Journal of Network and Computer Applications ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Network traffic application identification based on message size analysis

Amjad Hajjar [a], Jawad Khalife [a], Jesús Díaz-Verdejo [b]

[a] Faculty of Engineering IT Department, Lebanese University Beirut, Hadath, Lebanon
[b] Department of Signal Theory, Telematics and Communications - CITIC University of Granada, Spain

ARTICLE INFO

Article history:
Received 25 February 2014
Received in revised form
11 September 2015
Accepted 6 October 2015

Keywords:
Blind traffic identification
Message-size analysis
Classification
Clustering

ABSTRACT

Identifying network applications is centric to many network management and security tasks. A large number of approaches exist in the literature, most of which are based on statistical and machine learning techniques. For protecting the user privacy, the majority of the existing methods rely on discriminative traffic attributes at the network and transport layers, such as interaction schemes, packet sizes and inter-arrival times. In this work, we propose a novel blind, quintuple centric approach by exploring traffic attributes at the application level without inspecting the payloads. The identification model is based on the analysis of the first application-layer messages in a flow (quintuple), based on their sizes, directions and positions in the flow. The underlying idea is that the first messages of a flow usually carry some application level signaling and data transfer units (command, request, response, etc.) that can be discriminative through their patterns of size and direction. A Gaussian mixture model is proposed to characterize the applications, based on a study of the common characteristics of application-level protocols. The blind classifier is based on Markov models with low complexity and reasonable computational requirements, where the training procedure consists of profiling the target applications separately. Promising results were obtained for some popular protocols including many peer-to-peer applications.

1. Introduction

The ability to identify network applications (Khalife et al., 2014) is centric to many network management and security tasks, including quality of service assignment, traffic engineering, content-dependent pricing, resource allocation, and traffic shaping. With the proliferation of applications, many of them using different kinds of obfuscation, traditional port based classification has long become obsolete. Payload based classifiers face the challenges of obfuscation (Zink and Waldvogel, 2012), encryption and tunneling (Mujtaba and Parish, 2009), as well as user privacy rules. Blind classifiers do not inspect the payload and have the potential ability to deal with these obstacles, at the expense of an acceptable sacrifice in accuracy. Many of them rely on the analysis of patterns of traffic observed at the transport layer. The experience has shown that they are well suited to detect non-standard applications (e.g. Peer-to-Peer (P2P), Bittorrent, Zink and Waldvogel, 2012), which are intrinsically hard to detect due to their decentralization and dynamicity and especially their use of obfuscation and private, non-standard techniques.

Numerous methods have been proposed for traffic classification in the last decade. These methods have different characteristics at many levels, including the analyzed input, the applied techniques and the classified target objects (Khalife et al., 2014). Deciding upon which classification features to use is a strategic choice for any traffic classifier. Ideally, the selected traffic features should be discriminative, immune to network dynamics and obfuscation techniques, while still protecting the user privacy. Recent surveys (Khalife et al., 2014) show that most traffic identification methods use non-payload traffic features, usually extracted at the network and transport layers of the OSI model (Moore et al., 2015).

Thus, the classification method should better use input features that are resilient to the diversity in the underlying network technology, as well as jitter, congestion and other random phenomena. In this sense, individual packet sizes depend on the network technology's MTU, and, on the other hand, packet inter-arrival times are sensitive to jitter. Another aspect of input resilience concerns traffic obfuscation. For one, the issue of port number obfuscation is well known and admitted by the research community; so that classifiers based on port numbers are considered obsolete. Concerning packet sizes, it was reported in Lacovazzi and Baiocchi (2010) that some applications use padding to tamper the packets and evade packet-size based classifiers, and in Wu et al. (2012) that packet sizes and many other traffic features exhibit similar distributions through different application protocols (e.g. packet size distributions of BitTorrent and HTTP). In Yang et al. (2012), where a Bittorent traffic identifier was presented, the authors reported that the first three messages of BitTorrent

http://dx.doi.org/10.1016/j.jnca.2015.10.003
1084-8045/

handshaking have distinctive size properties, but unfortunately they are not transmitted in single packets; they are rather divided into several packets for the purpose of obfuscating packet-size based classifiers. As a conclusion, the authors recommended the use of messages instead of packets to detect the size features.

Application-level messages are not totally immune to the above-mentioned variations and obfuscations. However, among the commonly used features, they present the highest resilience, together with the highly desirable property of being derived directly from the communication of application entities, and as such, they permit us to look straight to the target, the applications they are supposed to identify.

Our proposal is based on the use of messages as the basic feature. The key is to extract the features of these messages while limiting the sniffing to the transport layer without inspecting the payloads. This is feasible to high extent from the headers of the transport layer.

In this work, we present a blind, quintuple centric traffic classifier analysis based on the features of application-level messages, as observed at the transport layer. The aim is to classify individual flows (quintuples) through the analysis of the "message-sequence patterns". Although the keyword message designates data units exchanged at the application layer, we show that the sizes of these messages can be extracted from layer 4 data headers, without actually inspecting the payloads of the messages. We then show by experience that by applying a supervised Bayesian analysis to the sequence pattern (sizes, directions and positions of the exchanged messages) we can identify the involved application with good accuracy.

The remainder of this paper is organized as follows. Section 2 presents the related work in traffic classification. Section 3 presents the motivation of the proposed method as well as the method itself, including the proposed model and parameterization. The procedures used for training the system, which includes a normalization of the sizes of the messages and a clustering algorithm, are described in Section 4. Section 5 presents the testbed and the experimental results. Section 6 analyzes the computational complexity of the system. Finally, Section 7 presents the conclusions of this work.

2. Related work

There is a relevant research activity in network traffic classification, employing many different approaches. Among them, payload based techniques, namely Deep Packet Inspection (DPI, http://www.ipoque.com), have the highest classification accuracy, thanks to their ability to inspect the packets' payloads and match discriminative application signatures. In Lu and Ling (2014) a hybrid method combining port numbers and packet inspection is suggested. However, packet inspection methods breach the users' privacy and fail to process encrypted payloads. Additionally, the need to analyze the whole payloads of every packet in the network represents a big challenge from the computational point of view. Though less accurate, the so-called blind methods are preferred in most environments because they guarantee the users' privacy, have the potential to classify encrypted communications and usually require less computational power. Various blind traffic classification techniques are analyzed and experimented in the literature, mainly falling in two categories: host based techniques that classify host activities by analyzing interaction schemes (Karagiannis et al., 2005); and quintuple-centric techniques that classify flows based on key features observed at layer 4 (i.e. Zander et al., 2005; Tabatabaei et al., 2012; Gu and Zhuang, 2010; Zhenxiang et al., 2011; Erman et al., 2007; Auld et al., 2007; Crotti et al., 2007; Yildirim and Radcliffe, 2010; Wang and Parish, 2010; Li et al., 2008; Dainotti et al., 2008; Huang et al., 2009; JinSong et al., 2007). These features typically include the flow size and duration, the packet sizes, and the packet inter-arrival times (Crotti et al., 2007; Wang and Parish, 2010). Our work falls in the latter category, with the distinction of using application-level messages instead of packets, while ignoring inter-arrival times.

Different Internet applications present different distributions in their packet sizes (Wu et al., 2012), and many classifiers use this property. Some use simple statistical techniques (Yildirim and Radcliffe, 2010; Wang and Parish, 2010) such as Probability Density Functions (PDF) while others use advanced ones such as application profiling (Wang et al., 2009) or Machine Learning (ML) (Zander et al., 2005; Tabatabaei et al., 2012; Gu and Zhuang, 2010; Zhenxiang et al., 2011; Erman et al., 2007; Auld et al., 2007; Li et al., 2008; Dainotti et al., 2008; Huang et al., 2009; Erman et al., 2007; Moore and Zuev, 2005). Some authors (Zhenxiang et al., 2011; Erman et al., 2007) suggested Bayesian supervised classifiers, especially naïve ones, as they are particularly characterized by their low complexity, fast training and computational efficiency.

Various relevant Machine-Learning (ML) approaches can also be found in the literature. *K*-Means (Erman et al., 2007) and AutoClass in Zander et al. (2005) were reported to identify some P2P applications with up to 80% of accuracy. Using ANNs in Gu and Zhuang (2010) and SVMs in Tabatabaei et al. (2012) yielded up to 85% accuracy for detecting a set of P2P applications. KNN algorithm provides 90% of reported accuracy (Huang et al., 2009) for some known applications including BitTorrent. However, the long training time and high complexity associated with most supervised learning algorithms (e.g. ANN and SVM) and the high storage and computational resources associated with others (i.e. KNN) imply a low scalability and a lack of generalization capabilities regarding the monitored network and the temporary evolution of the traffic.

On the other hand, despite the high reported accuracy, only a very limited set of protocols has been checked in these contributions.

