A Machine Learning Approach for Efficient Traffic Classification

Wei Li* and Andrew W. Moore†
* Department of Computer Science, Queen Mary University of London
† Computer Laboratory, University of Cambridge

***Abstract*— Online traffic classification continues to be of long-term interest to the networking community. It serves as the input for practical solutions such as network monitoring, quality-of-service and intrusion-detection. In this paper we present a machine-learning approach that accurately classifies internet traffic using C4.5 decision tree. Accuracy is not our only concern; the latency and throughput are also of extreme importance. Without inspecting packet payload, our method can identify traffic of different types of applications with 99.8% total accuracy, by collecting 12 features at the start of the flows.**

I. Introduction

The Internet is evolving towards a vast, ubiquitous infrastructure, supporting an increasingly-huge market of data communication and digital media and producing trillions of dollars of revenue each year. The data transmission is governed by simple end-to-end transmission protocols such as TCP and UDP, without efficient monitoring, auditing and intelligent control over the traffic, but the success of the Internet has led to the emergence of a seemingly-uncountable variety of applications.

Along with the development and evolution of the applications on the Internet, an efficient application classification scheme is highly desirable to support various solutions such as advanced network monitoring, network resource management, anomaly detection, application-specific strategies and network auditing activities. Moreover, the application-level knowledge of the Internet is extremely useful for those who set out to model Internet traffic or to investigate the long-term changes and requirements for the Internet.

The Internet traffic, in principle, is the product of a complex multifactor system involving a range of networks, hosts, applications and different people closely interacting with each other. The complexity is continuously increasing as people keep producing a vast variety of network applications and application layer protocols that, in many ways, break the traditional assumptions:

1) [1] reported that only 50-70% of the Internet traffic was classifiable using the official International Assigned Number Authority (IANA) list. Emerging applications and proxies often avoid the use of standard host ports.
2) Port-based schemes are also overly simplistic confusing applications. For example, VoIP telephony system, chat-messenger systems such as MSN Instant Messenger, and regular web-page browsing would use the same port.
3) The proportion of encapsulated or encrypted traffic is increasing. Examples include proxies, VPN, tunneling, and applications using a different protocol to exchange data (e.g. GetByMail [2]). Encapsulation would change the pattern of the original application level protocol, while encryption of packet payload also renders the identification mechanisms based on payload inspection inefficient.

However, the nature of each Internet application allows it to be classified into one of several discrete categories. Examples include: web-browsing, multimedia data such as VOIP, email activities, peer-2-peer and FTP file transfers and malicious traffic. This taxonomy, originating with the “Class of Service” (CoS) [3], was further extended in [1]. It was noted that different kinds of applications have diverse objectives and characteristics, which may also cause diverse behaviour in the traffic flows, i.e. “traffic patterns”.

Based upon observed traffic patterns, we developed a classification scheme providing near-real-time classification of up to 99.8% of traffic, using behavioural features and C4.5 decision tree algorithm [4]. This approach is fundamentally different from traditional traffic classification approaches in that:

1) It does not rely on port numbers. Further, we presume no prior knowledge about port-application mapping in our approach.
2) Our approach does not require the inspection of traffic payload.
3) The behavioural features, e.g. distribution of the size of packets, TCP window size, TCP flag bits and packet directions, are derived from the packet-headers, the same source of information that is expected to be used by the routers of the Internet.

In this paper we document the machine learning approach as well as a traffic classification system operating in near real-time. In addition to the demonstration of accuracy, the trade-off between accuracy and complexity which is a significant practical issue is also fully discussed. The remainder of the paper is organised as follows: the next section is a review of related work. Section III illustrates our classification approach and system complexity for online traffic classification. Section IV presents detailed results and evaluation. Section V concludes the paper and outlines our future work.

II. Related Work

In recent literature, many different methods have been introduced to solve the traffic classification problem. The majority of current classification approaches still relies on the information of host port numbers, IPs and signatures for classification and intrusion detection, such as light-weighted intrusion detection systems including Bro [5] and Snort [6].

Moore and Zuev in [7] presented a statistical approach to classify the traffic into different types of services. A naive Bayes classifier, combined with kernel estimation and a correlation-based filtering algorithm was used to solve the classification problem on offline TCP traces. The resulting accuracy, up to 96% (which degrades to 93% after 8 months), demonstrated the discriminative power of a combination of 10 flow-behaviour features, with an unsophisticated machine-learning mechanism.

