples and perform the SFS for each of the four traces using the same parameters for the learning algorithm.

Figure 2 shows an example result of a single run of the algorithm with a fixed attribute set. It shows how the applications are distributed among the classes that have been found. The classes are ordered with decreasing class size from left to right (increasing class number). For each of the classes the homogeneity is the maximum fraction of flows of one application e.g. H(leftmost)=0.52 and H(rightmost)=1. The overall homogeneity H is the mean of all class homogeneities and in this case H=0.86.

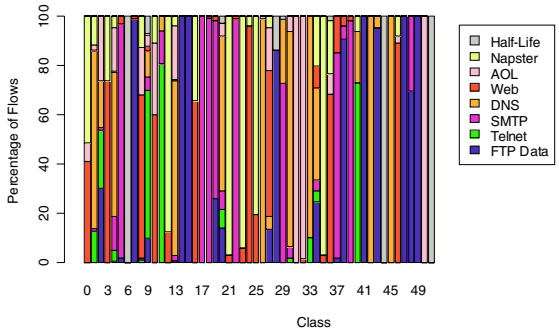

Figure 2: Example distribution of applications across the classes

Figure 3 shows the overall mean H depending on the number of attributes. It shows the overall efficiency in identifying the different application increases with the number of attributes until it reaches a maximum between 0.85 and 0.89 depending on the trace. That means on average there is fairly high separation between the flows of different applications. The number of attributes in the final best sets varies between four and six but is always significantly smaller than the total number of attributes used in the SFS (eleven). We also performed training on the full feature set for all traces but found no significant homogeneity improvement (values between 0.85 and 0.90) but the learning was significantly slower.

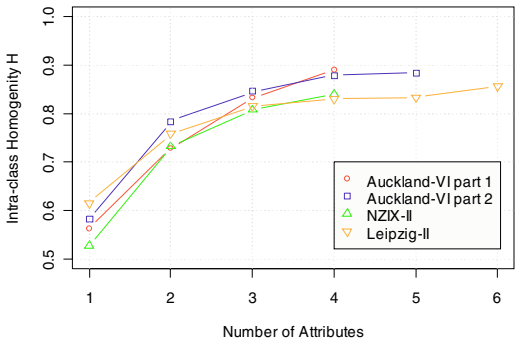

Figure 3: Homogeneity depending on the number of flow attributes and different traces

Figure 4 shows what features have been selected for the best sets for each of the traces. The y-axis shows the percentage of traces a feature made it into the best set. Although the best feature sets are different for all the traces there is a clear trend towards some of the features. Packet length statistics seems to be preferred over inter-arrival times statistics and duration and backward volume also seem to be of limited value.

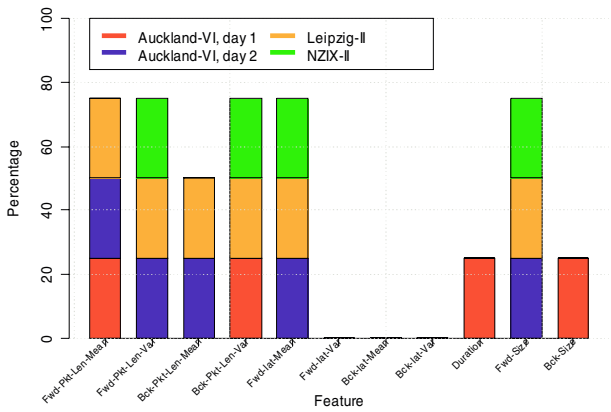

Figure 4: Features selected for the best feature sets of different traces

We believe the reason why the packet length is preferred over inter-arrival times is because none of the applications we investigate has very characteristic inter-arrival time distributions. If for instance we had chosen voice communication where applications have very characteristic inter-arrival times (e.g. one packet every 20ms) we would expect inter-arrival times to be much more useful.

