Automatic Discovery of Network Applications: A Hybrid Approach

Mahbod Tavallaee⋆, Wei Lu†, Ebrahim Bagheri‡, and Ali A. Ghorbani⋆

Information Security Centre of Excellence, University of New Brunswick ⋆
Q1 Labs Inc., Fredericton, New Brunswick, Canada †
Institute for Information Technology, National Research Council Canada ‡

Abstract. Automatic discovery of network applications is a very challenging task which has received a lot of attentions due to its importance in many areas such as network security, QoS provisioning, and network management. In this paper, we propose an online hybrid mechanism for the classification of network flows, in which we employ a signature-based classifier in the first level, and then using the weighted unigram model we improve the performance of the system by labeling the unknown portion. Our evaluation on two real networks shows between 5% and 9% performance improvement applying the genetic algorithm based scheme to find the appropriate weights for the unigram model.

1 Introduction

Accurate classification of network traffic has received a lot of attentions due to its important roles in many subjects such as network security, QoS provisioning, network planning, class of service mapping, to name a few. Traditionally, traffic classification relied to a large extent on the transport layer port numbers, which was an effective way in the early days of the Internet. Port numbers, however, provide very limited information nowadays due to the increase of HTTP tunnel applications, the constant emergence of new protocols and the domination of P2P networking applications. An alternative way is to examine the payload of network flows and then create signatures for each application. It is important to notice that this approach is limited by the fact that it fails to detect 20% to 40% of the network flows because of the following reasons:

- Lack of regular update to support new releases of applications.
- Not being aware of all types of applications being developed around the world.
- Great deal of variation in P2P applications.
- The emergence of encrypted network traffic.

Observing daily traffic on a large-scale WiFi ISP network, Fred-eZone[1], over a half year period (from June 2007 to December 2007), we found that there are about 40% of network flows that cannot be classified into specific applications by the state-of-the-art signature-based classifiers, i.e., 40% network flows are labeled as unknown applications.

[1] http://www.fred-ezone.com

Addressing the limitations of the signature-based approach, we propose a hybrid mechanism for online classification of network flows, in which we apply two signature-based methods sequentially. In the first level, we employ a powerful traffic classification software, MeterFlow[2], to detect the network flows containing known applications signatures. We then apply a machine learning based approach to examine the payloads of network flows. However, instead of searching for the exact signatures, we extract the characteristics of payload contents using the Weighted Unigram Payload Model. The main contributions of this paper are:

- Employing the unigram of packet payloads to extract the characteristics of network flows. This way similar packets can be identified using the frequencies of distinct ASCII characters in the payload.
- Applying a machine learning algorithm to classify flows into their corresponding application groups. For the experiments we used the J48 decision tree, the Weka implementation of C4.5 [1], since it demonstrated to have high accuracy while maintaining a reasonable learning time. In addition, decision trees are shown to have a reasonable height and need a few comparisons to reach a leaf which is the final label, and therefore have a reasonably short classification time.
- Assigning different weights to the payload bytes to calculate the unigram distribution. We believe that depending on the applications those bytes that include signatures, should be given higher weights.
- Applying genetic algorithm to find the optimal weights that results in improvement in the accuracy of the proposed method.

The rest of the paper is organized as follows. Section 2 introduces the related work, in which we provide a brief overview of signature-based traffic classiffication approaches. Our proposed hybrid traffic classification scheme will be explained in Section 3. Section 4 summarizes the unigram payload model. In Section 5, we formally define our problem and explain how we apply genetic algorithms to improve the accuracy of our proposed traffic classification method. Section 6 presents the experimental evaluation of our approach and discusses the obtained results. Finally, in Section 7, we draw conclusions.

