Classifying Network Protocols: A '2 Way' Flow Approach

Hurley, R., Garcia-Palacios, E., & Sezer, S. (2011). Classifying Network Protocols: A '2 Way' Flow Approach. IET Communications, 5(1), 79-89. DOI: 10.1049/iet-com.2009.0776

Published in:
IET Communications

Published in IET Communications
Received on 8th December 2009
Revised on 21st May 2010
doi: 10.1049/iet-com.2009.0776

ISSN 1751-8628

Classifying network protocols: a 'two-way' flow approach

J. Hurley E. Garcia-Palacios S. Sezer

The Institute of Electronics, Communications and Information Technology (ECIT), Queen's University of Belfast, Beltfast, UK
E-mail: J.hurley@ecit.qub.ac.uk

Abstract: The identification and classification of network traffic and protocols is a vital step in many quality of service and security systems. Traffic classification strategies must evolve, alongside the protocols utilising the Internet, to overcome the use of ephemeral or masquerading port numbers and transport layer encryption. This research expands the concept of using machine learning on the initial statistics of flow of packets to determine its underlying protocol. Recognising the need for efficient training/retraining of a classifier and the requirement for fast classification, the authors investigate a new application of *k*-means clustering referred to as 'two-way' classification. The 'two-way' classification uniquely analyses a bidirectional flow as two unidirectional flows and is shown, through experiments on real network traffic, to improve classification accuracy by as much as 18% when measured against similar proposals. It achieves this accuracy while generating fewer clusters, that is, fewer comparisons are needed to classify a flow. A 'two-way' classification offers a new way to improve accuracy and efficiency of machine learning statistical classifiers while still maintaining the fast training times associated with the *k*-means.

1 Introduction

Traffic classification is required in many security and quality of service (QoS) policies by network administrators and internet service providers (ISPs). It is the process of determining the application or protocol that has generated a flow of traffic. If a network administrator wishes to block certain protocols from entering their network (in a firewall), or an ISP tries to process different types of connections with different priority (e.g. limiting delays in real-time data), then identification of the protocol in use is key.

In recent years, the volume of traffic on the Internet has increased and the protocols utilised have evolved. This, in turn, has meant that the process of traffic classification has had to evolve to detect protocols that may intentionally or unintentionally attempt to avoid classification. No longer can the server port of a flow be relied upon to accurately identify the protocol in use. Many flows now run across ephemeral ports that are randomly generated at runtime. In a similar way, flows may choose to utilise the well-known port of a different protocol to 'trick' a port-based classifier in a process called port masquerading (e.g. Kazaa operating over port 80).

Application signatures offer an alternative to port classifications [1]. These are specific strings or byte patterns that occur in the transport layer payload of a handshake packet from a given protocol and so their identification can classify all packets within the 5-tuple flow (source and destination IP address and port number, transport layer protocol). Although application signatures are accurate, they can be computationally expensive to search for, are not applicable for detecting protocols that do not have any (known) signatures and become invalid if a protocol encrypts its payload data.

Modern research proposals have applied machine learning to flow statistics to generate classifications [2]. Such methods have been shown to function even when encryption is applied to transport layer payloads [3]. To apply machine-learning-based strategies in QoS and security policies, there are three factors that need to be considered. First, the accuracy of the classification achieved needs to be high. Second, the

processing and time costs of training/retraining and classifying new flows need to be low. Finally, as few as possible packets from a flow should be used to make a classification (referred to as 'early' classification) meaning that minimal packets are allowed to pass a network point before a policy is applied to the flow.

In this paper, we investigate the process of early statistical traffic classification using the *k*-means clustering algorithm. Statistical discriminators are described and pre-processing techniques discussed with a new approach to statistical classification referred to as 'two-way' classification proposed. A 'two-way' classification uniquely analyses a bidirectional flow as two separate unidirectional flows and combines the results to predict the protocol in use. We demonstrate how this approach can improve on other proposed *k*-means methods both in terms of accuracy and clustering costs while still obeying the three factors mentioned in relation to QoS and security-based statistical classification systems.

2 Related research

Various traffic classification methods have been proposed to overcome problems associated with port classifications and encrypted traffic. These include analysing host activity to determine the origin of possible flows connected to that host [4–7] or analysing different layers of functionality between hosts and flows on a network [8]. More active methods of traffic classification include the addition of a decoy host onto the network to determine further IP addresses and port numbers that are running a given application [9].

The use of statistical-based machine learning has shown promise in the process of traffic classification [2]. Moore *et al.* [9] proposed 249 flow discriminators and used machine learning to select those best to classify new flows [10, 11]. Similar strategies were applied in [12–16] to determine either the protocol or class of protocol that a flow is involved in. However, these methods require full information on a flow (e.g. total bytes passed) and are not applicable to real-time, early classification.

Bernaille *et al.* [17, 18] showed that classifications can be achieved if machine learning is applied to only the initial packet lengths of a flow. The first packet length becomes the first input to the machine learning algorithm, the second packet length becomes the second input and so on, up to the point where *N* packet lengths are input and can be clustered in an *N*-dimensional space. Huang *et al.* [19] examine the initial packet exchange in a similar format to Bernaille but show that considering packets in groups can improve accuracy. However, both of these methods utilise well-known port numbers in their classification strategies leaving possible gaps in the security of the system.

Erman *et al.* [20] propose a real-time classification system using flow statistics that functions in layers. The first layer classifies after eight packets, the second after 16 and so on. Therefore a classification at 16 packets can overwrite a classification at 8 packets after more information is gathered.