Game traffic such as Half-Life traffic is known to have very characteristic inter-arrival times. However, this is only true for the traffic that exchanges game state information during a game. Most Half-Life traffic flows in our dataset are actually caused by players just querying information from the server such as the number of active players etc. A potential problem with inter-arrival times is that packet queuing in routers can change their distributions especially in case of congestion. In contrast packet lengths are usually constant in case there is no intermediate fragmentation or encryption.

We also estimate the influence of the different attributes on the outcome of the learning. The influence of an attribute on a particular class is defined as the cross entropy for the class distribution w.r.t. the global distribution of a single class classification. The total influence of an attribute is then the class probability weighted average of the influence in each of the classes. The influence values range from 0 (no influence) to 1 (maximum influence). Table 1 shows the mean values (based on the learning results when using the best attribute sets) across the different traces. The results are similar to Figure 4 in that packet length and volume statistics are most influential while inter-arrival times and duration have less influence.

We computed the homogeneity of the classes for each of the different applications. Figure 5 shows the per-application homogeneity distribution across the different traces. The per-application homogeneity is defined as the mean homogeneity of all classes where an application has the largest fraction. The distributions are shown as boxplots. The lower end, middle and upper end of the box are the 1st quartile, median and 3rd quartile of the distribution. The whiskers extend to the most extreme values.

Table 1: Attribute influence

Attribute	Influence
Forward-Pkt-Len-Var	1.0
Backward-Pkt-Len-Var	0.89
Backward-Bytes	0.84
Forward-Pkt-Len-Mean	0.77
Forward-Bytes	0.75
Backward-Pkt-Len-Mean	0.69
Duration	0.62
Forward-IAT-Mean	0.56

The figure shows that the classes of some applications are quite homogenous (e.g. Half-Life) but for others they are less homogenous (e.g. Telnet, Web). The higher the homogeneity the more likely it is to separate an application from all the others. Some applications such as Half-Life can be well separated from the rest but others such as FTP seem to have characteristics very similar to other applications.

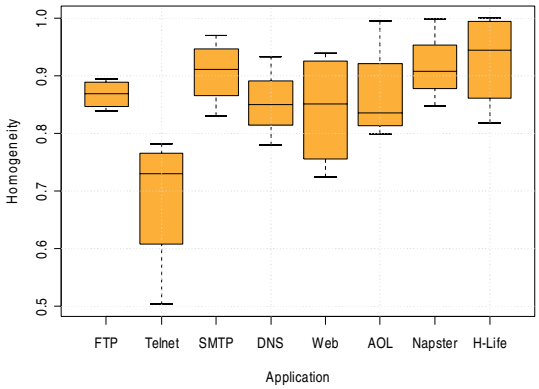

Figure 5: Mean homogeneity per application across different traces

Figure 6 shows the percentage of flows in the 'correct' classes for each application and all traces (this is usually called accuracy). To compute the accuracy we map each class to the application that is dominating the class (by having the largest fraction of flows in that class). The figure indicates the expected classification accuracy. It shows that some

applications have a very high accuracy but there are some problems e.g. for Napster there is one trace where it is not dominating any of the classes (hence the accuracy is 0%). However, it should be noted that the median is always 80% or higher and for some applications it is close to 95%. The mean accuracy across all trace files is 86.5%.

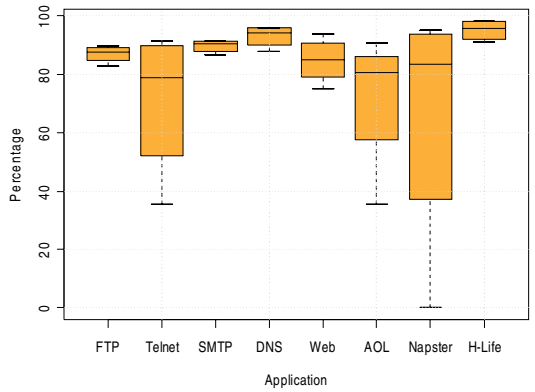

Figure 6: Accuracy per application across different traces

While the accuracy gives the percentage of correctly classified flows it does not provide a measure into which applications flows are likely to be misclassified. To address this issue we also compute the false positive rate per application, which is defined as the number of misclassified flows divided by the total number of flows in all classes assigned to the application. Figure 7 shows the percentage of false positives for each application. FTP, Telnet and Web traffic have the highest percentage of false positives. Telnet has the problem that it heavily overlaps with other applications while FTP and Web seem to have the most diverse traffic characteristics and are spread across many classes.