2 Related Work

Early common techniques for identifying network applications rely on the association of a particular port with a particular protocol [2]. Such a port number based traffic classification approach has been proved to be less effective due to: 1) the constant emergence of new peer-to-peer networking applications that IANA does not define the corresponding port numbers[3]; 2) the dynamic port number assignment for some applications (e.g. FTP); and 3) the encapsulation of different services into a same application (e.g. chat or steaming can be encapsulated into the same HTTP protocol). To overcome this issue, there have been recently significant contributions towards traffic classification. The

[2] http://www.hifn.com

[3] http://www.iana.org/assignments/port-numbers

most currently successful approach is to inspect the content of payloads and seek the deterministic character strings for modeling the applications. For most applications, their initial protocol handshake steps are usually different and thus can be used for classification. Moreover, the protocol signatures can be modeled through either public documents such RFC or empirical analysis for deriving the distinct bit strings on both TCP and UDP traffic.

In [3], Gummadi et al. develop a signature model for KaZaA workload characterization through analyzing a 200-day trace of over 20 tera bytes of Kazaa P2P traffic collected on a campus network. In [4], Sen et al. analyze the application layer protocols and then generate the signatures of a few P2P applications. Although the protocol semantic analysis improves the accuracy of signatures, it makes the real-time analysis of the backbone traffic impossible since the underlying assumption is that every packet is being inspected. In their consequent work [5], Sen et al. examine available specification and packet-level traffic traces for constructing application layer signatures, and then based on these signatures, P2P traffic are filtered and tracked on high-speed network links. Evaluation results show that their approach obtains less than 5% false positives and false negatives.

To have a better understanding of this approach, Table 1 illustrates the signatures of 11 typical applications in which to print the signatures, alphanumeric characters are represented in the normal form, while non-alphanumeric ones are shown in the hex form starting with "0x".

Table 1. Payload signatures of some typical network applications

Application	Payload	Offset	Signatures
Bit Torrent	source	1	BitTorrent
HTTP Image Transfer	destination	4	image/
HTTP Web	source	0	GET
Secure Web	destination	0	0x16 0x03
MSN Messenger	source	0	MSG
MS-SQL	destination	0	0x04 0x01 0x00 0x25 0x00 0x00 0x01 0x00 0x00 0x00 0x15 0x00 0x06 0x01 0x00 0x1B
POP	destination	0	+OK
SMTP	source	0	EHLO
Windows File Sharing	destination	4	|FF|SMB
Yahoo! Messenger	destination	0	YMSG

As can be seen in Table 1, each application has a unique set of characters to be identified. *Secure Web* traffic can be identified by searching the hex string "1603" in the beginning of a flow payload. Similarly *Yahoo! Messenger* traffic can be detected by finding the ASCII string of "YMSG" in the payload. However, these signatures do not necessarily start from the beginning of the payload. For example, to identify *HTTP Image Transfer* traffic, we should search for the string "image/" starting from the 5^{th} byte of the payload. This starting point is referred as *offset* in Table 1. Moreover, it is important to know which side of the connection, client or server, produce the signature. For example, the signature to detect *HTTP Web* is in the source payload, i.e., the ASCII string "GET" is sent by the client, initiator of the connection, to the server. This information helps signature-based methods to improve their performance by looking at a fewer number of signatures in either the source or destination payloads.

Although this approach has a high accuracy for identifying network applications, it fails to detect 20% to 40% of the network flows. Some of the reasons that cause this problem are:

- Whenever a newer version of an application is released, all the signatures should be updated which usually cannot be done immediately.
- There are hundreds of applications being developed by small groups around the world which networking companies are not aware of.
- Peer to peer (P2P) applications employ lots of protocols with different signatures to prevent their traffic being filtered by organizations. Usually these signatures are variants of famous P2P applications (e.g., Bit Torrent, Gnutella, eDonkey) with small changes and kept confidential by crime organizations.
- Encrypted traffic is one the biggest challenges traffic classifiers are facing. Data encryption is becoming more popular with the growth of commercial services on the Internet. Besides, it is widely used to hide malicious activities such as botnet traffic.

In the next section, we propose our hybrid method which is capable of detecting new releases and variants of existing applications. Furthermore, newly created applications and different variant types of P2P software will be classified in a relevant category with similar characteristics. However, detecting encrypted traffic still remains as a challenging issue.