Bayesian techniques, as in Zhenxiang et al. (2011), Auld et al. (2007), and Moore and Zuev (2005), have particular simplicity and low computational resource requirements (Khalife et al., 2014). Given an element, characterized by the observation of its features, these techniques estimate the probability that a class generates such an observation and label the element with the class that provides the highest probability. As such, these methods train very quickly, have low complexity and require reasonable computational resources. However, the main characteristic of naïve Bayes classifiers, which is the reason behind their low complexity, is the assumption that the different features are independent and have standard Gaussian distributions under the normality assumption. To overcome some of its limitations, the naïve Bayes model has been subject to many enhancements when applied to traffic classification, for example through incorporating neural networks (Auld et al., 2007) and payload inspection (Zhenxiang et al., 2011).

On another approach, the work in Wang et al. (2009) suggested identifying applications through the detection of the Longest Common Subsequence in their packet sizes. Reportedly, the idea was successful on a specific set of four P2P applications (Maze, Thunder, PPLive and Feindian). More general results are not available, and some preliminary experiments carried out at our lab did not show such packet-size signatures in a significant number of applications, even with a reduction of the alphabet by rounding or clustering methods. Although many applications have some discriminative packet sizes, the majority of application methods rather generate variable sized packets that are better described with probability distributions than with discriminative key values.

In this work, our proposed system analyzes the sizes of application layer messages using a Bayesian approach to label each flow from the probabilities provided by a set of Markov models, each

Please cite this article as: Hajjar A, et al. Network traffic application identification based on message size analysis. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.10.003

one associated to a given protocol. To the best of our knowledge, we are the first to combine the discriminative power of messages at the application layer with the simplicity and performance of Markov-based classifiers and the use of Multi-Peak Gaussian distributions to characterize message sizes. Our classifier analyzes messages based on their size, their direction and their specific positions in the flow.

Very few approaches analyzed traffic properties at the application layer without inspecting the payloads. Probably the most related previous works to this one are Waizumi et al. (2011) and Jaber et al. (2009). In Jaber et al. (2009), packets sizes are analyzed instead of message sizes. In the training phase, the authors used *K*-Means to classify the distributions of the packet sizes and provide one global cluster model. Then for each target application, and for each packet position in the flow, a probability was assigned to each cluster. The classification process examines the packet sizes as they come and computes accordingly the probability of the flow (packet-size vector) to be generated by an application. The flow is then assigned to the application providing the highest probability in a naïve Bayesian way. In Jaber et al. (2011), the same authors propose an enhancement via the use of the packet inter-arrival times as a feature. Although a previous feature analysis by Erman et al. (2007), using backward greedy search, reported that "features that have a time component such as duration, inter-arrival time, and throughput were found not to be useful by the feature selection algorithm", the proposal in Jaber et al. (2011) was to subtract the observed "monitor-to-server" round-trip-time from the inter-packet times, and as a result, this feature is argued to become meaningful when observed at approximate positions in the flow. Although the reported results suggest a potential usefulness of the inter-packet time feature, the suggested technique permits us to cope only with the randomness of host locations. The authors admittedly neglected the effect of variations in network conditions and jitter, and it is not clear to which extent this assumption is valid under various network conditions and congestion. Also, the evaluation, both in Waizumi et al. (2011) and Jaber et al. (2009), was applied to only five standard applications (HTTP, SMTP, HTTPS, SSH and IMAP). As a main difference, packet time information is not used in our model. Other key differences of our work with Jaber et al. (2009) are the use of the message level sizes instead of packet sizes and the use of a per-application Gaussian mixture model for clustering the message sizes.

The work presented in Waizumi et al. (2011) has a common point with this one in the fact that it analyzes the message sizes for the classification. Despite a different mathematical model, the key difference is that Waizumi et al. (2011) quantify the message size in terms of number of packets. Our experience showed that this coarse-grained quantification overlooks important and meaningful characteristics of the message. For instance, following this quantification, all messages involving one sole packet are considered similar. This does not permit us to identify key message sizes, especially those involved in some application level handshaking and methods, such as SMTP "HELO", HTTP "GET", and FTP "USER". These methods often generate one-packet messages; still their expected sizes are different and provide potential information for the classifier. For these reasons, our quantification of the message size is fine grained, based on the total number of bytes in the message and not only the number of packets.

Finally, we should note that a trend is emerging to monitor message sizes as the main source of information on encrypted traffic flows. This is noticeable in Pironti (2014) and Iacovazzi and Baiocchi (2014), and it reveals again the importance of the message size feature for flow classification.

At the time this work was conducted, Rizzi et al. (2013) proposed a neuro-fuzzy classifier with low structural complexity, yet comparable results to SVM. There are two major differences to note with this work. First, Rizzi et al. (2013) use packet sizes and inter-arrival times as features, while our method uses message sizes and ignores the timing data. Second, in the context of network traffic, new applications appear frequently, and a method based on independently "profiling" the applications is highly desirable. Our method follows a profiling approach, where the training consists of profiling each application standalone.

The main contributions in this work are the following:

- *Message size-based vectors*: the use of message-size parameters instead of packet-size parameters. Most previous works on identifying applications from packet lengths used the (non-empty) packets to generate the parameter vectors, based on their sizes and direction. Although we use the same notion for the direction, we followed a different approach for the sizes and the definitions of the parameter vectors, which removes from the process-level messages the "noise" induced by layer-4 segmentations, retransmissions and acknowledgments.
- *Message size scaling*: definition of a normalized message size measure and a metric distance that is appropriate to their semantic meanings in network communications.
- *Gaussian mixture*: the use of a Multi-Peak Gaussian model (MPG) for estimating the probability density functions of the vectors' components.
- *Profiling*: the model is simple and extensible with minimal effort, as it consists of profiling each application standalone. The addition of a new application protocol requires rerunning the training procedure on a sample set to extract its parameters, without any need to review the other applications' model parameters and sampled data.
- *Testing*: the model was tested on a relevant number of applications mixing standard client-server and P2P protocols. Many previously reported works were tested on a small and specific set of application protocols. We tested our method on a dataset containing about 3 million flows and 18 application protocols, 10 using TCP and 8 using UDP.

3. Flow classification based on initial messages

As previously mentioned, the proposed blind classification method is built upon message size analysis. The classifier of choice for our experiments is based on Markov models with a training procedure that mixes supervised and unsupervised schemes to produce a model for each application (a model-per-class). The approach consists in considering the messages' sizes as the productions (observations) of a first order Markov model. Thus, each single state of the model is associated to a single message from an application. During training, by using randomly sampled data from each application separately we estimate the probability distribution functions (PDFs) of the message sizes at given positions in the flow, with the underlying idea that these messages are generated from (a priori unknown) "methods" defined in the design of the application protocol. Thus, the probabilities for each of the observations are obtained from a multi-Gaussian probability distribution function which is also dependent on the application. Finally, when classifying traffic, the class associated to each flow will be that of the model providing the highest probability. This probability is evaluated by a naïve Bayes classifier that computes the Bayesian probability that a messages size sequence was produced by the model associated to a given application and assigns the flow to the model (application) that provides the highest probability score.

Prior to explaining the details of the system, it is worth to mention that the approach is based on some observations and findings regarding the characteristics of application protocols. These findings

Please cite this article as: Hajjar A, et al. Network traffic application identification based on message size analysis. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.10.003

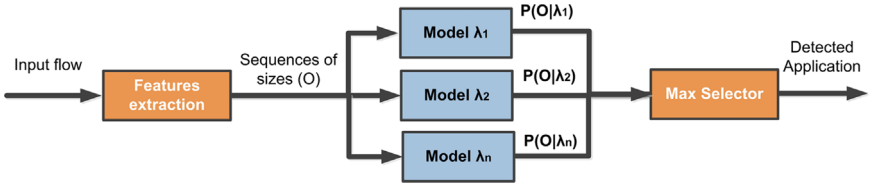

Fig. 1. Diagram of the proposed system.

motivate not only the proposal but also some design decisions made by the authors and are briefly explained next.