This paper expands on the concept of real-time early statistical classification by defining the 'two-way' classification. The 'two-way' classification aims to improve the performance of statistical classifiers by generating more accurate classifications with fewer clusters. Although some research proposals [18, 21] have tried to improve accuracy by using more complex clustering algorithms than *k*-means, the 'two-way' tries to achieve the improvement by still using *k*-means thereby maintaining the good training times associated with this clustering algorithm.

Concurrent research published by Crotti *et al.* [22] shows similar benefits when using unidirectional flows to improve the accuracy of bidirectional flow classifications. Their work utilises a 'fingerprint' classification method based upon probability density function (PDF) vectors [23] and packet length/inter-arrival time to investigate the use of statistical classification when asymmetric routing is present. This technique is then expanded upon to show how accuracy can be increased when using asymmetric routing techniques to classify bidirectional flows. Our work presents the 'two-way' classification, which focuses on optimising *k*-means-based techniques for early bidirectional classification, retaining the fast training and classification times required for online use. Our results and the results of Crotti [22] complement each other in showing that the use of unidirectional flows in bidirectional classification can be an efficient and effective way of improving statistical classification systems. We show that these improvements can be made without moving away from computationally efficient and simplistic clustering techniques.

3 Data traces

Throughout the research described in this paper, six network traces are used. These are separated into two groups of three that we refer to as 'company' and 'home users' data. The company traces have been recorded from the offices of a European telecommunications corporation, which apply their own security policies and firewall systems. In total, three traces of over 4 GB were captured in January 2008. The home users' data have been captured from a core switch on the ADSL network of a European ISP. The security policies of the company data are not present across this traffic and therefore it contains different flows and protocols (e.g. P2P). Two traces of 14 and 40 GB have been captured in March 2008. A third trace of 3 GB was taken in 2005.

The two groups of traces were split into a set containing one training trace and two test traces. These supply the input to build a classifier and then to test it. All traces were

doi: 10.1049/iet-com.2009.0776

pre-processed with application signatures [24] and port numbers to determine the protocols in use. Although the purpose of this research is to find alternatives to applications signatures, their use is justified in pre-processing because of their accuracy [1] – port numbers were considered for verification and in situations where application signatures cannot be used (e.g. encrypted HTTPS). Within the training traces the most occurring protocols were selected to act as a training sample. A sample number of flows were then selected from each protocol. To ensure a varied sample we selected flows based on differing IP addresses. The reason for this is that flows travelling between the same IP addresses using the same protocol are likely to be involved in the same task and therefore will have similar characteristics. For example, multiple HTTP flows between the same IP addresses are likely to be involved in the parallel downloading of web page data [3]. In reality, the HTTP protocol can be used for many different tasks including file transfer and video streaming [25]. Therefore adopting this approach means that our training data include a random sample of flows from each given protocol from a wide variety of end hosts (this will include different operating systems etc.). The company data were accounted for by 100 flows from each of the eight protocols with 500 samples from each of the seven protocols selected in the home users' training data. To allow experimentation for 'early' packet classification, it was ensured that all of these sample protocol flows had at least 20 packets passed in the incoming (server to client) and the outgoing (client to server) direction. The protocol selections in the company data are shown in Table 1 with those for the home users' data given in Table 2. Also included is the number of flows from each of the protocols found within the test traces.

Table 1 Company trace protocols

Reference	Description	Test 1 flows	Test 2 flows
HTTP	used for passing hypertext documents (e.g. web pages)	11 591	8476
HTTPS	encrypted HTTP channel	2145	1269
NETBIOS	allows applications on separate machines to communicate	1973	2307
SMB	used to share resources on Windows	7743	9795
ORACLE	oracle database access	502	323
EPMAP	end point mapper for remote management of services	612	558
LDAP	queries to directory services	1442	1220
SQL	access to SQL databases	152	140

Table 2 Home users' trace protocols

Reference	Description	Test 1 (2008) flows	Test 2 (2005) flows
BITTORRENT	P2P file swapping protocol	16 830	32
EDONKEY	P2P file swapping protocol	2810	259
GNUTELLA	P2P file swapping protocol	2057	64
HTTP	used for passing hypertext documents (e.g. web pages)	5465	1177
HTTPS	encrypted HTTP channel	498	45
MSN	instant messaging protocol from Microsoft	70	30
POP3	used to receive emails from a server	212	12

Although other protocols (e.g. SMTP) were found within our training data, too few examples of the required flow lengths (20 packets) existed to justify their inclusion within the training samples. Therefore the protocol choices presented in Tables 1 and 2 are based upon availability within our training data. If the system proposed in the following sections is to be applied to other networks then it will need training to fit the needs of that network with samples of the expected protocols. However, in this research the training and test traces (Tables 1 and 2) are used on new proposals and the previously published techniques. Therefore they allow direct comparison between the results of different strategies.

4 Classification through clustering

The use of machine learning, and in particular, clustering for traffic classification has two phases. In the training phase, training data are used to generate clusters in an N-dimensional space. These clusters are then defined to represent a given protocol. In the classification phase, new flows are mapped to the same N-dimensional space and their cluster determined. The protocol that defines this cluster classifies the flow.