3 Hybrid Traffic Classification Scheme

As explained in the previous section, to overcome the shortcomings of port-based traffic classification, researchers have proposed new approaches based on the examination of payload content. These signature-based methods are able to detect a wide range of applications providing the specific signatures. Besides, They are fairly accurate and generate almost no false positives. These advantages have made the signature-based traffic classifiers very popular. However, conducting a thorough analysis of state-of-the-art signature based classifiers, we observed that depending on the type of network (e.g. Universities, ISPs, Enterprises) between 20% to 40% of the traffic is still unknown. This unknown traffic mainly consists of new applications, variation of old applications or encrypted traffic. To overcome this issue, we propose a hybrid mechanism for online classification of network flows, in which we apply two signature-based methods sequentially. In the first level, we employ a powerful traffic classification software, MeterFlow, to detect the network flows containing known applications signatures. We then apply a machine learning based approach to examine the payloads of network flows. However, instead of searching for the exact signatures, we extract the characteristics of payload contents using the Weighted Unigram Model.

In order to achieve reliable results, the machine learning based classifier needs to be provided with an up-to-date training set. To this end, we have defined learning time intervals, e.g. 1 day, at the end of which the classifier will be trained by the latest training set. This training set is composed of known flows labeled by the signature-based traffic classifier in the previous time interval. Figure 1 illustrates the structure of the hybrid traffic classifier. In the first interval which we do not have any training set, we only rely on the labels from MeterFlow. The flows with known labels will be used as the training set for the weighted unigram classifier in the next time interval. During the second time interval, flows are given to MeterFlow to do the labeling. The unknown flows will then be given to the weighted unigram classifier to be labeled based on the learned information in the previous time interval. Finally, the known flows from MeterFlow and the labeled flows by the weighted unigram classifier will be mixed together and reported to the administrator as the application labels.

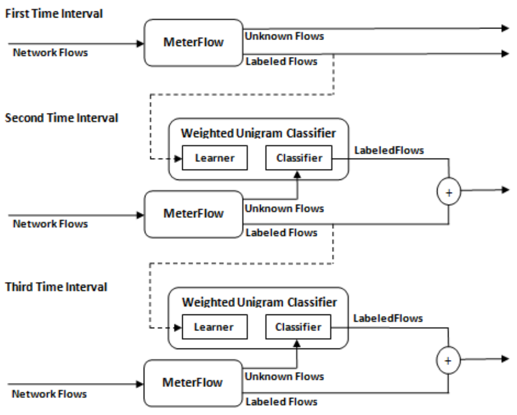
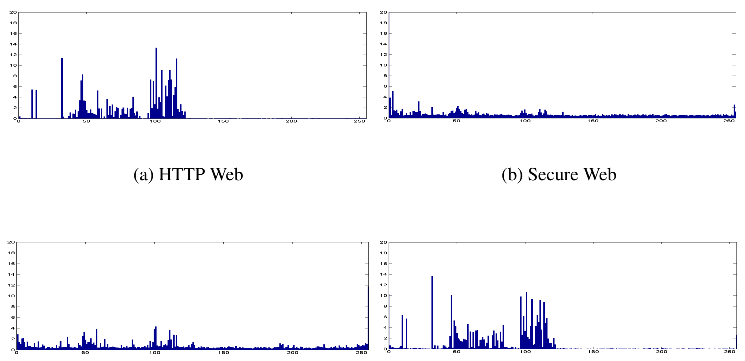


Fig. 1. General structure of the hybrid traffic classifier

4 Unigram Payload Model

N-grams are language-independent means of gauging topical similarity in text documents. Traditionally, the n-grams technique refers to passing a sliding window of n characters over a text document and counting the occurrence of each n-gram. Being first introduced by Damashek [6], this method is widely employed in many language analysis tasks as well as network security.

Applying the same idea on network packets, one can consider unigram (1-gram) of a network packet as a sequence of ASCII characters ranging from 0 to 255. This way similar packets can be identified using the frequencies of distinct ASCII characters in the payload. In order to construct the unigram payload distribution model, we extract the first M bytes of the payload and count the occurrence frequency of each ASCII character. However, since some of the applications bare their signatures in the source payload such as *HTTP Web* and some of them have their signatures in the destination payload like *Secure Web*, we consider source and destination payloads as separate pieces of information. In other words, each ASCII character has two frequency values, one for the source payload (data sent by the client, initiator of the connection, toward the server) and one for the destination payload (data received by the client from the server).