Williams et al. in [8] carried out an empirical study of comparing five widely-utilised machine learning algorithms to classify Internet traffic. Among these algorithms, AdaBoost+C4.5 achieved the highest accuracy in their results. This serves as a guidebook for algorithms, but their feature set used is relatively unsophisticated, containing only packet lengths, total bytes, total packets, inter-arrival times, flow duration and protocol. Based on the mechanism in this work, [9] further moved on to classify game traffic with an observation window of no more than 25 packets, which is also looking forward to real-time classification.

Bernaille et al. presented an approach to identify applications using start-of-flow information in [10]. The authors utilised the packet size and direction of the first 4 data packets in each flow as the features with which they trained Gaussian and Hidden Markov Models respectively. These models received 98.7% overall accuracy when assisted by an expanded port-number list, and 93.7% overall accuracy using simple prediction heuristics. It is worth noting that this work only retrieved a very small amount of information from the flows. The authors further specialised their work to the identification of encrypted traffic in [11].

Another statistical fingerprinting mechanism was proposed in [12]. The information they use include size of the IP packets, inter-arrival time and the order of packets seen on the link. A number of other works attempted the same problem with different machine learning mechanisms such as clustering [13] [14], LDA and k-NN [3].

Karagiannis et al. [15] studied multi-level behaviour of the traffic such as analysing interaction between hosts, protocol usage and per-flow average packet size. Their results show an ability of classifying 80%-90% of the traffic with 95% accuracy. In their recent work [16] they presented an interesting investigation to profile the users activity and behaviours, and to analyse the dynamic characteristics of the host behaviours.

From the results in terms of accuracy we may assert that the combination of a small number of flow-behaviour features already has strong discriminative power to differentiate services or network applications. For non-anomalous traffic and within a small scale of time (e.g. in the magnitude of within a day), these classification mechanisms can be considerably promising (with an accuracy up to 98% and real-time potential shown). However, the results in the past remained incomplete. Some major concerns and open problems included:

1) The accuracy is still insufficient. For general purpose classification, the volume of Internet traffic is huge and is dominated by a few major traffic classes (Web-browsing in terms of flows, or Peer-2-Peer and FTP in terms of bytes or packets). In terms of composition, the major traffic is comparably easy to discriminate. However, those hard ones such as encapsulated, ambiguous, non- standard, misused or anomalous traffic, would only comprise a not large proportion. An overall error rate of several percent would mean either the major traffic is not efficiently classified; or the hard objects are unable to be classified.
2) Insufficient understanding has been taken into account of which features can be used and how to use them. Further, little understanding between the accuracy and the use of different types of features (port numbers, IP Addresses, flow-behaviour) has been presented in the past.
3) Limited practicability. Notably, real-time implementation is highly desirable, but insufficient work has been done to show the feasibility and the system performance.
4) Inability to quickly discover and correctly identify critical flows, such as intrusions and network anomalies. Intrusion detection systems would ideally require zero false-negative rate and low-latency identification of malicious traffic.
5) The incompleteness of data. Many past works selected their classification object from a few applications such as web-browser, email, FTP or multimedia applications, which can not represent fully the traffic patterns of the whole Internet.

Finally, the practical criteria for real-time online traffic classification systems can be different and remains application-centric. A trade-off potentially exists between the four metrics of system accuracy, completeness, latency and throughput. Also, algorithm would play a key role in optimising such a system where [17] has provided a lot of experience.

III. Semi-Automated Machine Learning Approach

We use a semi-automated machine learning approach to build the classifier which classifies the Internet traffic into application classes. A series of mechanisms are applied to select the feature set, the classification algorithm and the size of observation window before the final classifier is built.

Firstly the total traffic-mix is divided into classifiable objects. In this paper, the basic object of our classification system is a TCP flow, defined as a bi-directional session between two hosts with the same 5-tuple host-IP, client-IP, host-Port, client-Port and timestamp of the first packet. The server and the client of a flow are decided based on observation of SYN packet.

Two real traces are used to derive and evaluate our approach. A rich set of 248 flow features are collected from the beginning of individual network flows with different observation window sizes. Using this data, we are able to apply feature-selection algorithms to find a best subset of the features, to justify the classification algorithm. Finally, we train a model upon this feature subset, and apply this model to classify unknown flows.

Fig. 1 shows how different types of services exhibit different behaviour in two group each of two features: 1) variance of total bytes in packets (client to server) by the total number of bytes sent in initial window (client to server) and 2) count of packets with Push bit set in TCP header (server to client) by minimum segment size (client to server). One can observe that it is applicable to discriminate between the traffic flows of each class, using a combination of these features.