3.1. Common application protocols characteristics

Up to five properties or characteristics of applications protocols are of interest for the proposed classification system. They are the following:

- *Message size analysis potential for TCP*: For TCP-based applications, message size analysis is expected to outperform packet size analysis, because the former reflects what is really exchanged between application processes, while the latter is affected by transport layer segmentations and retransmissions, and these can be considered as a source of noise to the classifier. On the other hand, this assumption is not relevant in UDP-based applications, because no segmentation or retransmission occurs at layer-4, and the flowing packets are directly exchanged between end processes. In this case, the messages are usually the packets themselves.
- *Visibility of message sizes*: It is possible to extract message size information from layer-4 headers, without inspecting the payloads. A method to extract this information will be shown in a subsequent section.
- *Use of methods*: Network application protocols are designed around limited sets of "methods", data units corresponding to the semantics of the communication. Any exchanged message is a data unit belonging to one of these methods. Examples of these methods are HTTP "GET" and "POST"; SMTP "HELO", "200 OK" and "USER"; FTP "USER" and "PASS". As will be shown later, the proposed classifier does not require prior knowledge of these methods, but exploits this characteristic of network protocol designs and applies an unsupervised training process to estimate the main characteristics of the underlying methods. As an example, a method that is common to many network applications is data transfer, which takes the form of a sequence of full-size packets flowing in the same direction, while empty "ACK" packets flow in the reverse direction. From the point of view of a message-based classifier, this sequence consists of a single large message. On the other hand, many methods are used in handshaking procedures, and these usually generate relatively small messages. Our proposal assumes that both directions of the communications are monitored.
- *Distribution of the size of the messages*: Each application-level method generates messages, with a size that can be represented as a random variable with inherent statistical characteristics. Our study of many protocols has shown that some methods generate fixed size messages, while most others have a probability distribution centered over a mean. The proposed classification method assumes that these random variables have a normal distribution. Thus, the method attempts to estimate the essentials of the Gaussians through an unsupervised learning process, using clustering. As stated before, this clustering eliminates the need of any prior knowledge of the list of methods of any protocol.
- *Sequence of methods*: Application methods occur at typical positions in network flows. Although this assumption is not deterministic, it is statistically significant, especially at the beginning of the flows, in the first exchanged messages which are often dedicated to some handshaking procedure such as identification, authentication, and request/reply. Accordingly, a classifier can exploit this characteristic by examining the first messages in the flow. An interesting exception occurs in the FTP protocol, where the handshaking occurs in one flow called "the main connection", while the data transfer (upload or download) occurs on a "secondary connection", opened specifically for the data transfer, and identified by a different quintuple than the main one. No handshaking occurs on the secondary connection, and the file transfer takes place immediately. As a consequence, a quintuple centric classifier will observe some flows consisting of a single large message. As long as FTP is the only application that produces such a scenario, this is not problematic for the classifier. However, in the presence of other such applications, the classifier needs additional information to classify the flows consisting of a single large message. The needed information goes beyond the quintuple, mainly by examining the set of flows between the same two hosts. This leads to a "stateful" approach, which is a radical modification based on the tracking of host-to-host flow sets. The idea of analyzing host-to-host flow sets is open for further exploration, but in the current status, one-message flows are ignored, and the classifier was tested on flows that include at least two messages.

3.2. System architecture

The proposed system consists of three main components (Fig. 1):

- a parameterization module, which obtains the vector $\mathbf{O_i}$ of message sizes, s_j, for each flow, F_i:

$$\mathbf{O_i} = \langle s_1, s_2, \ldots, s_L \rangle \quad (1)$$

  with L being the selected analysis length specifying the number of messages to be considered in the parameter vector.
- a model, composed of a set of N Markov models (one per application to be detected), which obtains the probabilities for the sequences of observed vectors for each of the potential applications

$$\{P(\mathbf{O_i}|\lambda_n)/1 \le n \le N\} \quad (2)$$

  It is worth mentioning that the underlying first-order Markov model is simple and present the same topology for all the applications (providing one state for each message position), but the probability distributions for the message sizes at each state differ and are specific to each application and
- a decision module, which selects the application the flow belongs to as that of the model providing the maximum probability

$$class(F_i) = \arg\max_n \{P(\mathbf{O_i}|\lambda_n)/1 \le n \le N\} \quad (3)$$

Therefore, the system uses a Maximum a posteriori probability (MAP) approach in which all the classes are supposed to be equally probable a priori.

Please cite this article as: Hajjar A, et al. Network traffic application identification based on message size analysis. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.10.003

A key aspect in the proposal is the evaluation of the observation probabilities for each of the sizes. For this, as previously mentioned, multivariate Gaussians dependent upon the state and the model are used as probability distribution functions (PDFs). That is, we assume a discrete time Markov Chain (DTMC) with a state associated to each position in the flow, and a symbol (message-size) is generated from each state following a Gaussian Mixture PDF defined over a continuous space (this will be further illustrated in Fig. 3). Obviously, the training of the system requires the evaluation of each of the Gaussians for each of the models. In other words, profiling an application consists of estimating its Gaussian mixture at each state, using a set of training samples.

The details on the operation of each module are described in the next subsections while we devote a section for the details of the training procedure.

3.2.1. Features extraction

As previously explained, the first step to classify a flow is the extraction of the features vector which will be used as the input to the models. For this, each flow is modeled as a vector of relative numbers that parameterizes the sequence of messages. Specifically, each message is modeled by its byte length with a sign that is positive if the message is generated by the flow's initiator and negative in the other case.

To compute the message sizes, the TCP header information is taken into account in order to track any sequence of packets that constitute one message. In fact, it is known that large data transfers, such as the transfer of an image, a file, or an HTML document, are decomposed by TCP into many packets according to the negotiated Maximum Segment Size (MSS), which derives from the Maximum Transmission Unit (MTU) of the underlying network. These data transfers can be tracked by inspecting the layer-4 headers. At the same time, tracking the layer-4 headers permits us to:

- Remove retransmissions from the flow, which is desirable to improve the accuracy and focus on the application-level messaging instead of the "brute" packet flow.
- Remove "pure acknowledgment" packets, which are relevant to layer 4 but have no relevance to the application level messaging.

The number of messages to be considered in the parameter vector, L, is predefined globally for the system (or a maximum is fixed). Ideally, L should be as low as possible in order to classify the flow as soon as possible and to handle short flows. This factor should be selected during the training of the system.

Therefore, the vector generation method acts as follows. Given a flow, $F_i(a \rightarrow b)$, initiated by host a and directed to b, a vector, $\mathbf{O_i}$, of L signed integer values, corresponding to the message sizes and their directions with respect to the flow initiator, is built. TCP handshaking packets and pure ACK packets are removed from the flow and not considered in the vector. That is,

$$\mathbf{O_i} = \langle s_1, s_2, \ldots, s_L \rangle, s_m = \begin{cases} size(message_m) & \text{if } message_m(a \rightarrow b) \\ -size(message_m) & \text{if } message_m(b \rightarrow a) \end{cases} \quad (4)$$

being $message_m$ the m-th message in the flow. A distinction is made in the treatment between UDP and TCP flows. Obviously, in UDP flows, packets correspond to messages and there is no distinction between the two concepts. However, as depicted in Fig. 2 for TCP flows, as long as the payload data flows in one direction, the payload sizes are accumulated into the same message (the same vector component), until one of the following occurs:

- A packet carrying data is detected in the opposite direction.

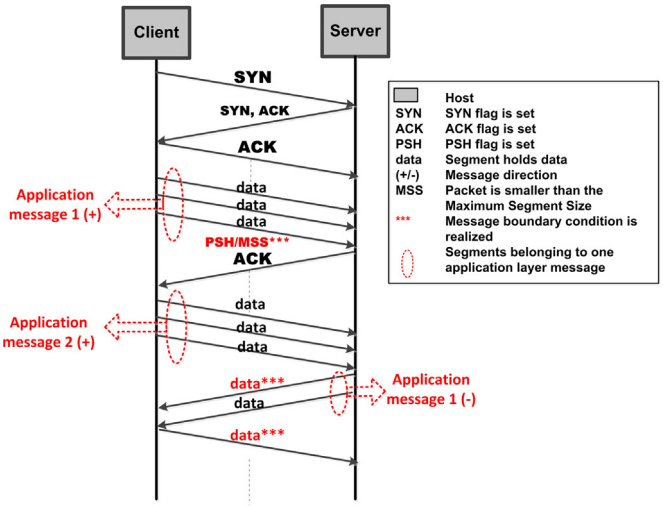

Fig. 2. Extracting application layer messages from a TCP session.

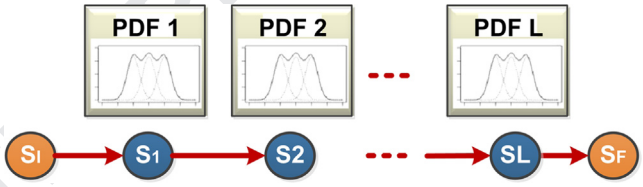

Fig. 3. Model topology for a single application.

- A PUSH flag is detected in TCP's header. This flag usually indicates that the sender process has finished its message. An exception is when the receiver's window is full, but this case is ignored in our approach.
- A packet that is smaller than the MSS is detected. This heuristic is based on the assumption that large data transfers use full MSS packets until the transfer is over or the window is full (the latter case is ignored as we stated earlier). Therefore, a small packet indicates the end of a message even if no PUSH is detected.

Regarding the last condition, it is important to note that for the experimental part in the present work we do not extract the MSS from the TCP handshaking packets, but we simply set it to 576 bytes, which is the smallest MTU specified in RFC-791. Another important remark is that in some application protocols, it is possible that two consecutive messages follow the same direction. Chatting protocols (i.e. MSN) are obvious examples. This rule permits us to detect such situations, and makes an improvement over some previous work (i.e. Waizumi et al., 2011) where all consecutive non-empty packets flowing in the same direction are considered as one message.