(a) HTTP Web

(b) Secure Web

(c) Bit Torrent

(d) SMTP

Fig. 2. Average unigram distribution of source payload

By observing and analyzing the known network traffic applications, labeled by a signature-base classifier called MeterFlow, over a long period on large-scale WiFi ISP network, we found that the unigram distribution of source and destination payloads can be used as a powerful tool to detect the applications. Figure 2 shows the unigram distribution of several applications namely, HTTP Web, Secure Web, SMTP, POP, Bit Torrent, Oracle. The x axis in the figures is the ASCII characters from 0 to 255, and the y axis shows the frequency of each ASCII character in the first 256 bytes of the source payload. As it can be seen in Figure 2, text-based applications (e.g., *HTTP Web*) can be easily distinguished from binary applications (e.g., *Bit Torrent*), as well as encrypted applications (e.g., *Secure Web*) since they use only a portion of ASCII characters, especially alphanumeric ones. Moreover, having an exact investigation of similarly behaving protocols such as *HTTP Web* and *SMTP*, we can still separate them based on the most frequent characters.

After applying the unigram distribution model on network flows, we can map each flow to a 512-tuple feature space $\langle f_0, f_1, ..., f_{511} \rangle$ such that f_0 to f_{255} shows the frequencies of corresponding ASCII characters in the source payload. Similarly, f_{256} to f_{511} hold the frequencies in the destination payload.

Having specified the applied network features, we focus on the selection of a high-performance classifier. In order to choose an appropriate classifier, we selected some of the most popular classification methods implemented by Weka [7] and performed some evaluations to compare the accuracy, learning time, and classification time. We finally selected the J48 decision tree, the Weka implementation of C4.5 [1], since it has a high accuracy while maintaining a reasonable learning time. In addition, decision trees are shown to have a reasonable height and need fewer comparisons to reach a leaf which is the final label, and therefore have a very short classification time. Taking advantage of the J48 decision tree as a traffic classifier, we evaluated our proposed method on two real networks.

Although our evaluations on the two networks showed promising results, we still believe that the performance can be improved by assigning different weights to the payload bytes based on the degree of importance. However, finding the appropriate weights is a challenging task. To this end, we employ a genetic algorithm based scheme to find the weights.

In the next section, we formally define our problem and explain how we apply genetic algorithms to improve the accuracy of our proposed traffic classification method.

5 Problem Formulation

In this section, we formally describe how the network application discovery problem can be performed through the combination of genetic algorithms and decision trees. Essentially, we formulate the network application discovery problem as a classification problem, i.e., given the values for a specific set of features extracted from the network flows, we identify the possible application that has generated this payload using a statistical machine learning techniques (decision trees).

As was mentioned earlier, payload can be viewed as a multidimensional vector, where in the context of a unigram analysis of the payload, the frequency of each ASCII character in the payload represents one feature in the vector space. For instance, if the character '∞' is repeated 20 times in the payload, the value for the 236^{th} dimension of the representative vector would be 20[4]. So with this simple yet intuitive formulation of the features of the payload vector space, it is possible to construct a learning classification machine that operates over features that are the frequency of each ASCII character in the payload. We formally describe this as follows:

Definition 1 *Let $\kappa_0^s, ..., \kappa_{255}^s$ be the set of 256 available ASCII characters in the source payload and $\kappa_0^d, ..., \kappa_{255}^d$ be those in the destination payload, and Ψ be a given payload generated by an application $\lambda \in \Lambda$, Λ being the set of all known network applications. We denote $\nu_{\kappa_i^{s/d}}(\Psi)$ as the frequency of $\kappa_i^{s/d}$ in Ψ. Further, $\Im : \Psi \rightarrow \Lambda$ is a learning classification machine that maps a payload Ψ with frequency ν to a network application such as λ.*

Based on this definition, we need to employ a classifier that infers the correct network application label given the features from the payload. To achieve this we employ the J48 decision tree learning algorithm [1] that builds a decision tree classifier from a set of sample payloads and their corresponding network application labels. The learned decision tree classifier will enable us to find the possible network application label for a payload from an unknown network application. Here, the learned J48 classifier is our required $\Im$ mapping function.