A. Data

Our experimental data consists of two consecutive week-days of Internet traffic with an 8 month interval. The traces were collected using a high speed monitoring box [18] installed between a research campus and the Internet. The campus is a research-facility with about 1,000 employees and is connected to the Internet via a full-duplex Gigabit Ethernet link. The datasets, Day1 and Day2, consisting of TCP traffic only, are chosen from a collection detailed in [19]. Every flow in the two datasets was hand-classified using a content-based mechanism into one of the 10 applications classes.

We left a number of TCP traffic in the datasets unconsidered: those we havent seen their start of the flow (typically those are with very long duration) and junk flows. The resulting traces contain 31 GBytes and 42 million packets in 377 thousand TCP flows in Day1, and 28 GBytes and 35 million packets in 175 thousand TCP flows in Day2. Hence, a moderately complex mix of applications exists in the traffic, as shown in Table I. Note that for some traffic classes such as Multimedia, Services, Games and Attacks, a greater amount of traffic may used other transport layer protocols such as UDP.

B. Online Classification Approach

Our online traffic classification methodology was originated in previous offline methodology where the features were collected from complete TCP flows [7] [20].

However, if we performed our analysis of flows offline, the practicability of such a system would be very much limited to merely analytical and auditing purposes. For a much wider application prospect, we moved on to investigate the problems and challenges of online real-time or near real-time traffic classification. Our concerns are no longer solely focused upon the accuracy but also upon the latency and throughput of the system. Since many applications would benefit from early identification of the traffic, our approach considers features taken from an observation window of only a few packets, rather than those based upon the entire flows.

In theory, in order to more accurately classify an object, it would require collecting more information (entropy) for classification. However, collecting more information, for example: more packets or a larger number of features may introduce higher latency and higher cost in both computation and memory usage. For the traffic classification system to be operated at near real-time with a considerable throughput, we should capture an appropriate, small number of features, and from a small number of packets and a limited duration, rather than from a complete flow. It means, in order to gain the real-time quality, an amount of information has to be sacrificed from the complete-flow objects which in theory would result in some level of degradation in accuracy.

In this way, the trade-off between accuracy, latency and throughput becomes the key of our choice of subset of features and size of the observation window and the classification algorithm.

C. Feature Selection

Our complete feature set contains 248 different features as detailed in [19], each of which has varied distribution in the datasets and has associated with it different collection / computation costs. Now what we want is to find a subset of this set of features within an upper bound of cost but containing sufficient information that leads to the desired accuracy.

In general, we limit the cost by reducing the number of features in the feature set, utilising a correlation-based filtering method. The output of such method is an approximately-best subset of features.

Our feature selection procedure is as follows: firstly, Day1 dataset is divided into 10 different entries each representing a volume of traffic at different hours of the day; then correlation-based filtering is applied to each entry. We observe that these feature subsets selected by the algorithm possess moderately-good stability, and we manually picked 10 behaviour-features which appear in at least 1/3 of the subsets, and each subset would at least contain 1/3 of these features. The intention of this criterion is to look for a best-possible feature set which can be more stable and independent to the condition of the end-to-end link. The resulting features are almost all depended on the applications on the end-hosts. It mirrors this consideration. Table II lists these features, as well as the information and complexity properties of these features.

Unrelated to the IANA port list or any prior knowledge of port-application mapping, we still adopt port pairs as two features.