3.2.2. Sequence evaluation

The proposed method assimilates each application as a pure left-to-right Markov chain (Fig. 3), where each message in turn is generated from a state of the model: the message at index 1 corresponds to state 1, the message at index 2 corresponds to state 2, and so on.

A set Λ composed of N models, one per considered application, is estimated during the training of the system

$$\Lambda = \{\lambda_n / 1 \leq n \leq N\} \quad (5)$$

As part of the training of the models, a set of PDFs for the observations, G, is estimated. Each PDF is associated to a model and a

state

$$G = \{g(n,l)/1 \le n \le N, 1 \le l \le L\} \quad (6)$$

Therefore, for an input vector, $\mathbf{O_i}$, the system has to evaluate all the probabilities of generation of the sequence according to each of the considered models (applications), as per Eq. (2).

As a pure left-to-right model is assumed, only the observation probabilities have to be evaluated. Thus, the probability of vector $\mathbf{O_i}$ to be generated from application n can be computed as the product of the observation probabilities on all indexes

$$P(\mathbf{O_i}|\lambda_n) = \prod_{l=1}^{L} P(s_l|g(n,l)) \quad (7)$$

It is worth mentioning that the used modeling assumes that the sizes of the messages at each state (index) are independent from each other. That is, the probability of having a certain message size at position l is independent of the sizes observed at previous positions $l-1, l-2 \ldots 1$.

3.2.3. Classification

A flow, F_i, will be classified as belonging to the application whose associated model, λ_{app}, provides the maximum probability for generating the observations from the flow, $\mathbf{O_i}$, that is,

$$class(F_i) = \arg\max_n \{P(\lambda_n|\mathbf{O_i})/1 \le n \le N\} \quad (8)$$

This probability of the model given the sequence of observations is not directly provided by the set of Markov models, but the opposite, $P(\mathbf{O_i}|\lambda_n)$, that is, the probability of the sequence given a model. By applying Bayes' rule,

$$P(\lambda_n|\mathbf{O_i}) = \frac{P(\lambda_n)P(\mathbf{O_i}|\lambda_n)}{P(\mathbf{O_i})} \quad (9)$$

Assuming that all the models are equally probable a priori, the decision rule can be rewritten as

$$class(F_i) = \arg\max_n \{P(\mathbf{O_i}|\lambda_n)/1 \le n \le N\} \quad (10)$$

Although the prior class probability, $P(\lambda_n)$, might be predefined with prior statistics on the network traffic subject to classification, and hence can be set as a parameter to the method, the proposed approach and the results that we will show assume no such prior knowledge. There are three reasons behind this. The first reason is that prior probabilities may not be stationary, but change over time as users come and go or engage in different kinds of network activities. The second reason is that these probabilities are highly host dependent and also environment specific. The third reason comes from the way in which the performance of the classifier will be assessed.

As will be detailed in Section 5, the evaluation of the system is driven by the "worst case" detection rate. That is, we consider as a main criterion of performance the worst case detection rate, which is defined as the percentage of correctly identified flows from the application that yields the minimum such percentage. Following this logic, the classifier should not exploit prior statistics by favoring the predominant applications. This is especially true when a classifier is evaluated on datasets that have a largely unbalanced number of elements from each class, as is usual in real traffic, because favoring the predominant application misleads to optimistic results. For these reasons the prior probabilities are ignored and assumed equal for all applications.

In order to evaluate the confidence on the classification of each individual flow, a probability for the flow belonging to each application is set as

$$P(F_i \in class(c)) = \frac{P(\mathbf{O_i}|\lambda_c)}{\sum_{n=1}^{N} P(\mathbf{O_i}|\lambda_n)} \quad (11)$$

An ideal system will provide a probability value of 1 for the correct class and 0 for the incorrect classes.

Therefore, the higher the value, the higher the confidence in the classification provided and a better operation of the system.

Furthermore, this measure could be used to train the system instead of the most common case in which the training focuses just in maximizing the production probabilities for the correct classes (MMI vs. ME training).

4. Training of the system

The method used for training the models plays a fundamental role in the proposed system. Up to now, its core is a standard Markov model based recognizer applied to features vectors composed of sizes of the messages. As the topology and transitions of the Markov model are fixed by design, the parameters to be obtained by training are the PDFs to be used, G. Therefore, it is in the choice and estimation of these PDFs where the major novelty of the proposal resides. For this, as previously mentioned, we propose to use model and state dependent PDFs based on multi-peak Gaussians. The fundamentals for using multi-peak Gaussians instead of simple Gaussians are related to the proposed modeling. As previously mentioned, each model is supposed to represent a single application (protocol) behavior. But most of the protocols can be split into many methods with different behaviors regarding message sizes. Therefore, the model would be a mix of all the observed methods from a single application. To account for this, different Gaussians are associated to each state of the model, as will be detailed in Section 4.2. Anyway, it is worth to mention that no differentiation between those methods will be done explicitly nor for training the system nor for evaluating a sequence.

Estimating one single Gaussian for every model/state would be done by using just all the samples associated to each model/state and fitting the parameters. This assignment would be almost trivial as the training samples should be labeled and each observation is directly associated to a state in the Markov model according to its position in the sequence. But, as multiple Gaussians are to be estimated for every state, a method to assign a given observation to one or many of them is required. For this, the proposed solution uses a quantization and normalization of message sizes based on a proposed metric providing a weighting of the belonging of an observation to different distributions.

4.1. Metric and normalization of message sizes

The protocol methods are encoded inside the payloads, and we are seeking a blind classification method that does not investigate these payloads, but exploits potential information from the sizes of the messages generated by the methods. In general, the size of a message involved in a method is a random variable.

In order to estimate the probability of a message belonging to a method, based on its size, we need a way to assess whether two messages have similar sizes and to which extent. This is the role of the metric distance that we should define.

A rather naïve way to define the distance between two messages is to compute the "absolute difference" between their sizes. The absolute difference is the Euclidean distance between the messages and, in topological notation, it possesses the "translation invariant" property:

$$D_E(x,y) = D_E(x+a, y+a) \quad (12)$$

This property is not appropriate for our application, because it does not fit well with the way the network applications messages are generated. As an example, consider a data transfer message. The transferred data might be a file of 20 KB or 30 KB. The absolute

Please cite this article as: Hajjar A, et al. Network traffic application identification based on message size analysis. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.10.003

byte-difference is 10 KB (big), but still, the two messages have the same semantic meaning, and likely belong to the same method in any network application (i.e. transfer of an icon or a small image). On the other hand, a "HELO" message in SMTP has a typical size of 30 bytes while a HTTP GET request has a typical size of 300 bytes. The difference here is less than 300 bytes, but it is more meaningful than the 10 KB difference in the above-mentioned image transfer. The defined metric distance must point out that the same byte-difference is more significant between two small messages than between two big ones. The bigger the message, the less significant is the byte-difference.

Therefore, any metric distance defined on message sizes must consider relative differences rather than absolute differences. In other words, it should not be translation invariant.

In the proposed approach, the message sizes are "normalized" into the]-1, 1[space using a transformation that permits us to scale the difference between two messages, in terms of bytes, relatively to the sizes of these messages. The transformation, T, is defined as:

$$T(s): \quad \mathcal{Z}^* \rightarrow]-1,1[$$
$$s \rightarrow s' = \frac{s}{(B+|s|)} \tag{13}$$

where B is a positive constant that corresponds to the middle of the space, that is, a threshold for considering a message as big or small. A typical value, as described in the literature, is around 500 bytes. In our experience, any value from 300 to 600 bytes would not dramatically change the results. Figure 4 graphically shows the proposed rescaling function.

After rescaling the sizes of the messages, a metric distance, $D()$, between two messages with sizes s_1 and s_2 is derived from their signs and the absolute difference between their normalized sizes, as:

$$D(s_1,s_2) = \begin{cases} 1 & \text{if } s_1 \cdot s_2 < 0 \\ \left| \frac{s_1}{(B+|s_1|)} - \frac{s_2}{(B+|s_2|)} \right| & \text{otherwise} \end{cases} \tag{14}$$

Notice that the distance between a negative value and a positive one is set to the maximum of 1. This property is motivated by the fact that, in network communication, two messages flowing in opposite directions are semantically different even if both have similar sizes.