In this research the k-means clustering is utilised. k-means is a form of unsupervised learning in that it is able to generate clusters without any knowledge of the 'true classification' of the input data. Each entry into the k-means algorithm contains N tuples (N flow statistics). Also input to the algorithm is a value C indicating the number of clusters to generate. k-means randomly assigns the C clusters with cluster centroids (centre coordinates of the cluster in N-dimensions) and maps all training data to their closest cluster. In an iterative manner, the algorithm then adjusts the centroids to the coordinate at the centre of all training points assigned to it. The training points are then reassigned to their closest cluster centroid in relation to the new centroid positions. The process continues until all data points are mapped to the same cluster for two iterations in a row. The closest centroid to an input data point is calculated by the Euclidean space equation as shown in (1)

$$\text{dist}(i, c) = \sqrt{\sum_{j=1}^{n} (i_j - c_j)^2} \tag{1}$$

where i is an input variable of n dimensions and c is a cluster of n dimensions.

The output of k-means is a set of C cluster centroids and the protocol that they define. This involves supervised learning where the protocol that generates each training data flow needs to be known. Then the cluster is defined by the highest frequency of training data protocols mapped to it. In the classification phase, a newly input data point finds its closest cluster centroid (defining protocol) through further use of the Euclidean space (1).

The k-means was selected to perform classification because of its relative simplicity and fast training time compared to other clustering algorithms – it is $O(iCn)$ where 'n' is the number of flows input and 'i' is the number of iterations required (which is normally low) [18]. This choice is supported in research carried out by Erman *et al.* [21] into the best machine learning algorithm for traffic classification. They show that k-means produces similar accuracy to other methods while having a far superior training time – satisfying our requirement for the classifier to have fast training/retraining. Fast training and retraining of a classification system is vital to its success. In Section 3 it was shown that traffic captured on different networks can be dominated by different protocols. It is likely that different statistical identifiers will exist to best distinguish between different groups of protocols and so separate training will be required for different networks. It is also likely that protocols utilising a given network will vary over time meaning that statistical identifiers may become more or less prominent. Therefore a statistical classifier must be retrained on a regular basis to perform best at a network point. This is a fundamental reason for our attempt to improve the accuracy of k-means clustering strategies rather than looking to other more complex clustering schemes with significantly longer training times [21].

4.1 Parameters for clustering

To detect protocols using flow statistics, certain parameters or discriminators need to be proposed. It is determined that there are three different groups of discriminators: packet length information, packet time information and transport layer information (recorded in the transport layer header of the packet/s). Initial experiments on clustering suggested that time information was not useful in separating flows from our sampled protocols. This supports findings in [14, 18, 20] where packet times were also found to be of little use in clustering.

Sixty-two discriminators based on packet length statistics and transport layer header information were proposed. Transport layer information has been considered in data mining strategies [9] but has not been considered in many real-time proposals. Thirty-one of the 62 proposed discriminators gather statistics for a flow in the outgoing (client-to-server) direction based on the sender of the initial SYN message, with the other 31 discriminators accounting for the incoming (server to client) direction. These 31 statistics are described in Table 3.

4.2 Pre-processing of data

To successfully apply k-means clustering using Euclidean distance, pre-processing of data is vital to allow the statistics to be comparable to each other. For example, a discriminator such as the 'number of packets with TCP payloads' was measured by incrementing a counter whereas 'throughput' was measured in bytes and will therefore contain much higher values meaning they cannot be fairly compared with a distance equation. In a similar way, a statistic such as the 'maximum packet length' is likely to have a distribution skewed towards a TCP Maximum Transmission Unit (MTU) of 1500 bytes whereas the 'minimum packet length (excluding non-payload packets)' is likely to be skewed closer to zero payload bytes. The skewing means that cluster centroids may be generated in close proximity to each other which may lead to failure to detect the statistical significance of a discriminator. To overcome the problem with skewing, Box–Cox transformations were applied to each discriminator individually meaning the significance of each could be highlighted as much as possible. Box–Cox is a power transformation strategy that attempts to make data resemble a normal distribution by using a predetermined power value (λ) as shown in (2) [26]

$$x_i(\lambda) = \frac{(x_i^{\lambda} - 1)}{\lambda} \tag{2}$$

where $\lambda \neq 0$ or $x_i(\lambda) = \ln(x_i)$, where $\lambda = 0$.

doi: 10.1049/iet-com.2009.0776

Table 3 Proposed flow discriminators

no. of packets with payloads	most occurring payload length	bytes in pushed packets	ave. IP/TCP header length
max. payload length	no. of most occurring	no. of urgent packets	difference in max./min. TCP window
no. of max. lengths	2nd most occurring payload length	bytes in urgent packets	no. of ACK piggy-backed packets
min. payload length	no. of 2nd most occurring	max. TCP window size	bytes in piggy-backed packets
no. of min. lengths	3rd most occurring payload length	no. of max. TCP window sizes	non-ACK piggy-backed packets
ave. packet length	no. of 3rd most occurring	min. TCP window size	bytes in non-piggy-backed packets
ave. payload length (only calculated on payload packets)	no. of retransmitted SYN or SYN/ACK	no. of min. TCP window sizes	no. of pure ACK packets (no payload)
throughput (bytes)	no. of pushed packets	ave. TCP window size	