[4] 236 being the index of ∞ in the ASCII character list.

Now lets consider the *HTTP Web* application payload. The payload starts with `GET` and continues with further detailed information about that specific connection such as the protocol version, host and others. So for this specific application, i.e. *HTTP Web*, the first few characters are representative enough of the application; therefore, by just looking at these 3 characters we are able to identify the signature for this application and easily assert that this payload has been generated by the *HTTP Web* network application. Similarly, other network applications have specific signatures that are present in some designated positions in the payload. So, it can be inferred that some positions in the payload are more *discriminative* in the process of classifying the payloads for their generating network applications. For this reason, it is important to place more weight on the features that appear in these more important positions. We achieve this through a weighted scheme over the features.

Definition 2 *Suppose Ψ is a given payload of length η. We let Ψ_p, $p \in [0, \eta]$ denote the p^{th} position in the payload Ψ, and $\omega(\Psi_p)$ denote the weight (importance) of position p in the payload. The weighted feature of the vector space is defined as:*

$$\omega\nu_{\kappa_i^{s/d}}(\Psi) = \sum_{p \in P} \omega(\Psi_p) \tag{1}$$

where P represents the positions in which $\kappa_i^{s/d}$ has appeared in Ψ.

Simply, this definition defines that each position in the payload has its own significance in the application discovery process; therefore, some of the features may be more discriminative of the applications and hence should receive a higher weight. Based on this weighting scheme, we now revise the dimensions of our vector space such that the frequency of each ASCII character observed in the payload is multiplied by the weight of the position that it was located. For instance, if ∞ is observed in positions 5 and 210, and the weight for 5 and 210 are 1 and 8, respectively, the value for the 236^{th} dimension of the representative vector would be 9 rather than simply being 2. Accordingly, we have:

Definition 3 (Extends Definition 1) $\Im^{\omega} : \Psi \rightarrow \Lambda$ *is a learning classification machine that maps a payload Ψ with frequency $\omega\nu$ to a network application such as $\lambda \in \Lambda$.*

Now, since $\Im^{\omega}$ is sensitive to the weights given to the positions in the payload, a method needs to be devised to compute the appropriate weights for the positions. For this purpose, we employ a genetic algorithm based process to find the weights.

Briefly stated, genetic algorithms are non-deterministic and chaotic search methods that use real world models to solve complex and at times intractable problems. In this method the optimal solution is found by searching through a population of different feasible solutions in several iterations. After the population is studied in each iteration, the best solutions are selected and are moved to the next generation through the applications of genetic operators. After adequate number of generations, better solutions dominate the search space therefore the population converges towards the optimal solution. We employ this process over the weights of the positions in the payload to find the optimal weights for each location. The process that we employ is explained in the following:

- The objective is that for a payload Ψ with length of η the optimal corresponding weight vector ω_O is found. So, initially a pool of random weight vectors of length η are generated: $\omega_1, ...\omega_n$;
- For each ω_i a learning classification machine ($\Im^{\omega_i}$) is developed given a set of learning instances;
- $\Im^{\omega_i}$ is evaluated based on its performance accuracy over a given set of test instances. The accuracy of $\Im^{\omega_i}$ represents the genetic fitness function; therefore, the accuracy of $\Im^{\omega_i}$ is the fitness of ω_i among the other weight vectors in the genetic algorithm pool of possible solutions;
- The best set of weights based on their fitness value are selected and are moved onto the next generation and the rest of the weight vectors are discarded;
- The genetic operators, i.e., the mutation and crossover operators, are applied on the weight vectors present in the genetic algorithm pool of possible solutions and new weight vector solution instances are generated;
- The process of the genetic algorithm is repeated for Gen generations;
- Once the algorithm reaches a steady state and stops, the weight vector with the best fitness is selected and will be used as the most appropriate weight vector (ω_O) for the application discovery process using $\Im^{\omega_O}$.