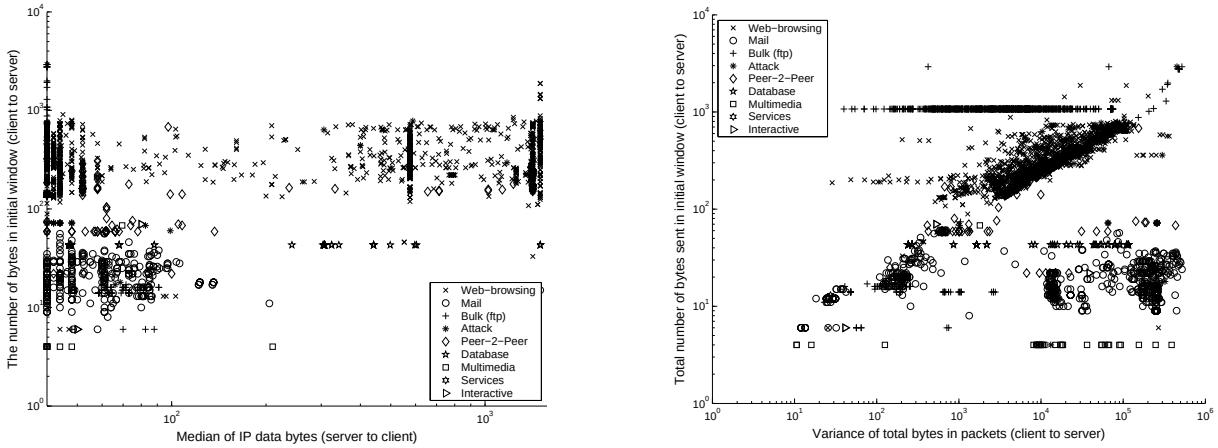


Fig. 1. Scattered plot of 5,000 random selected samples from Day1.

Class	By flow		By Packets		By Bytes		Applications
	Day1 (%)	Day2 (%)	Day1 (%)	Day2 (%)	Day1 (%)	Day2 (%)	
Web-browsing	84.558	80.198	22.529	16.383	29.438	17.623	http, https
Mail	8.682	9.384	6.777	1.884	7.904	1.763	imap, pop2/3, smtp
Bulk	3.800	6.146	69.483	80.850	60.569	79.372	ftp
Attack	0.787	0.562	0.084	0.016	0.132	0.002	portscan, worms, viruses, imail attacks
P2P	0.589	1.572	0.331	0.548	0.567	0.777	napster, kazaa, eMule, gnutella, eDonkey
Database	0.862	1.483	0.387	0.253	0.703	0.400	mysql, dbase, Oracle SQLNet
Multimedia	0.137	0.002	0.112	0.000	0.196	0.000	windows mediaplayer, realmedia
Service	0.555	0.633	0.135	0.026	0.194	0.009	X11, dns, ident, ldap, ntp
Interactive	0.027	0.021	0.156	0.040	0.285	0.053	ssh, telnet, klogin, rlogin
Games	0.002	0.0	0.006	0.000	0.013	0.000	microsoft direct play

TABLE I

COMPOSITION OF DATASETS.

Abbreviation	Description	Collection Time	Symmetrical Uncertainty	Memory Overhead	Complexity
serv_port	Server Port	S	0.8398	O(1)	O(1)
clnt_port	Client Port	S	0.0742	O(1)	O(1)
push_pkts_serv	Count of all packets with push bit set in TCP header (server to client)	D	0.2137	O(1)	O(n)
init_win_bytes_clnt	The total number of bytes sent in initial	D	0.1993	O(1)	O(1)
init_win_bytes_serv	window(client to server & server to client)	D	0.3040	O(1)	O(1)
avg_seg_size_clnt	Average segment size: data bytes divided by #packets. (client to server)	E	0.1949	O(1)	O(n)
IP_bytes_med_clnt	Median of total bytes in IP packet (client to server)	E	0.2574	O(n)	$O(n^2)$
act_data_pkt_serv	Count of packets with at least 1 byte of TCP data payload (server to client)	D	0.1680	O(1)	O(n)
data_bytes_var_clnt	Variance of total bytes in packets (client to server)	E	0.2195	O(n)	O(n)
min_seg_size_serv	Minimum segment size observed. (server to client)	D	0.2996	O(1)	O(n)
RTT_samples_serv	Total numbers of RTT samples found (server to client), see also [19]	D	0.5022	O(1)	O(n)
push_pkts_clnt	Count of all packets with push bit set in TCP header (client to server)	D	0.2360	O(1)	O(n)

TABLE II

PROPERTIES OF THE SUBSET OF FEATURES SELECTED. S=START OF OBSERVATION; D=DURING OBSERVATION; E=END OF OBSERVATION. SYMMETRICAL UNCERTAINTY GENERALLY EVALUATES THE DISCRIMINATIVE POWER OF INDIVIDUAL FEATURES WHEN USED SEPARATELY.

	Naive Bayes + kernel est.	C4.5	AdaBoost+ C4.5
Overall accuracy	92.38%± 0.35%	99.834%± 0.052%	99.816%± 0.057%
Complexity	O(features)+ O(classes)	O(tree depth)	O(tree depth×rounds)
Time taken for testing	1412s	1.5s	11s
Time taken for training	<8s	133s	1505s

TABLE III
ALGORITHMS FOR ONLINE TRAFFIC CLASSIFICATION.