In (2), x_i represents a series of discriminator values in the set of all discriminators (x) where i ranges from 1 to 62. Each series of discriminator values was transformed by the λ value that best handled the distribution skewing. The most effective power value (λ) was determined for each discriminator by increasing λ from -5 to $+5$ (in steps of 0.1) and calculating the value that maximised the logarithm of the likelihood function [26]

$$f(x_i, \lambda) = -\frac{n}{2}\ln\left[\sum_{j=1}^{n}\frac{(x_{i,j}(\lambda) - \bar{x}_i(\lambda))^2}{n}\right] + (\lambda - 1)\sum_{j=1}^{n}\ln(x_{i,j}) \quad (3)$$

where $\bar{x}_i(\lambda) = (1/n)\sum_{j=1}^{n} x_{i,j}(\lambda)$

Once the data were transformed by Box–Cox, it was then linearly transformed for each discriminator using the max/min normalisation method (4), which mapped all discriminator values to between 0 and 1. This meant that all discriminator values were directly comparable in Euclidean distance (1)

$$x_{i,j} = \frac{x_{i,j} - \min(x_i)}{\max(x_i) - \min(x_i)} \quad (4)$$

where $\min(x_i)$ and $\max(x_i)$ return the smallest and largest values in the discriminator series x_i.

4.3 Discriminator selection

After pre-processing the training data, the discriminators that performed best for the given protocols (Tables 1 and 2) were selected. To do this our version of a sequential forward selection (SFS) algorithm [14] was implemented. This algorithm starts by clustering the training data according to each discriminator individually. It then calculates the best discriminator by determining how many of the training flows are assigned to a cluster that represents that protocol. The second iteration of the algorithm uses the previously selected best parameter combined with all other discriminators individually to cluster the data. The best combination is selected and the algorithm continues by applying each remaining discriminator to the combination. When no improvement is made to the clustering of data by adding any of the remaining parameters the algorithm completes, returning the best selection of discriminators to separate the training data.

The SFS algorithm was also used to vary the packet measuring limit and the number of clusters (C). The packet measurement limit is the number of packets passed within the flow before the classification prediction is made. The measurement limit is counted in both directions of the flow meaning that if the packet limit is five then five packets are measured in both the incoming and outgoing directions (excluding SYN and SYN/ACK packets). Therefore rather than selecting a 'layer' (as in [20]) for classification, the optimum number of packets to measure in order to generate the best clustering of the training data is determined.

The SFS algorithm was used to determine the best parameters for bidirectional classification and also for unidirectional classification. Unidirectional classification is considered to only use packet statistics in either the incoming or outgoing direction but not both. Unidirectional classification has the potential to be used in the core of the network where asymmetric routing [27] may be in place. However, it also forms the basis for the 'two-way' classification. Because the recorded statistics were made up of 31 in the incoming direction and 31 in the outgoing direction and the packet limit was measured in both directions, the SFS algorithm was easily adapted to determine the best parameters for unidirectional classification.

A best compromise for the number of clusters, C, to cluster the training data was found to be 50. Experiments showed that this value allows enough clusters to make accurate predictions while being small enough to reduce computational costs in the classification phase by reducing the comparisons needed to determine a closest cluster. The best discriminator and packet limit selections are as shown in Table 4.

Table 4 indicates that the same set of discriminators is not calculated as most effective on the different training samples. This suggests that the discriminators chosen will be specific to the protocols that the system is attempting to classify. This again highlights the need for efficient training and retraining of statistical classification systems for different networks and further motivates our decision to improve k-means clustering.

4.4 'two-way' classification

The parameters described in Table 4 are optimised for early detection of our given protocol flows in bidirectional or unidirectional format. Unidirectional statistics may be used if asymmetric routing is occurring. However, if the packets in both directions of a flow are visible then statistics from both directions should be used. The 'two-way' classification is a proposed optimisation to k-means traffic clustering. Rather than classifying with bidirectional parameters, the 'two-way' classification clusters data in both the incoming and outgoing directions independently and combines the results.

In 'two-way' classification, the training phase returns C cluster centroids along with a protocol set for each, rather than a single defining protocol. This protocol set contains all the protocols mapped to the cluster in the training phase. Also returned is a further indicator of the number of occurrences of each protocol within the set – this is considered fair as the same number of samples of each protocol are used for training. In the classification phase, two protocol sets are returned for each new flow – one for their determined outgoing cluster and one for the incoming cluster. The two protocol sets are then compared with three outcomes. First, one protocol exists in the intersection of both sets in which case this protocol classifies the flow. Second, more than one flow exists in the set intersection in which case the flow is considered partially classified. Third, no protocols exist within both the sets, marking the flow as unknown by the 'two-way' classification.

If the flow is partially classified then we utilise the number of occurrences of each protocol within the set to make a decision. After experimentation with several methods, we determined that the most effective way to do this was to add the proportions of the occurrences of all set intersecting protocols. The protocols that intersect both the incoming and the outgoing sets have their proportions calculated by dividing their occurrence in each set by the total intersecting protocols occurrences in that set. The proportions for both sets are then added and the largest total represents the protocol that classifies the flow.

In cases where no protocol intersects both sets, the unidirectional flows are used to define the classification. We determined that the best method to do this is to calculate the percentage of the most occurring protocol in the incoming and outgoing sets in relation to the total

Table 4 Discriminators for classification

Trace set	Direction	Packet limit	Discriminators
company	bidirectional	6	outgoing – ave. packet length, 2nd most occurring payload length, bytes in pushed packets, no. of max. TCP window sizes
			incoming – min. packet length, second most occurring packet length, no. of pushed packets, min. TCP window size
	outgoing	3	no. of max. lengths, bytes in pushed packets
	incoming	2	no. of most occurring payload length, no. of packets with payload, bytes in pushed packets, min. TCP window size, difference in max./min. TCP window, ave. IP/TCP header length
home user	bidirectional	2	outgoing – no. of pure ACK packets, no. of max. payload lengths, min. payload length, ave. TCP window size
			incoming – min. payload length, no. of retransmitted SYN/ACK packets
	outgoing	3	min. payload length, ave. packet length, throughput, most occurring packet length, bytes in ACK piggy-backed packets
	incoming	5	max. payload length, no. of max. lengths, min. payload length, bytes in pushed packets

doi: 10.1049/iet-com.2009.0776

protocol occurrences in these sets. The largest unidirectional percentage was then used to classify the flow.