The defined distance is a metric distance that verifies the following properties:

$$\begin{aligned} &\textit{Range} \quad \forall x,y \in \mathcal{Z}^*, \quad D(x,y) \in [0,1] \\ &\textit{Identity} \quad \forall x,y \in \mathcal{Z}^*, \quad D(x,y) = 0 \Leftrightarrow x = y \\ &\textit{Symmetric} \quad \forall x,y \in \mathcal{Z}^*, \quad D(x,y) = D(y,x) \\ &\textit{Triangular inequality} \quad \forall x,y,z \in \mathcal{Z}^*\mathcal{Z}^*, \quad D(x,z) \le D(x,y) + D(y,z) \\ &\textit{Translation} \quad D \text{ is not translation invariant} \end{aligned} \tag{15}$$

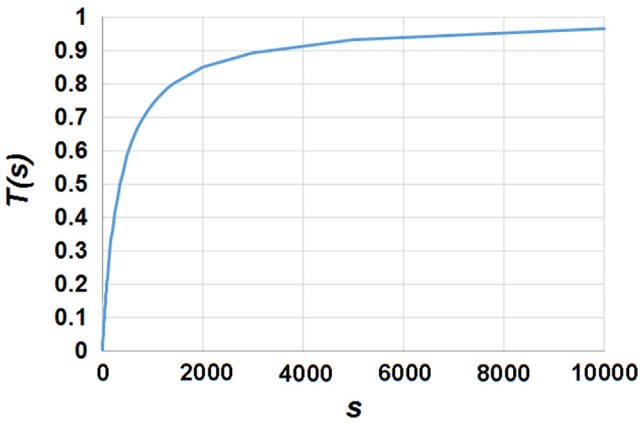

Fig. 4. Function used to normalize the message sizes ($B = 350$).

This metric permits us to assess the degree of similarity between any pair of messages from the point of view of the application-level protocols that generate them.

4.2. Gaussian mixture model

Once a distance with the desired properties is set, a clustering of the observed message sizes is made in order to account for different methods of a protocol using this distance. In this section, we propose and justify a model for the Probability Distribution Functions (PDFs) of the observed message sizes. There are three observations that drove the derivation of the model; in what follows we will recall these observations and discuss the derivation of the model:

- *Observation* 1 *(re-statement)*: Application layer messages are generated from methods defined in the design of application-level protocols. Each observed message is an occurrence of one of these methods.
- *Observation* 2: Each method generates similar message sizes with some reasonable amount of randomness. This similarity is best expressed in the sense of the distance defined previously, which is more tolerant toward large messages than small ones. Accordingly, we assume that the normalized message sizes generated from a method follow a normal distribution, characterized by a mean μ and a standard deviation σ. Occasionally, a few methods, usually encountered in signaling schemes (i.e. bit-torrent handshaking), generate fixed size messages ($\sigma = 0$).
- *Observation* 3: Given an application and a state, the observed message may be generated from any of the application's methods, with some probability for each. For example, the HTTP-GET-request is generally more frequent than the HTTP-POST-request; an HTTP-GET-response is not likely to occur as the first message of an HTTP flow, etc.

Combined, observations 2 and 3 imply that the message-size distribution is close to a multi-peak Gaussian mixture. For a given application/state, each peak or cluster represents a method, with an underlying mean, μ, standard deviation, σ, and a weight, w, which is the probability of occurrence of the method at the given state.

4.3. Profiling the applications

A supervised training approach for the proposed modeling would be based on the enumeration of the methods for each application protocol. With enough samples, each method would be analyzed individually and its parameters (μ, σ, w) estimated. However, this task is likely impractical for several reasons with the most relevant one being the need for a huge volume of traffic labeled accordingly. Therefore, an unsupervised approach based on clustering is designed to capture the essentials of the most frequently occurring methods.

The aim of the clustering is, given an application, to provide a means to estimate the number of methods and their statistical characteristics in an unsupervised way. This proposal is similar to that in Jaber et al. (2012), in which the authors used K-Means to find a global set of clusters from the packet sizes coming from all the applications, and then to assign a probability to each (application, state, cluster) triplet.

In our proposed model, a cluster is presumably associated to each individual method of each application protocol, and the training uses Expectation Maximization (EM) of a Gaussian mixture. The training process, which extracts the clusters and their characteristics, is applied to each class separately (a model-per-class). In a sense, this is a "profiling" approach: the training

process applies the profiling routine to each protocol, and extracts its Gaussian mixture PDF at each state. A few points are important to note here.

First, this approach is easily extensible to new classes. To add a new target class to the classifier, it is sufficient to apply the profiling routine to samples of the new class, without any involvement of the already profiled classes and their sampled data. This advantage is particularly important in the context of network traffic classification, as the number of network applications is usually high, and new applications appear regularly on the Internet. This contrasts to other approaches that use global profiles or models, as SVM or neuro-fuzzy learning, such as Rizzi et al. (2013), which require rerunning the training process on the entire sample set when a new class is added to the classifier. This argument justifies the use of a maximum likelihood approach, such as EM for estimating the multi-Gaussian PDFs, though it is arguable whether SVM's "optimal separation" would yield a significant improvement on the classifier's accuracy. But this potential increase in the accuracy could be obtained at the cost of increasing the specificity of the obtained models in relation to the training dataset, which is not a desirable property.

Finally, the training approach is both supervised and unsupervised, but it should not to be confused with mixed supervised/unsupervised training. It is supervised in the sense that the flows are a priori labeled. On the other hand, it is unsupervised in the sense that, in the profiling phase, there is no prior mapping of individual messages to the application's methods.

4.4. Clustering and estimation of gaussian mixtures

As previously stated, the aim of clustering is to capture the essential methods and their characteristics for each application/state instance. In our experiments, the method used to estimate the clusters is a combination of *K*-Means and Dempster's EM. However, any method for estimating Gaussian mixtures would be appropriate, such as a greedy learning method (Verbeek et al., 2003).

Given an application, n, and a position in the flow, l, the first stage of the training uses iterative *K*-Means to cluster the message-size samples. As usual, the clustering is applied till a threshold for the distortion or a previously fixed maximum number of clusters, K, is reached. The result after the clustering is a certain number of $K_{n,l} < K$ meaningful clusters. The $K_{n,l}$ meaningful clusters from *K*-Means are used as a starting point for Expectation Maximization (EM) algorithm. Each cluster is considered as a cloud around a peak in the multi-peak normal distribution, and Dempster's EM algorithm (Dempster et al., 1977) is used to estimate for each cluster, $C_{i,n,l}$, the centroid, $\mu_{C_{i,n,l}}$, the standard deviation, $\sigma_{C_{i,n,l}}$, and the weight, $w_{C_{i,n,l}}$. The aim is to maximize the overall likelihood of all the observations in the training set.

A certain number of clusters may have all their samples lying exactly on the centroid, with a null standard deviation. This happens when some protocol methods have a deterministic message size, but it also might be the result of random sampling, with "missing data". To deal with this phenomenon without losing generality, and in order to ensure a Bayesian analysis, a minimum standard deviation, σ_{min}, is predefined. Thus, if for a given cluster, $C_{i,n,l}$, its standard deviation, $\sigma_{C_{i,n,l}}$, is lower than σ_{min}, σ_{min} is used as its standard deviation.

Given cluster $C_{i,n,l}$ and an observation s_l, the probability density that the observation belongs to this cluster, $P_{C_{i,n,l}}(s_l)$, can be computed as

$$P_{C_{i,n,l}}(s_l) = P(s_l \in C_{i,n,l}) = N_c(D(s_1, \mu_{C_{i,n,l}})) = \frac{1}{\sigma_{C_{i,n,l}}\sqrt{2\pi}} \cdot e^{-0.5\left[D\left(s_1,\mu_{C_{i,n,l}}\right)/\sigma_{C_{i,n,l}}\right]^2} \quad (16)$$

where $N_c()$ is the normal distribution function.

From this, the probability of the observation being generated from the application n at position l can be evaluated as

$$P(s_l \mid g(n,l)) = \sum_{i=1}^{K_{n,l}} w_{C_{i,n,l}} \cdot P_{C_{i,n,l}}(s_l) \quad (17)$$

It is worth mentioning that the value space is split into two disjoint sub-spaces before applying the clustering: one for the positive values and the other for the negative ones. This "gap at zero" means that the sign of any observation is its top-level characteristic. Thus, two multi-peak Gaussians will be estimated: one for positive observations and the other for the negative observations.

5. Experimental results and assessment

5.1. Test bed

In order to assess the proposed method and provide experimental results, an appropriate testbed has to be set. For this purpose, the first step should be to prepare traffic captures in which the flows are labeled according to the application protocol. This set will be used as the ground truth to train and test the system. The methodology used to capture and label the traffic dataset is described next.

5.1.1. Traces and collection methodology

To obtain a significant and real dataset with enough mixture of application protocols, we captured real traffic at the access link of a medium-sized institution (Shannon et al., 2014). The collected traces include traffic from LAN and Wireless LAN users. As depicted in Fig. 5, end clients consist on fixed PCs and mobile devices. The network infrastructure is policed by traditional access control devices (firewall, web filter, etc.) allowing the access for P2P applications.

For training and validation purposes, full packet traces in which complete flows in both directions were captured. To assess the longevity of the proposed classifier, traces were captured over separated periods of time. Specifically, we collected two separate traces of 3 and 2 days over a span of six months, totaling around 225 GB of real traffic. We also tested our method on a publically available dataset, namely the m57-patents (http://digitalcorpora.org/corpora/scenarios/m57-patents-scenario).