The port number rules in our classifier are built upon hand-classification data, representing the association between the port numbers and those hand-classified application flows.

D. Classification Algorithm

Concerning the classification algorithm, we utilised Weka [21] toolkit to compare between different algorithms. In our approach, C4.5 decision tree exhibits highest accuracy (99.834%) among all the algorithms. AdaBoost [22]+C4.5 also exceeded 99.8%, while Logitboost, JRip, Nave Bayes Tree and Bayesian Neural Network also achieved an accuracy of more than 99%. Moreover, C4.5 decision tree has the lowest testing complexity among these algorithms. In Table III, the performance of C4.5 and Adaboost+C4.5 are shown compared with the Nave Bayes method which was used in [7]. Note that the approach in [7] did not incorporate many standard performance improvements for Naive Bayes method and the figure above is a worst case bound. A similar comparison was seen in [8] based on a simpler set of complete-flow features.

We also masked the port pairs and generated the models again purely with the 10 packet- and flow-level behaviour features with an observation window of 5 max packets. The highest total accuracy seen drops to 99.50% (C4.5 and Adaboost+C4.5). This indicates that using port numbers in combination with packet- and flow-level features in such a way provides more information than only using packet-and flow-level features.

The accuracies are from two-fold cross validation on Day1 dataset, using 12 features described in Section III.B, with an observation window of 5 packets. Time values are for training and testing 325,000 flows from Day1, collected from Weka.

E. Observation window

The requirement for online traffic classification necessitates the use of an observation window to collect the set of features from a part of a flow rather than using full flows. There are two major concerns:

1) Latency. Firstly, for real-time applications such as application-specific queuing, monitoring or anomaly detection, the latency in identifying a flow should be as low as possible. For this, it is ideal to classify the flow early in its existence, except for special purposes.
2) Cost. The use of an observation window reduces both the memory footprint in aggregating the packets and the computational complexity in calculating features. In other words, this enables a higher system throughput.

Therefore, we calculate the features with a small number of packets in the observation window, and within a limited duration. The observation window size (number of packets) is tuned based on empirical results. 1 to 10 packets are collected from the start of the flow (SYN packet), within a max-duration of 5 seconds, to compare the resulting accuracy. Fig. 2 shows the result: from a total of 5 or 6 packets collected it can achieve the highest 99.842% ten-fold cross-validation accuracy in Day1 (99.834% for corresponding two-fold cross-validation). We also notice that the small-observation-window accuracy is not lower than the accuracy of larger observation windows.

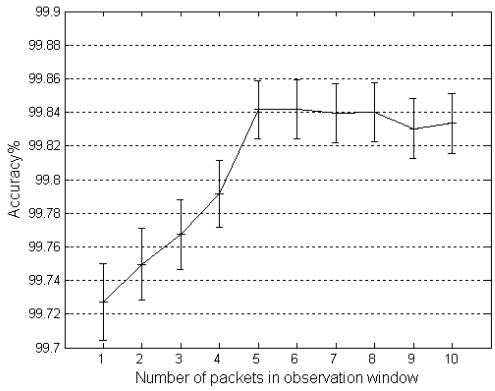


Fig. 2. Accuracy with different packet number limits (ten-fold cross-validation)

F. Complexity and memory footprint

The cost in our system pipeline has a complex composition. If we denote M as total flows currently in the memory, N as the average number of packets in a flow, and n as observation window size, then the total computational overhead would comprise all the following (also note that roughly M has a linear relationship with N):

1) For capturing packet headers and aggregating packets into flows, there will be a memory footprint of $O(M)$ to store M flows in the memory and computational complexity of at least $O(log_2 M))$ for each packet captured to find the flow it belongs to. Assuming M is proportional to N, for each flow, the complexity in aggregating the packets is roughly $O(N \times log_2 N)$.
2) For feature collection and calculation of each flow, different features in the complete set would cause a memory footprint varied from $O(1)$ to $O(n)$ and computational complexity varied from $O(1)$ to $O(n^2)$.

Roughly, the total cost of feature collection of one flow would have a super-linear relationship with the number of features in the feature set.

3) Assuming C4.5 decision tree is being used as the classifier, for classification of each flow, the computational complexity of the classifier is of the order of the average depth of the decision tree which is $O(1)$ - the same order of simple port-based rule-sets.