5 Results

Classifiers were trained with the bidirectional features described in Table 4 and applied to our company and home users' test traces. The 'two-way' classification was also applied through the unidirectional parameters featured in Table 4. In the classification phase, the test trace flow statistics were adapted by the Box–Cox power value and the max/min values determined in the training phase for the discriminators featured in Table 4. To determine the effectiveness of these approaches, the real-time proposals of Erman *et al.* [20] and of Bernaille [18] were also applied to our data. For Erman *et al.*, their defined packet length discriminators and pre-processing techniques were used along with their recommended cluster value of 400 and Layer 1 (eight packets) and Layer 2 (16 packets) measurement values. In the Bernaille *et al.*, experiment, their *k*-means strategy of clustering the first four packet lengths from a flow into a recommended value of 40 clusters was utilised. In the interests of fairness, Bernaille's additional port classification feature was not used. Both of the bench marking proposals utilise *k*-means clustering meaning they are directly comparable to our strategy. The results of applying Erman's and Bernaille's classification technique to our test data is given in Tables 5 and 6 with our proposals presented in Tables 7 and 8.

To visually compare the results of Tables 5–8, the average number of correct protocol classifications for each technique is calculated and displayed in Fig. 1. We also include the average results for the Erman layer classifications when the number of clusters in the *k*-means algorithm is reduced from 400 to 100 and in Bernaille when it is raised from 40 to 100. Hundred is the total number of clusters utilised in the 'two-way' technique (50 in each direction).

Table 5 Erman [20] and Bernaille [18] applied to company traces

Protocol	Company Trace 1			Company Trace 2		
Technique	Erman		Bernaille	Erman		Bernaille
	Layer 1	Layer 2		Layer 1	Layer 2	
HTTP	86.08	77.80	63.58	84.41	85.89	79.29
HTTPS	94.36	71.56	71.10	93.62	57.53	27.19
NETBIOS	94.63	81.50	99.65	96.40	93.06	99.65
SMB	97.75	81.44	99.45	98.66	87.75	99.64
ORACLE	100	91.06	99.40	100	96.59	99.39
EPMAP	98.69	95.59	99.18	97.67	94.44	97.67
LDAP	98.85	69.76	99.58	89.02	75.51	99.59
SQL	93.42	42.11	87.5	97.86	49.29	92.86

Table 6 Erman [20] and Bernaille [18] applied to home users' traces

Protocol	Home user's Trace 1			Home user's Trace 2		
Technique	Erman		Bernaille	Erman		Bernaille
	Layer 1	Layer 2		Layer 1	Layer 2	
BITTORRENT	69.32	60.62	54.17	71.88	53.13	53.13
EDONKEY	68.68	73.38	36.40	51.74	55.60	78.38
GNUTELLA	58.58	62.71	47.54	23.44	34.38	28.125
HTTP	56.14	58.24	59.40	65.76	64.66	63.55
HTTPS	65.85	53.58	33.54	75.56	60.00	80
MSN	78.57	54.29	88.57	80.00	60.00	63.34
POP3	98.58	78.30	100	75.00	66.69	100

Table 7 Our classification of company traces

Protocol	Company Trace 1				Company Trace 2			
	Out	In	Bi	two-way	Out	In	Bi	two-way
HTTP	74.15	83.25	89.50	82.45	65.60	76.26	84.66	74.59
HTTPS	83.92	95.85	95.76	94.36	62.88	93.77	93.14	88.73
NETBIOS	99.49	95.29	99.65	99.85	99.65	94.36	99.91	100
SMB	99.56	91.35	87.86	99.59	99.73	97.75	96.75	99.87
ORACLE	100	88.84	89.04	89.04	100	100	100	100
EPMAP	99.84	98.69	100	100	99.82	96.95	100	99.82
LDAP	95.21	91.61	72.68	95.21	91.48	85.74	80.25	91.48
SQL	87.5	88.82	100	96.05	92.86	92.86	97.14	99.29

Table 8 Our classification of home users' traces

Protocol	Home user's Trace 1				Home user's Trace 2			
	Out	In	Bi	two-way	Out	In	Bi	two-way
BITTORRENT	89.07	72.91	83.20	89.88	96.88	65.63	78.13	90.63
EDONKEY	72.35	84.52	86.73	91.21	50.19	77.99	59.85	70.27
GNUTELLA	44.34	73.16	62.18	63.93	35.94	53.13	54.69	42.19
HTTP	89.13	70.96	85.91	90.43	84.03	63.30	73.32	87.00
HTTPS	83.64	63.80	76.89	89.16	84.44	44.44	93.33	71.11
MSN	75.71	92.86	87.14	97.14	83.33	76.66	90	96.67
POP3	97.64	75.94	91.64	96.70	100	75	100	100

The results on the company traces in Figs. 1*a* and *b* indicate that 'two-way' classification surpasses the accuracy achieved through Erman's [20] and Bernaille's [18] strategies in most situations. When Erman is applied at Layer 1 with 400 clusters approximately the same accuracy is achieved as with the 'two-way' technique; however, the 'two-way' classification approach utilises fewer clusters (100). Although 'two-way' classification requires some extra processing to compare protocol sets, the reduced clusters mean that fewer comparisons will be required for the 'two-way' classification in real-time, increasing the traffic bandwidth that the classifier can operate at. In the home users' traces, the 'two-way' classification increases accuracy by 18% in Trace 1 (Fig. 1*c*) and 16% in Trace 2 (Fig. 1*d*) above the best results produced by the Erman strategy even when 400 clusters are used, and by 18 and 5%, respectively (Figs. 1*c* and 1*d*), over Bernaille's proposal.