5.1.2. The ground truth and data preparation

To build the set of correctly labeled flows that will be used as the reference (i.e. the "ground truth") for each dataset, we used a customized nDPI tool (http://www.ipoque.com). This tool is the

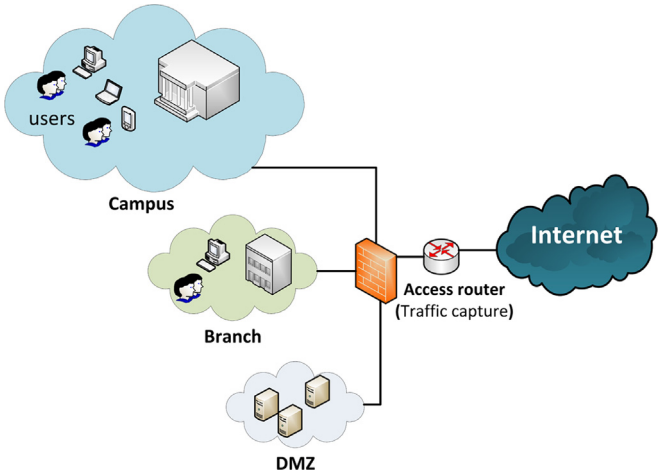

Fig. 5. Diagram of the network used for the caption.

open source evolution of OpenDPI after it became commercial. Its core is a software library designed to classify internet traffic. In its current version, more than 150 different protocols can be identified, including some P2P and encrypted application protocols.

In addition to pattern matching, nDPI improves its classification by using additional techniques such as behavioral patterns, statistical indicators and entropy.

Each of the flows in the dataset is assigned a vector including its DPI classification and some useful information from the flow: the start time, the IP addresses, the layer-4 ports and the layer-4 protocol (TCP or UDP), since our method treats differently the TCP and the UDP flows. Additionally, the sequence of sizes for the $L=6$ first messages in each flow is also obtained, according to the features extraction procedure (Section 3.2.1).

One of the requirements of the proposed classifier is to have at least two messages to classify a flow. Therefore, the flows that have only one message are ignored. As we stated earlier, this does occur in some protocols, such as FTP, that use signaling on the main TCP connection and open other TCP connections for large data transfers.

A set of independent experiments are to be made with increasing values for L. For each of these experiments, a random set of up to 4000 samples per application protocol is chosen from the dataset as the training set. But, as not all the applications present in the dataset have enough flows with the required minimum number of messages, especially when L increases, a strategy has to be applied to the number of flows in the training set.

Thus, in order to avoid overfitting and unbalancing, the number of samples from an application in the dataset is, at most, half the number of available samples with a maximum of 4000 (if more than 8000 flows are available). On the other hand, and in order to ensure a proper training, a minimum is also set to include the application in the experiments. This minimum is set to 500 samples for TCP applications and 10 for UDP applications. For each value of L, the applications that do not have enough flows to satisfy this requirement are ignored in the experiment, which leads for the selection of up to 18 different applications from the used dataset. Table 1 lists for each of the usable applications the number of available flows and the number of samples used for training, for the case $L=3$.

5.2. Experimental results

In this section, we show experimental results obtained during the training and the classification phases.

5.2.1. Training phase results

As previously mentioned, clusters and PDFs associated with each application and state of the model are the main outcomes of the training process.

Table 1
Number of flows and training samples (for $L=3$) for each application in the experimental dataset.

TCPapp	# flows	# samples	UDPapp	# flows	# samples
Bittorrent	15,487	4000	iMesh	422	211
iMesh	1090	531	Pando	448	224
MSN	17,723	4000	MSN	813	400
SSL	223,132	4000	NTP	879	400
HTTP	1,581,831	4000	RTP	220	110
Gnutella	2022	1047	Gnutella	158	79
Oscar	1666	833	Stun	1973	400
POP3	5197	2624	NetBios	543	272
FTP	1352	689			
SMTP	2041	1061			

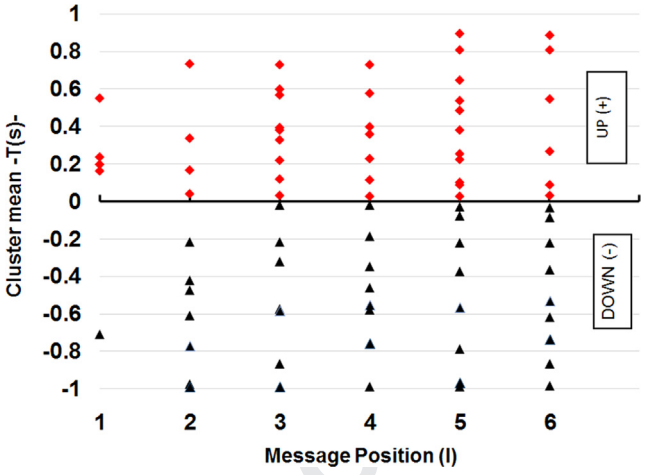

Fig. 6. Mean message size (normalized) for the clusters obtained for Bittorrent ($L=6$).

Figure 6 shows some preliminary results for the clusters obtained for Bittorrent considering the first six message positions ($L=6$) and a maximum of 12 clusters ($K=12$) per state.

As shown in Fig. 6, clusters belonging to the same applications may not be sparse when independently considering different message positions and directions (e.g. clusters 5th and 6th for the up direction for $L=3$). This reveals the number of maximum considered clusters, K, and their associated standard deviations, σ, as relevant factors for a proper clustering of the data.

Thus, some preliminary experiments were made in order to evaluate the impact of K and σ_{min}. First, many values of K were tested. Our experience showed that the method is not highly sensitive to the choice of K. The typical value of K, which is used in the experiments, detailed in this paper is $K=20$.

On the other hand, the exact value of σ_{min} is not critical, as our experiment showed that the results are resilient to the choice of *minimal deviation*, σ_{min}, in a wide range of values, from 10^{-12} to 10^{-6} (the exact significance of these values derives from the defined metric distance).

As explained in Section 4.2, once the clusters are set, the parameters for the associated Gaussians are obtained. For illustration purposes Fig. 7 shows the set of PDFs associated with Bittorrent and HTTP applications for the second message position ($l=2$).

As shown in Fig. 7, the peaks of the Gaussians (centroids) together with their deviations partially overlap at some cases. This is an indicator of the goodness of the obtained clusters and the appropriateness of a multimodal approach by using multiple Gaussians. On the other hand, an additional relevant observation is related to the final target of the system, that is, to the classification capabilities. Thus, the distributions obtained for Bittorrent and HTTP are clearly different, which points to the fact that message sizes at the second position in the flows can discriminate between both applications. However, as mentioned previously, the discrimination among different applications is made based on a Markov model which accounts for the different message sizes at different initial positions in the flow, which is expected to contain more discriminative information than the individual sizes at a given position. To further illustrate this idea, Fig. 8 shows application message sizes for sample Bittorrent, SSL and Gnutella flows at different message positions.

As shown in Fig. 8, message sizes for those applications are similar at particular positions for some of them (e.g. for $l=1$ for all applications, $l=5$ for SSL and Gnutella) while being clearly different from others. Thus, considering message sizes at different positions and in both directions can discriminate between both

Please cite this article as: Hajjar A, et al. Network traffic application identification based on message size analysis. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.10.003

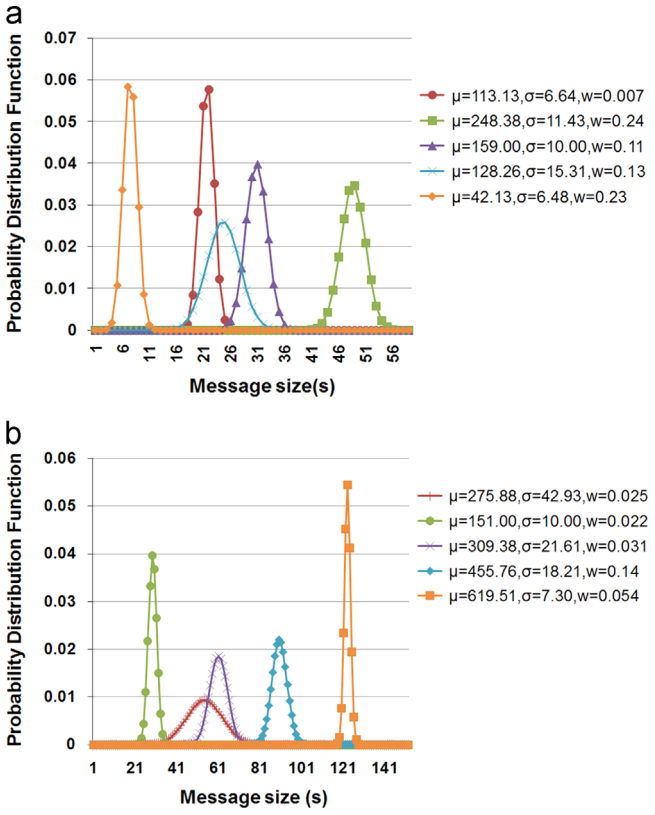

Fig. 7. PDFs associated with (a) Bittorrent and (b) HTTP for $L=2$.