Clearly we know from the cost breakdown above, as the complexity of the classifier is $O(1)$, the bottleneck in this pipeline may be in reconstructing the flows and calculating the features, instead of the calculations in the classifier. However, the total cost can be bounded within $O(N \times log_2 N + n^2 \times K)$ where N is average number of packets in a flow, n is the number of packets in the observation window and K is the number of features collected.

IV. Evaluation

In this section, the model is trained from the first 5 packets seen at the start of every TCP flow. A duration limit of 5 seconds is also applied to enhance the real-time quality and system robustness. The set of features being used are the 12 features shown in Table 4. We believe it is valuable to examine the per-class quality, the temporal stability, and the relationship between accuracy and the size of training set, so as to justify the general methodology for wider practical application.

A. Per-class Accuracy

The per-class results shown in Table IV, V demonstrate the accuracy of C4.5 algorithm, using the 12 features described in the prior section. These features were taken from traffic with an imposed maximum limit of 5 packets and a maximum sampling duration of 5 seconds applied. The results are two-fold cross validations of day1 and day2 model respectively. It represents the accuracy of the model in the same network and same time when the datasets were collected.

The accuracy shown is sufficient for most of the major application classes (>99%), except for Attack class. As a general class, Attack is highly varied and complex; it contains many different subtypes, such as port-scans, various worms and viruses. This complex variety leads to a number of false predictions to and from other classes, e.g. to Web-browsing and from Mail in Day1 and Day2 respectively. However, the results show that these models can still achieve either good precision or good recall for Attack class over shorter time periods (e.g. the same day).

B. Temporal Stability

We may assume that a general traffic model cannot be as good as one which is closely up-to-date and specific to one network location, but we believe it should work moderately well with most conditions. In order to evaluate our model as general purpose traffic model, the temporal (i.e. for different time) stability of the model is further examined. This illustrates the quality of this model after a significant period of time.

Class	Precision (%)	Recall (%)
Web-browsing	99.8891	99.9538
Mail	99.9703	99.9888
Bulk (ftp)	99.3522	99.7612
Attack	96.2542	83.2273
Peer-2-Peer	97.2588	97.7225
Database	99.6782	99.7853
Multimedia	99.7753	99.7753
Service	100.0	99.6124
Interactive	100.0	100.0

TABLE IV
PER-CLASS TEN-FOLD CROSS VALIDATION RESULT FOR DAY1.

Class	Precision (%)	Recall (%)
Web-browsing	99.9865	99.9915
Mail	99.5364	96.4310
Bulk (ftp)	99.8518	99.8888
Attack	60.8144	92.2999
Peer-2-Peer	99.3098	99.0221
Database	99.8848	99.8465
Service	99.5520	100.0
Interactive	97.2222	97.2222

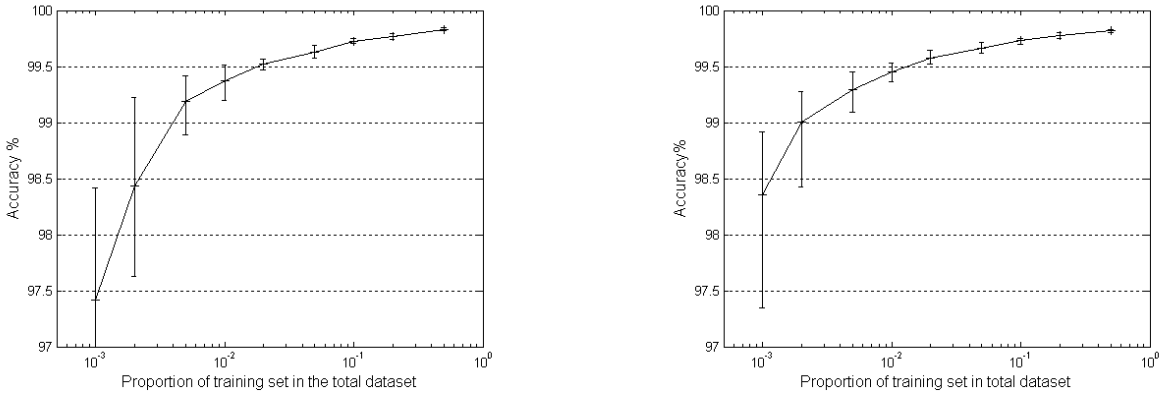
TABLE V
PER-CLASS TEN-FOLD CROSS VALIDATION RESULT FOR DAY2. MINOR CLASSES WITH NO MORE THAN 5 FLOW SAMPLES ARE NEGLECTED.