In the above process, the weight vectors that are needed for the payload are defined using the genes in the genetic algorithm and the fitness function is defined as the accuracy of the learning classification machine developed based on that specific weight vector (gene). The outcome of the genetic algorithm process provides us with the optimal set of weights for the positions of the ASCII characters in the payloads. This optimal set of weights can be used to learn a classifier that can best find the network application for new payloads whose generating application is not known.

To implement this process, we employed the JGAP (Java Genetic Algorithms Package) software package [8]. In our experiments, reported in the following section, we apply the default crossover technique implemented in JGAP with the population size of 500 evolving for 50 generations. The default crossover rate is [population size/2], and the value for mutation rate is 1/15. Moreover, we consider the first 256 bytes of source and destination payloads since the rest of the payload does not contain any signature and decreases the classifier performance by including noise data.

6 Experiments

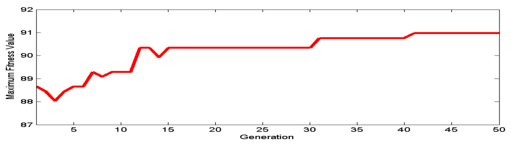
To evaluate our proposed method we prepared two data set from various networks. The first data set is prepared using the traffic captured in the Information Security Center of Excellence (ISCX) in the University of New Brunswick during 10 days. Having passed this traffic through MeterFlow we ended up with 28% remaining as unknown. For our second data set we used 3-hour captured traffic from a large-scale ISP network[5]. This data set is composed of 35% unknown flows since it is an ISP network and contains lots of peer to peer applications which are really hard to detect by the state of the art signature-based traffic classifiers.

[5] Due to privacy issues we do not reveal the name of this ISP

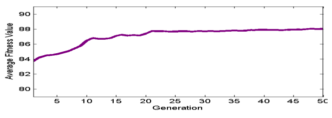
Table 2. Accuracy of the proposed method for the ISCX and ISP networks

	ISCX Network	ISP Network
Base Accuracy	81.93%	81.72%
First Generation Accuracy	83.69%	80.74
Optimal Accuracy	90.97%	86.55%

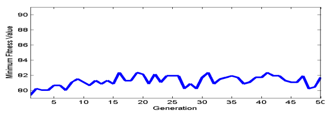
For the evaluation of the weighted unigram model, we filtered out the unknown flows from our data sets and used the rest as the training and testing set. We then extracted the unigram features of the source and destination payloads to evaluate our decision tree based classifier. However, to have enough samples of each application for training, we decided to only keep those with more than 200 records in the data set. As a result, in our first data set we ended up with 7 applications namely, *FTP*, *HTTP Image Transfer*, *HTTP Web*, *DNS*, *Secure Web*, *SSH*, *Web Media Documents*. Applying the same approach for the second data set we left with 14 applications namely, *Bit Torrent, HTTP Image Transfer, HTTP Web, DNS, Secure Web, MSN Messenger, MS-SQL, NTP, Oracle, POP, SMTP, Windows File Sharing, Yahoo Messenger, Web Media Documents*.



(a) Best fitness value



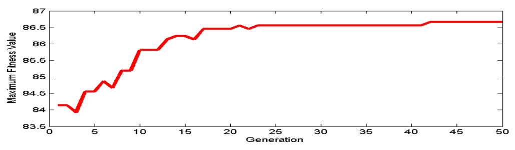
(b) Average fitness value



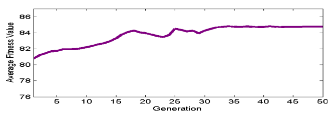
(c) Worst fitness value

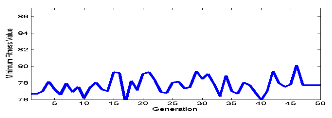
Fig. 3. Fitness value vs. generation number calculated for the ISCX network

For the experiments we applied J48 decision tree classifier. In the first step we evaluated our classifier using the payload unigram features with equal weights. For the evaluation we employed 10-fold cross-validation to obtain reliable result. We then applied the genetic algorithm technique to find the appropriate weights to obtain higher accuracy. The experiment took approximately five days to complete since the classifier model should be updated during each run of the fitness function which takes a few seconds. As illustrated in Table 2, the best fitness value, *optimal accuracy*, returned is 90.97% and 86.55% for ISCX and ISP networks, respectively. This means that we have achieved about 9% and 5% performance increase in our applied data sets, respectively. Moreover, *base accuracy* shows the performance of the classifier using equal weights and *first generation accuracy* is the average fitness value in the first generation.