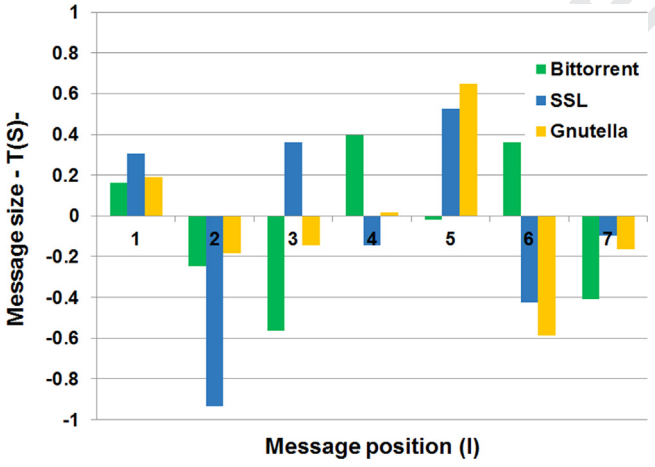

Fig. 8. Sequences of application message sizes in both directions for sample Bittorrent, SSL and Gnutella flows.

applications, as intuitively observed in the graph trends of Fig. 8. To prove the discriminative power of the proposed model for different applications, we have run many experiments using different datasets, as detailed next.

5.2.2. *Classification phase results*

In this section, we show the results of the classification method on 18 applications, 10 of which use TCP and 8 use UDP.

Various metrics (Khalife et al., 2014) can be used to evaluate classifiers. Basic ones include the rates of True Positive (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). Ideally, a good classifier is the one that maximizes TP and TN rates (the overall rate of correct classifications) while minimizing FP and FN rates (the overall rate of incorrect classifications).

To reflect the correlation with FP and FN, combined metrics are commonly used in multi-classification scenarios: precision (or accuracy) and recall (or sensitivity). Precision is the ratio of traffic instances correctly classified as class A to the total number of instances classified as class A. Recall is the ratio of traffic instances correctly classified as class A to the number of actual class A instances.

At this stage, it is important to explain that according to our experience, the correct assessment of the classification method should not be based on overall evaluation metrics (e.g. overall accuracy). This is especially true for datasets that have largely unbalanced numbers of elements from each class. This is the case in the considered dataset, which contains an overwhelming percentage of HTTP flows. In these situations, assessing a classification method based on the overall percentage of correctly classified flows is misleading, because it favors any classifier that is biased towards the most frequent application. Since the aim of a classifier is to detect with good percentage all the applications, a better assessment should be based on the worst case evaluation metric, that is, to assess the application that exhibits the least such metric.

For these reasons, we choose different metrics (Figs. 9–11) to evaluate our proposed classification model. These include precision and recall. We also consider classification results for various scenarios: Overall applications, TCP and UDP applications, per application and worst case application.

As a first insight into the results, in Fig. 9 we show the averaged recall, that is, the average of the recall rates for all the considered protocols, as a function of the number of observed messages per flow for TCP and for UDP applications. As shown, the recall increases up to around 96% for TCP applications when considering at least $L=3$ messages. Also in Fig. 9 we can notice a slight decrease in the accuracy for $L=6$, which needs some analysis. On the other hand, UDP average recall rates exhibits a bigger performance with a maximum of 99.02 for $L=4$.

As previously argued, these measures are not the best ones to assess the performance of the system, as they assume equal relevance to any application and a balanced dataset, which is not the case. Therefore, we consider the confusion matrices from which we derive some more meaningful figures. Thus, Table 2 shows the confusion matrix for TCP both as absolute values and relative to the number of samples in the test set for each application when using $L=3$.

As can be observed, there exist a non-negligible number of classification errors (Table 2a), mostly related to HTTP and SSL. Nevertheless, when the percentages for the number of flows in each class are considered, the results show high recall rates, with a minimum of 88.92 for MSN. The results are similar for higher values of L, except minor differences for $L=6$, as depicted in Fig. 10a), in which maximum,

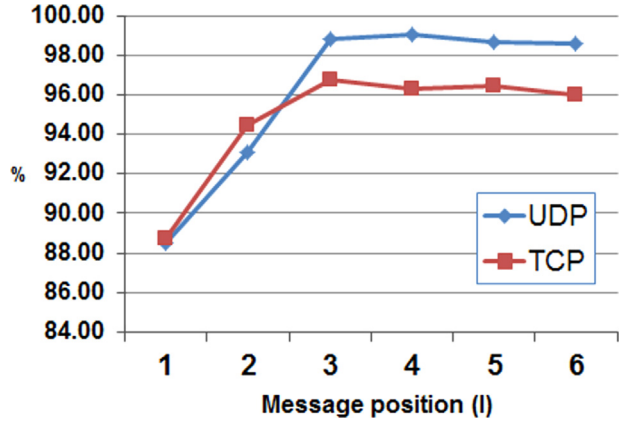

Fig. 9. Average classification recall for TCP and UDP applications.

Table 2
Confusion matrices for TCP applications for $L=3$: (a) Absolute number of samples and (b) percentage of samples relative to the total number of application samples (row).

Classified As	Input class									
	Bittorrent	iMESH	MSN	SSL	HTTP	Gnutella	Oscar	Mail_POP	FTP	Mail_SMTP
(a)										
Bittorrent	15,023	0	108	16	324	1	15	0	0	0
iMESH	0	1085	4	0	0	0	1	0	0	0
MSN	106	0	15,760	92	1727	22	12	1	3	0
SSL	838	0	580	216,551	3715	15	1433	0	0	0
HTTP	19,766	1	19,233	7764	1,527,055	3172	60	0	4780	0
Gnutella	1	0	4	2	104	1911	0	0	0	0
Oscar	0	0	1	5	6	0	1654	0	0	0
Mail_POP	0	0	3	0	9	0	0	5122	63	0
FTP	0	0	0	0	14	1	0	9	1328	0
Mail_SMTP	2	0	0	0	28	0	0	3	13	1995
(b)										
Bittorrent	97.00	0.00	0.70	0.10	2.09	0.01	0.10	0.00	0.00	0.00
iMESH	0.00	99.54	0.37	0.00	0.00	0.00	0.09	0.00	0.00	0.00
MSN	0.60	0.00	88.92	0.52	9.74	0.12	0.07	0.01	0.02	0.00
SSL	0.38	0.00	0.26	97.05	1.66	0.01	0.64	0.00	0.00	0.00
HTTP	1.25	0.00	1.22	0.49	96.54	0.20	0.00	0.00	0.30	0.00
Gnutella	0.05	0.00	0.20	0.10	5.14	94.51	0.00	0.00	0.00	0.00
Oscar	0.00	0.00	0.06	0.30	0.36	0.00	99.28	0.00	0.00	0.00
Mail_POP	0.00	0.00	0.06	0.00	0.17	0.00	0.00	98.56	1.21	0.00
FTP	0.00	0.00	0.00	0.00	1.04	0.07	0.00	0.67	98.22	0.00
Mail_SMTP	0.10	0.00	0.00	0.00	1.37	0.00	0.00	0.15	0.64	97.75

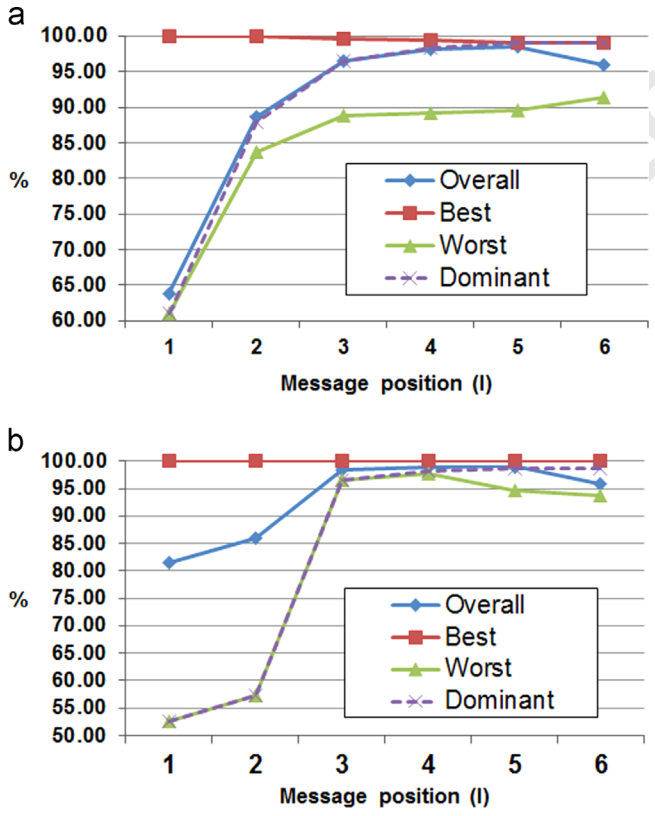

Fig. 10. Classification recall as a function of L: (a) TCP and (b) UDP.

minimum, overall and dominant (HTTP) recall rates are shown. It is noticeable with a slight drop in HTTP recall for $L=6$ which also induces a slight decrease in the overall recall for $L=6$ when compared with that for $L=5$. As previously mentioned, the number of training samples drops with L for some applications, as the number of available flows with at least this number of messages decreases. For $L=6$, there are three applications with a relatively low number of flows for training (below 300), when compared with those available for the other applications, which is clearly degrading the quality of the representations for these applications and introducing classification errors.