For temporal stability we use the model generated with Day1 dataset to classify the Day2 dataset, which was collected 8 months thereafter. The results are shown in Table VI. We observe the accuracies of most of the major classes still maintain to be at a high level; however, the temporal stability for Attack class is 0. This reflects the fact that Attack class is a special type of traffic that is more dynamic than other classes and changes completely over a long period of time. However, though it may not be possible to provide a long-term stable traffic model for Attack class, there can be alternative solutions such as:

1) By re-training the traffic model on a regular basis;
2) By combining this with other mechanisms such as clustering methods or host/port profiling [15] [16];
3) By providing a feedback channel from deep-checking the upcoming traffic to dynamically updating the flow model.

C. Training Data

As noted in Section IV.B, although not significant, the temporal decay is still visible after a period of 8 months. Concerning the practicality of such a classification mechanism, the requirement for hand-classifying the ground-truth training data in order to train and re-train the model can be a serious concern. As the labour required in hand-classification process increases along with the size of the training set, it is useful to know how much training data is required for a desired accuracy.



(a) C4.5 (b) Adaboost+C4.5

Fig. 3. Average accuracy vs size of training set.

Class	Precision (%)	Recall (%)
Web-browsing	99.9630	99.7860
Mail	94.2179	99.9939
Bulk (ftp)	83.2559	99.6479
Attack	0.0	0.0
Peer-2-Peer	92.5035	96.0884
Database	96.7181	19.2249
Service	99.6323	97.5696
Interactive	97.2222	97.2222

TABLE VI
TEMPORAL STABILITY RESULT. THE MODELS WERE BUILT WITH DAY1 DATASET, AND TESTED ON WHOLE DAY2 DATASET.

In the following results shown in Fig. 3, a proportion of the data is randomly picked out for training, and the rest for testing. Models are trained by C4.5 and AdaBoost+C4.5 respectively. The process is repeated 10 times for each proportion ratio. Error-bars shown in this figure are the minimum and maximum accuracy observed. We observe that using C4.5 decision tree, 99% total accuracy can be achieved with less than 0.5% of flow objects (1875 flows) randomly selected as training set. Moreover, AdaBoost+C4.5 can further reduce this proportion to 0.2% (750 flows). This implies that with a comparably small training set the model trained can still retain a fairly high accuracy. Further, same accuracy may be achieved with fewer training data if we could by any means, consciously select a similar amount of samples for each traffic class. This fact has more practical implications: it can be possible to use some seed data collected from a small number of hosts, or use some artificially generated application flows to model an access network.

D. Discussion

We may assert that discriminative learning technique (notably C4.5) is a great tool in this problem space. This is in common with the results in [8]. However, the methodology itself has not been specially tuned to adapt to our classification problem: the features are not being post-processed to make them more distinguishable. The final aim in this paper is the overall accuracy on the number of flows, rather than on specific traffic classes; and the use of C4.5 algorithm is still very coarse-grained. According to the no-free-lunch theorem [23], for a special type of problem the performance of a highly specialised algorithm can be much better than that of a general algorithm. Many applications would only require the identification of single traffic classes, for example, Multimedia, Game, Bulk, Peer-2-Peer or Attack. It is easier to achieve high precision and recall for a binary classification than for this multi-class classification.

Moreover, in order to tackle the temporal instability and to further adjust the model to other specific network locations, we note that the incremental learning (i.e. incrementally learning new information from datasets that consecutively become available, without access to previous datasets) mechanism such as Learn++ [24] may provide some inspiration. Additionally, it is also not difficult for other kinds of rules (e.g. based on port numbers or flow metrics) to manually merge into a decision tree model. Further investigation is required to demonstrate the feasibility of this kind of mechanism in specific applications.

At present the classification module using C4.5 only occupies a very small portion of CPU time in the pipeline. Therefore, from a view of the whole system, a large opportunity exists in which to refine the classification methodology.

V. CONCLUSION

In the Internet there is an ever-increasing volume and variety of traffic. Motivated by a desire to identify the applications of the Internet, in this paper we present a machine learning approach for network traffic classification based on traffic behaviour.

By collecting a small number of features, from a small number of packets or a short duration of a traffic flow, our approach can provide good balance on the overall performance: the accuracy, throughput and latency. It has many other advantages over traditional port and signature based systems, including the

potential to identify encrypted flows or flows using irregular ports, and the potential to tackle previously unknown applications. Finally, it has shown very promising feasibility for practical use.