From examining the Erman experiment results in Fig. 1, as expected, when more clusters are used in the training phase (400 against 100), more accurate classifications are achieved. However, it is suggested in [20] that as the layer a classification is made at is increased, the accuracy will also increase. Fig. 1 indicates that this is not the case as the Layer 1 (eight packets) classifications are more accurate in all traces than at Layer 2 (16 packets). These results support the observations in [18] that the early packets within a flow are the unique handshake messages and are best for distinguishing between different protocols in real time. For example, from the company traces in Table 4, it is shown that a Layer 1 classification with 400 clusters can correctly identify over 90% of SQL flows but only identifies 40–50% with a Layer 2 classification. This suggests that SQL has unique handshake packets that can be identified through statistical analysis but that become 'lost' or 'hidden' when more statistics are collected from further packets within the flow. These results justify our approach of determining the optimum number of packets with which to generate an 'early' classification strategy.

The Bernaille results of Fig. 1 show in all traces that, as expected, if the number of clusters is increased then the accuracy of the classification will increase. In their paper, Bernaille *et al.* [18] described the need to have as few

doi: 10.1049/iet-com.2009.0776

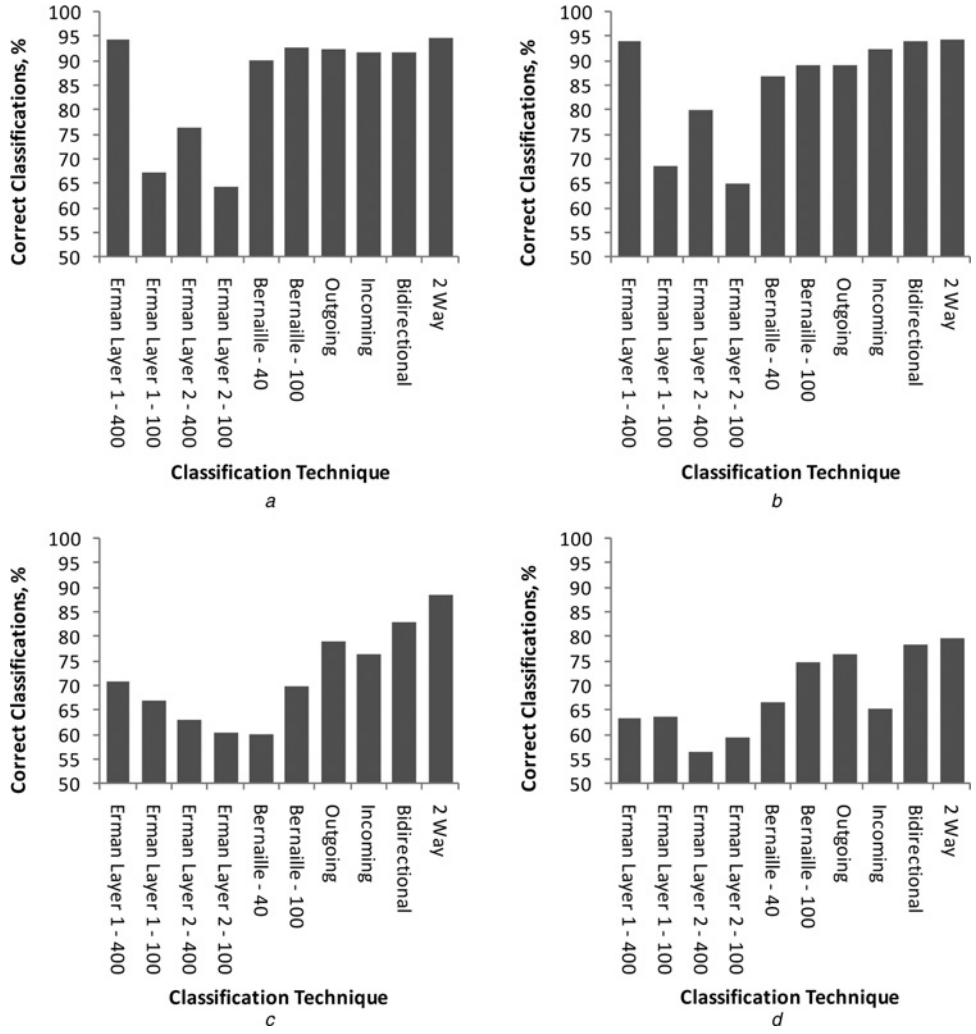

Figure 1 *Classification success of proposed techniques*

a Company Trace 1
b Company Trace 2
c Home user's Trace 1
d Home user's Trace 2

clusters as possible to improve the speed of making a classification. This is why they select a value of 40 clusters for *k*-means. Figs. 1*a* and *b* indicate that Bernaille can achieve significantly higher classification accuracy with 40 clusters than Erman can achieve with 100 clusters when tested on our company data traces. However, in all experiments the 'two-way' classification approach is shown to outperform Bernaille in terms of classification accuracy when the same number of clusters are utilised (100).