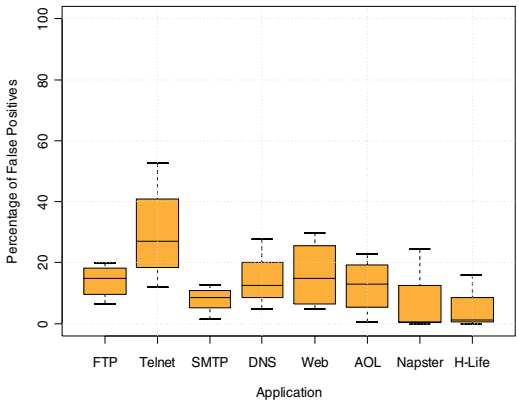

Figure 7: False positives per application across different traces

To visualize which applications have the most diverse characteristics the percentage of classes an application has at least one flow in is shown in Figure 8 (spread of an application among the different classes). Not surprisingly web traffic is the most distributed application (similar results were found in [13]) whereas game traffic is the least distributed. On average the total number of classes found was 100.

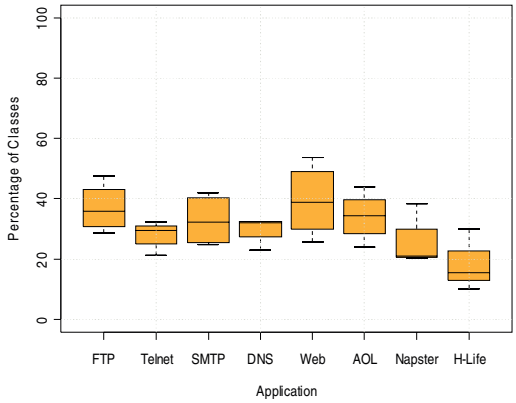

Figure 8: Spread of the applications over the classes across different traces

Although the main focus is on achieving a good separation of the different applications for the purpose of high classification accuracy, it would also be desirable to minimise the number of classes to improve the speed of the learning and classification and minimise memory requirements.

5. Conclusions and Further Research

We have proposed a novel method for ML-based flow classification and application identification based on statistical flow properties. We used a feature selection technique for finding the optimal set of flow attributes and evaluated the effectiveness of our approach. We also quantified the influence of different attributes on the learning. The results show that some separation of the applications can be achieved depending on the particular application. The average accuracy across all traces is 86.5%. While some applications seem to have more characteristic attributes and can be well separated others intermingle and are harder to identify.

We plan to evaluate our approach with a larger number of flows and more applications. Using packet traces with payload or flow data where the applications have already been identified would allow us to improve our evaluation. We work on comparing our algorithm with a simple Bayesian classifier as used in [16].

It is important to identify why some applications cannot be easily separated from others and to develop new, better flow attributes to improve homogeneity. We want to experiment with attributes such as idle time, burstiness and metrics computed from payload information in a protocol independent way.

The precision and recall of the resulting classifier and the classification performance needs to be evaluated. We also have not yet investigated the effect of flow sampling on our results, although it should be noted that the selection of any kind of data set already represents some form of sampling. In high-speed networks the use of packet sampling is inevitable and it is important to know how the sampling error affects the flow attributes (see [24]). Another interesting question is how many packets of a flow are required for reliable identification (in most scenarios the flows would have to be classified as quickly as possible) and how stable classifications are over time (the predicted class should only change when the network application changes).