A. Future Work

Behind this work, we have one clear aim which is to apply this classification scheme to practical applications. For this, there are still a number of areas where future work should further justify the feasibility and suitability.

We recognise that the challenge of finding the best possible combination of features and methodology remains highly application-specific and deserves further investigation. Long term temporal stability and spatial stability remains an important topic to establish the wider-applicability of the method but will require more training data sets.

Moreover, the classification objects in our current methodology are only TCP flows with their starts seen on the link, no matter they are complete or not. There is still a portion of traffic yet to be classified, such as UDP, ICMP and "mid-stream" TCP flows, those for which we have not observed the start of the flow. These kinds of traffic can also be tackled in a very similar way with the use of the standard timeout mechanism [25].

Also, further experiments would be carried out to extend the performance evaluation and to demonstrate the ability of handling encrypted traffic and previously unknown applications, based on more traffic traces.

Lastly, we observe the computational overhead in the pipeline has a comparably complex composition. It would be very helpful to find a cost function for the features as an input for feature selection in model-training. In this way a complexity bound can be applied to the model.

Acknowledgements

This paper is supported by EPSRC research grant GR/T10510/02. We are grateful to Grenville Armitage, Renata Teixeira and Laurent Bernaille for thought-provoking discussions. We also thank Jon Crowcroft, Ralphe Neill, Yao Zhao, Jian Zhang, Awais Awan and many anonymous reviewers for their valuable feedback.

REFERENCES

[1] A. W. Moore and D. Papagiannaki. Toward the accurate identification of network applications. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM 2005)*, volume 3431. Springer-Verlag LNCS, March 2005.

[2] Getbymail: Remote access & file sharing by mail. http://www.getbymail.com/.

[3] 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 *ACM SIGCOMM Internet Measurement Conference*, Taormina, Sicily, Italy, 2004.

[4] J. R. Quinlan. *C4.5: Program for Machine Learning*. Morgan Kaufman, 1993.

[5] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2435–2463, 1999.

[6] Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In *USENIX 13th Systems Administration Conference — LISA '99*, Seattle, WA, 1999.

[7] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of ACM Sigmetrics*, pages 50–60, 2005.

[8] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Computer Communication Review*, October 2006.

[9] T.T.T. Nguyen and G. Armitage. Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks. In *IEEE 31st Conference on Local Computer Networks*, November 2006.

[10] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *2006 ACM conference on Emerging network experiment and technology (CoNEXT06)*, December 2006.

[11] L. Bernaille and R. Teixeira. Early recognition of encrypted applications. In *Passive and Active Measurement Conference 2007 (PAM'07)*, April 2007.

[12] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Traffic classification through simple statistical fingerprinting. *SIGCOMM Computer Communication Review*, January 2007.

[13] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *In Proceedings of the 2006 SIGCOMM workshop on mining network data (MineNet '06)*, September 2006.

[14] K. Gopalratnam, S. Basu, J. Dunagan, and H. Wang. Automatically extracting fields from unknown network protocols. In *First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML06)*, June 2006.

[15] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: multilevel traffic classification in the dark. In *Proceedings of ACM SIGCOMM 2005*, pages 229–240, 2005.

[16] Thomas Karagiannis, Konstantina Papagiannaki, Nina Taft, and Michalis Faloutsos. Profiling the end host. In *Passive and Active Measurement Conference 2007 (PAM'07)*, April 2007.

[17] G. Varghese. *Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices*. Morgan Kaufman, 2004.

[18] Andrew Moore, James Hall, Christian Kreibich, Euan Harris, and Ian Pratt. Architecture of a Network Monitor. In *Passive & Active Measurement Workshop 2003 (PAM2003)*, La Jolla, CA, April 2003.

[19] A. W. Moore, D. Zuev, and M Crogan. Discriminators for use in flow-based classification. Technical Report RR-05-13, Department of Computer Science, Queen Mary, University of London, September 2005.

[20] T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on Neural Networks*, November 2006.

[21] I. H. Witten and E. Frank. *Data Mining*. Morgan Kaufmann Publishers, 2000.

[22] R. E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[23] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), April 1997.

[24] M. Muhlbaier, A. Topalis, and R. Polikar. Learn++.mt: A new approach to incremental learning. In *5th International Workshop on Multiple Classifier Systems (MCS 2004)*, June 2004.

[25] Cisco ios netflow. http://www.cisco.com/en/US/products/ps6601/prod_literature.html.