Fig. 1 also highlights the advantage of selecting statistical parameters (including transport layer information) specific to the protocols that are to be classified and individually pre-processing training data with Box–Cox and max/min normalisation. In all traces, the incoming, outgoing and bidirectional techniques perform more accurately with 50 clusters than the Erman classifications achieve with 100 clusters. In some cases they also outperform Erman's approach with 400 clusters (1*c* and *d*). It is also significant that all techniques (with the exception of Bernaille) perform significantly better on home users' Trace 1 than in Trace 2 given the fact that Trace 1 was recorded at a similar time period to the training samples whereas Trace 2 was recorded two year previous. These observations confirm that it is important to train and retrain classifiers to deal with specific protocol sets or specific periods in time to ensure that the classifier is up to date. This in turn indicates that *k*-means is a suitable algorithm to use to build such a classifier.

With the exception of outgoing classification on company Trace 1 (1*a*), the bidirectional technique always achieves better accuracy than the unidirectional strategies. However, the difference between the unidirectional and bidirectional results is small in many traces. In the company traces, both incoming and outgoing classification achieve above 89% accuracy (1*a* and *b*). This suggests that it is valid to apply this form of unidirectional classification in the core of the

network where phenomena such as asymmetric routing may occur. This proposes an alternative to that suggested in [27] whereby unidirectional flow statistics are used to predict the bidirectional statistics and classification carried out on the bidirectional information.

Combining the information gathered using incoming and outgoing parameters to form the 'two-way' classification is shown to increase classification accuracy over the bidirectional technique in all traces (over 6% in Fig. 1*c*) – for comparison we also carried out the bidirectional clustering with 100 clusters and found no significant accuracy increase over the results presented for 50 clusters. From the company trace results of Table 7, it is shown how the 'two-way' classification affected different protocols accuracy in various ways compared to bidirectional classification. HTTP showed the most significant drop in accuracy when 'two-way' classification was applied. Further analysis of the results showed that the misclassification of HTTP was because of confusion with several of the other protocols. As described in [16, 25], HTTP can be used for many different purposes including web browsing, tunnelling, file transfer etc. The 'two-way' classification relies on the unidirectional flows from a bidirectional behaving in similar ways within the early packets. In HTTP this can vary depending on what the HTTP flow is being used for (we viewed this by the more distributed clustering of HTTP in the training phase compared to other protocols). However, the 'two-way' classification has greatly increased the accuracy of protocols like LDAP and SMB (Table 7), which are likely to be only used for the one purpose (directory service queries or resource sharing). These results indicate that 'two-way' classification is more accurate in detecting flows that are involved in similar tasks as well as being generated by the same protocol, or flows that do not have generic functionality like HTTP. Therefore we have more confidence in the results of 'two-way' classification compared to other techniques.

In the home trace data (Table 8), the 'two-way' classification performs worst when detecting Gnutella flows. Gnutella flows also have lower detection rates in bidirectional, Erman and Bernaille classifications (Table 6). From analysing the results it was revealed that much of the Gnutella misclassifications (90.44% in Trace 1 and 83.78% in Trace 2) are because of confusion with HTTP. In a similar way, over 70% of the HTTP misclassifications in both traces are caused by confusion with Gnutella (the rest was confusion with HTTPS). This is because of Gnutella's use of a HTTP-like syntax for file distribution and, as with the company traces, highlights a problem with the detection of more generic protocols like HTTP that can have very different functionality. This indicates a new advance in the 'arms race' [20] between those classifying protocols and those wishing to avoid detection. Designing protocols to function with a similar (or the same) syntax as another popular protocol can help to avoid detection by a real-time statistical classifier. This is something that will need to be considered within the research community as further efforts are made to develop next generation classification systems.

6 Conclusions

This research expanded on the concept of machine learning by *k*-means clustering to identify the protocol that has generated a flow of traffic through analysing the initial flow statistics. The approach allows early identification and can overcome the drawbacks of port classification and, because only header information is used, can be applied across encrypted traffic. Adaption of the simplistic *k*-means algorithm allows for efficient retraining of a classifier (vital to its long-term use), which may be lost if more complex clustering methods are utilised. Discriminators were investigated with the importance of pre-processing techniques discussed. A 'two-way' classification was proposed, which uniquely analyses a bidirectional flow as two unidirectionals and compares the outcomes to determine the protocol that has generated the flow.

Results of tests on 'two-way' classification showed that it improved on the classification accuracy achieved in two similar *k*-means-based proposals in all cases. For both compared techniques, the 'two-way' classification showed an accuracy improvement of approximately 18% in at least one trace, even when the same number of clusters were generated (100). The 'two-way' classification advances machine-learning-based statistical classification by showing that accuracy and cluster efficiency can be significantly improved without having to move away from *k*-means clustering, thereby sacrificing the fast training times it offers.