Possibly our approach could also be applied to detect security incidents such as port scans or other malicious traffic but we have not yet extended our study into this area. Another issue left for further study is to quantify the performance in terms of processing time and memory consumption and to investigate the trade-off between the approach's accuracy and processing overhead.

6. References

[1] IANA, http://www.iana.org/assignments/port-numbers (as of August 2005).

[2] Ports database, http://www.portsdb.org/ (as of August 2005).

[3] Cisco IOS Documentation, "Network-Based Application Recognition and Distributed Network-Based Application Recognition", http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t8/dtnbarad.htm (as of August 2005).

[4] S. Sen, O. Spatscheck, D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures", WWW 2004, New York, USA, May 2004.

[5] Reno, NV. "An Analysis of Flow Identification in QoS Systems", Poster at ACM SIGCSE 2003, Reno, USA, February 2003.

[6] K. Lan, J. Heidemann, "On the correlation of Internet flow characteristics", Technical Report ISI-TR-574, USC/Information Sciences Institute, July, 2003.

[7] K. Claffy, H.-W. Braun, G. Polyzos, "Internet Traffic Profiling", CAIDA, San Diego Supercomputer Center, http://www.caida.org/ outreach/papers/1994/itf/ , 1994.

[8] Tom M. Mitchell, "Machine Learning", McGraw-Hill Education (ISE Editions), December 1997.

[9] Tom M. Mitchell, "Does Machine Learning Really Work?", AI Magazine 18(3), pp. 11-20, 1997.

[10] J. Frank, "Machine Learning and Intrusion Detection: Current and Future Directions", Proceedings of the National 17th Computer Security Conference, 1994.

[11] T. Dunnigan, G. Ostrouchov, "Flow Characterization for Intrusion Detection", Oak Ridge National Laboratory, Technical Report, http://www.csm.ornl.gov/~ost/id/tm.ps, November 2000.

[12] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, "Class-of-Service Mapping for QoS: A statistical signature-based approach to IP traffic classification, ACM SIGCOMM Internet Measurement Workshop 2004, Taormina, Sicily, Italy, 2004.

[13] A. McGregor, M. Hall, P. Lorier, J. Brunskill, "Flow Clustering Using Machine Learning Techniques", Passive & Active Measurement Workshop 2004 (PAM 2004), France, April 19-20, 2004.

[14] A. Soule, K. Salamatian, N. Taft, R. Emilion, and K. Papagiannaki, "Flow Classification by Histograms or How to Go on Safari in the Internet", In ACM Sigmetrics, New York, U.S.A., June, 2004.

[15] S. Zander, T.T.T. Nguyen, G. Armitage, "Self-learning IP Traffic Classification based on Statistical Flow Characteristics", Passive & Active Measurement Workshop (PAM) 2005, Boston, USA, March/April 2005.

[16] D. Zuev, A. Moore, "Traffic Classification using a Statistical Approach", Passive & Active Measurement Workshop, Boston, U.S.A, March/April 2005.

[17] A. Moore, and K. Papagiannaki, "Toward the Accurate Identification of Network Applications", Passive & Active Measurement Workshop, Boston, U.S.A., March/April, 2005.

[18] NetMate, http://sourceforge.net/projects/netmate-meter/ (as of August 2005).

[19] P. Cheeseman, J. Stutz, "Bayesian Classification (Autoclass): Theory and Results", Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, USA, 1996.

[20] A. Dempster, N. Laird, D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm, Journal of Royal Statistical Society, Series B, Vol. 30, No. 1, 1977.

[21] M. Hall, "Correlation-based Feature Selection for Machine Learning", Ph.D diss., Waikato University, Department of Computer Science, Hamilton, NZ, 1998.

[22] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization", Proceedings of ICML'97, 14th International Conference on Machine Learning", pp. 412-420, 1997.

[23] NLANR traces: http://pma.nlanr.net/Special/ (as of August 2005).

[24] N. Hohn, D. Veitch, "Inverting Sampled Traffic", ACM/SIGCOMM Internet Measurement Conference, Miami, USA, November 2003.