(a) Best fitness value





(b) Average fitness value (c) Worst fitness value

Fig. 4. Fitness value vs. generation number calculated for the ISP network

Figures 3 and 4 show the evolution of each generation in the experiments on the ISCX and ISP networks, respectively. In the first data set, Figure 3(a), the fitness value of the best individual for each generation has a noticeable increase until generation 15. However, in the next 35 generations there is only about 1% increase in the fitness value. Similarly, as it is illustrated in 4(a), for the ISP network there is an approximate steady increase until generation 17, at which point it is apparent that almost the best possible weights have been achieved.

Although the decision tree based classifier achieves a high classification accuracy during the experimental evaluation with cross-validation, the results might be misleading due to the fact that the hybrid classifier should deal with real unknown flows. However, having no information about the unknown flows, we experienced some difficulties to evaluate the weighted unigram classifier.

To have a better estimation of the accuracy of our method, we captured corresponding traffic of five different applications separately. We then pass the traffic throuhg MeterFlow and used the known flows as a training set to detect the unknown portion. Table 3 illustrates the total accuracy of our proposed hybrid classifier for each specific applications. As can be seen in the table, the weighted unigram approach has increased the performance of signature-based approach by detecting 61% (31 out of 51) of the unknown flows.

Table 3. Accuracy of the proposed method for the ISCX and ISP networks

Applications	Total Flows	Unknown Flows	Successfully Classified	Total Accuracy
MSN Messenger	53	0	0	100%
Bit Torrent	103	27	18	91.26%
HTTP Web	121	12	6	95.04%
SMTP	74	4	2	97.30%
Windows File Sharing	91	8	5	96.70%
Total	442	51	31	95.48%

7 Conclusions

In this paper, we propose a hybrid mechanism for online classification of network flows, in which we apply two signature-based methods sequentially. In the first level, we employ a powerful traffic classification software, MeterFlow, to detect the network flows containing known applications signatures. We then apply a machine learning based approach to examine the payloads of network flows. However, instead of searching for the exact signatures, we extract the characteristics of payload contents using the Unigram Payload Model. Having a detail analysis of application signatures, we observed that the signatures are present in some designated positions in the payload, and it is important to place more weight on the features that appear in these more important positions. This is achieved through a weighted scheme over the unigram features. However, finding the appropriate weights is a challenging task. To this end, we employ a genetic algorithm based scheme to find the weights. Our evaluation on two real networks, showed promising results compared to other methods in detecting tunneled applications and variation of existing ones while maintaining a comparable accuracy in detecting old applications.

References

1. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
2. Moore, D., Keys, K., Koga, R., Lagache, E., Claffy, K.: The CoralReef Software Suite as a Tool for System and Network Administrators. In: Proceedings of the 15th USENIX conference on System administration. (2001) 133–144
3. Gummadi, K., Dunn, R., Saroiu, S., Gribble, S., Levy, H., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. ACM SIGOPS Operating Systems Review **37**(5) (2003) 314–329
4. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment. (2002) 137–150
5. Sen, S., Spatscheck, O., Wang, D.: Accurate, scalable in-network identification of p2p traffic using application signatures. In: Proceedings of the 13th international conference on World Wide Web. (2004) 512–521
6. Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science **267**(5199) (1995) 843
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA Data Mining Software: An Update. SIGKDD Explorations **11**(1) (2009)
8. Meffert, K., Rotstan, N., Knowles, C., Sangiorgi, U.: JGAP–Java Genetic Algorithms and Genetic Programming Package. URL: http://jgap. sf. net