7 References

[1] SEN S., SPATSCHECK O., WANG D.: 'Accurate, scalable in-network identification of P2P traffic using application signatures'. WWW2004, 2004

[2] NGUYEN T.T., ARMITAGE G.: 'A survey of techniques for internet traffic classification using machine learning', *IEEE Commun. Surveys Tutorials*, 2008, **10**, (4), pp. 56–76

[3] BERNAILLE L., TEIXEIRA R.: 'Early recognition of encrypted applications'. Passive and Active Network Measurement, 2007 (*LNCS*, **4427**), pp. 165–175

[4] KARAGIANNIS T., BROIDO A., FALOUTSOS W., CLAFFY K.: 'Transport layer identification of P2P traffic'. Proc. Fourth ACM SIGCOMM Conf. on Internet Measurement (IMC 2004), Italy, October 2004, pp. 121–134

[5] JOHN W., TAFVELIN S.: 'Heuristics to classify internet backbone traffic based on connection patterns'. Int. Conf. on Information Networking, ICOIN 2008, January 2008, pp. 1–5

doi: 10.1049/iet-com.2009.0776

[6] CONSTANTINOU F., MAVROMMATIS P.: 'Identifying known and unknown peer-to-peer traffic'. Fifth IEEE Int. Symp. on Network Computing and Applications, 2006, pp. 93–102

[7] PERÉNYI M., DANG T., GEFFERTH A., MOLNÁR S.: 'Identification and analysis of peer-to-peer traffic', *J. Commun.*, 2006 **1**, (7), pp. 36–46

[8] KARAGIANNIS T., PAPAGIANNAKI K., FALOUTSOS M.: 'BLINC: multilevel traffic classification in the dark'. Proc. 2005 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, Philadelphia, Pennsylvania, USA, 22–26 August 2005

[9] OHZAHATA S., HAGIWARA Y., TERADA M., KAWASHIMA K.: 'A traffic identification method and evaluations for a pure P2P application'. Passive and Active Network Management, PAM 2005, Boston, MA, 2005 (*LNCS*, **3431**), pp. 55–68

[10] MOORE A., ZUEV D., CROGAN M.: 'Discriminators for use in flow-based classification'. Technical Report, Intel Research, 2005

[11] MOORE A., ZUEV D.: 'Internet traffic classification using Bayesian analysis techniques'. ACM SIGMETRICS, Banff, Canada, June 2005

[12] ZUEV D., MOORE A.: 'Traffic classification using a statistical approach', *Lect. Notes Comput. Sci.*, 2005, **3431**, pp. 321–324

[13] JUNIOR G.P.S., MAIA J.E.B., HOLANDA R., DE SOUSA J.N.: 'P2P traffic identification using cluster analysis'. Global Information Infrastructure Symp., GIIS 2007, July 2007, pp. 128–133

[14] ZANDER S., HGUYEN T., ARMITAGE G.: 'Automated traffic classification and application identification using machine learning'. Proc. IEEE Conf. on Local Computer Networks 30th Anniversary, LCN, 2005

[15] ROUGHAN M., SEN S., SPATSCHECK O., DUFFIELD N.: 'Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification'. Proc. Fourth ACM SIGCOMM Conf. on Internet Measurement, 2004, pp. 135–148

[16] MCGREGOR A., HALL M., LORIER P., BRUNSKILL J.: 'Flow clustering using machine learning techniques'. Proc. Fifth Passive and Active Measurement Workshop, PAM, April 2004

[17] BERNAILLE L., TEIXEIRA R., AKODJENOU I., SOULE A., SALAMATIAN K.: 'Traffic classification on the fly', *ACM SIGCOMM Comput. Commun. Rev.*, 2006, **36**, (2), pp. 23–26

[18] BERNAILLE L., TEIXEIRA R., SALAMATIAN K.: 'Early application identification'. Proc. Int. Conf. on 2006 ACM CoNEXT Emerging Networking Experiments and Technologies, Article number 6, Lisboa, Portugal, 2006

[19] HUANG N., JAI G., CHAO H.: 'Early identifying application traffic with application characteristics'. IEEE Int. Conf. on Communications, ICC'08, May 2008, pp. 5788–5792

[20] ERMAN J., MAHANTI A., ARLITT M., COHEN I., WILLIAMSON C.: 'Offline/realtime traffic classification using semi-supervised learning', *Perform. Eval.*, 2007, **64**, (9–12), pp. 1194–1213

[21] ERMAN J., ARLITT M., MAHANTI A.: 'Traffic classification using clustering algorithms'. Joint Int. Conf. on Measurement and Modelling of Computer Systems, 2006 SIGCOMM Workshop on Mining Network Data, Pisa, Italy, 2006, pp. 281–286

[22] CROTTI M., GRINGOLI F., SALGARELLI L.: 'Impact of asymmetric routing on statistical traffic classification'. IEEE Globecom 2009, Honolulu, USA, 30 November–4 December 2009

[23] CROTTI M., DUSI M., GRINGOLI F., SALGARELLI L.: 'Traffic classification through simple statistical fingerprinting', *Comp. Commun. Rev.*, 2007, **37**, (1), pp. 5–16

[24] L-7 Filter: http://l7-filter.sourceforge.net/

[25] LI W., MOORE A.W., CANINI M.: 'Revisiting HTTP traffic in the new age'. SIGCOMM'08 Poster Extended Abstract, August 2008

[26] BOX G.E.P., COX D.R.: 'An analysis of transformations', *J. Royal Stat. Soc.*, 1964, **26**, (2), pp. 211–252

[27] ERMAN J., MAHANTI A., ARLITT M., WILLIAMSON C.: 'Identifying and discriminating between web and peer-to-peer traffic in the network core'. Proc. 16th Int. Conf. on World Wide Web, Banff, Alberta, Canada, 2007, pp. 883–892