On the other hand, Table 3 shows the confusion matrix for UDP applications, both as absolute (Table 3a)) and relative (Table 3b) values. From these tables it is clear that the proposed system performs even better for UDP applications, with less non-null values out of the diagonal. The recall rates as a function of L are shown in Fig. 10b. As in the TCP case, there is a slight drop in the recall for $L=6$, which can be again explained by a bigger decrease in the training samples from some of the applications. In this case, it is the number of samples from Gnutella who drops from more than 200 for $L=2$ and $L=3$ to below 100 for $L=6$, while the decrease for the other applications is not that big in comparison. It is relevant to mention that the recall rate reaches up to 98.39% after $L=3$ messages are considered (recall that in the UDP case the messages are the packets themselves). This suggests that most UDP flows can be classified very quickly, with this being an early classification method.

A different insight into the results can be derived from the precision values. As shown in Fig. 11, despite the high recall rates, the values for the precision are not high for all the applications. This is clearly related to the highly unbalanced nature of the dataset that makes the results for the precision of those less frequent protocols highly sensitive to even small error rates for the dominant ones.

As a summary, the results show that using the proposed method it is possible to correctly classify up to around 98% of the TCP flows and 99% of the UDP sessions by only analyzing the first 3 to 5 messages.

In order to check the specificity of the obtained distributions, we also tested the models by applying them on the m57-patents public dataset (http://digitalcorpora.org/corpora/scenarios/m57-patents-scenario), where we applied the trained distributions obtained from our own dataset and applied them to detect the applications available in common with the m57-patents in sufficient number. These are SSL, HTTP and SMTP. Again, the overall classification results using our proposed model reached 96% of recall and 85% of precision. It is remarkable that no additional training nor tuning was made, which points to a good generalization of the obtained models.

Table 3
Confusion matrices for UDP applications for $L=3$: (a) Absolute number of samples and (b) percentage of samples relative to the total number of application samples (row).

Classified As	Input class							
	iMESH	Pando	MSN	NTP	RTP	Gnutella	STUN	NETBIOS
(a)								
iMESH	412	0	0	0	1	0	9	0
Pando	0	448	0	0	0	0	0	0
MSN	0	0	813	0	0	0	0	0
NTP	0	0	0	879	0	0	0	0
RTP	0	0	0	0	218	1	1	0
Gnutella	0	0	0	0	0	156	2	0
STUN	0	0	18	2	3	43	1906	1
NETBIOS	0	0	0	0	7	0	0	536
(b)								
iMESH	97.63	0.00	0.00	0.00	0.24	0.00	2.13	0.00
Pando	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
MSN	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
NTP	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
RTP	0.00	0.00	0.00	0.00	99.09	0.45	0.45	0.00
Gnutella	0.00	0.00	0.00	0.00	0.00	98.73	1.27	0.00
STUN	0.00	0.00	0.91	0.10	0.15	2.18	96.6	0.05
NETBIOS	0.00	0.00	0.00	0.00	1.29	0.00	0.00	98.71

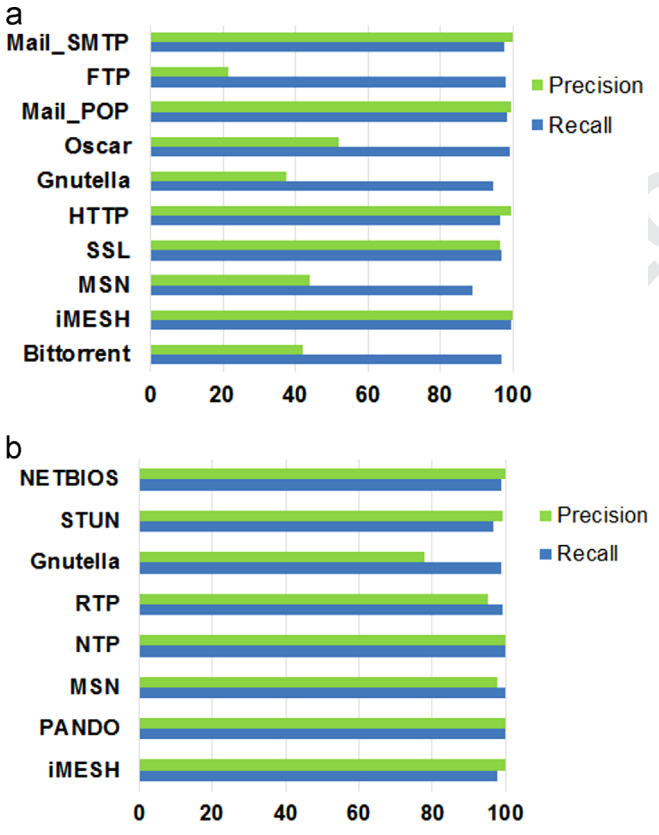

Fig. 11. Precision and recall values for each application ($L=3$): (a) TCP and (b) UDP.

6. Complexity analysis

As stated earlier, a key advantage of the classifier is that each application is profiled standalone without any involvement of other applications, which is ideal for extensibility, and particularly important in the context of network traffic classification. This is also important from the point of view of the computational complexity of the proposed method. But, previous to any cost analysis, it is important to distinguish between the two modes of operation of the system, that is, between training and classification. As previously mentioned, the training is made only once for each application, which means that the cost of training is linear with the number of applications (assuming equal number of samples in each class) and, more importantly, that it will not be the critical cost for the real usage of the system. On the contrary, in the normal usage, each time a flow is to be classified it should be analyzed by the system, this being the relevant cost for the effective deployment of the system.

Regarding the cost associated to the training, it is based on the estimation of Gaussian mixtures after a clustering of the data. According to Verbeek et al. (2003), the associated computational complexity is $O(k^2\, s)$, with k being the number of clusters and s the number of samples. According to Eqs. (6) and (17), a multipeak Gaussian with up to $K_{n,l}$ peaks is evaluated for each application (n) and state of the model (l) during the training phase. This means that $K_{n,l}$ clusters have to be estimated per application.

Therefore, the complexity of the training phase is $O((KL)^2 Ns)$ with:

- N the number of considered applications,
- L the number of messages analyzed from each flow,
- K the maximum number of clusters, and
- s the number of sampled flows per application.

The evaluation phase consists simply in evaluating the likelihood of the message size vector versus the Gaussian mixtures and finding the maximum, so the computational complexity to classify a flow is $O(NLK)$.

Finally, the storage requirements are quite limited, as each flow can be loaded alone for evaluation. Thus, only the N models need to be stored. As each model consists on LK Gaussians, each one with 3 parameters, Eqs. (16) and (17), the required storage is $O(3NLK)$.

7. Conclusions and future work

In this paper a new blind network traffic classifier has been presented. The classifier uses the sizes of the initial messages exchanged between the hosts involved in the communication as inputs. The classification is flow-based and can be considered an early classification method, as the number of messages required for the classification can be kept low. Unlike other similar approaches, this work focuses on the messages, not the packets, that is, which is considered a differential characteristic of a protocol is the sequence of sizes of the first messages, not the sizes of the initially exchanged packets. Although both approaches will be

almost the same for protocols with small message sizes, key differences may appear due to the potential segmentation of the messages into many packets.

Another differential characteristic of the proposed system is the use of multimodal distributions in an attempt to summarize all the possible methods included in a protocol in a single model. Thus, the classifier will consist of as many models as different applications it is able to detect. In this sense, previous similar works considered a reduced number of protocols (around 4–6 protocols) while in our proposal up to 18 different applications have been explored.

The experimental results are promising and an improvement over similar systems has been demonstrated. Nevertheless, more extensive experiments using bigger datasets are required in order to be able to improve the system, increasing the confidence and representativeness of the models and enabling the use of new and better heuristics. This is not an easy challenge, as the data to be used has to be properly labeled and has to be big enough. In our experiments, more than 200 GB of labeled data has proven to be insufficient for some protocols and only 18 of them could be used with some confidence. Another relevant challenge for the real usage of the system is related to the completeness of the models and the classification of a flow not belonging to any of the trained classes. In this case, it is necessary to develop a rejection mechanism.

We also suggest as candidate for future work the tracking of host-to-host activities in order to affect accordingly the prior probabilities of the Bayesian classifier. We expect interesting improvements from this approach, though at the cost of acceptable computational overhead.

References

Auld T, Moore AW, Gull SF. Bayesian neural networks for internet traffic classification. IEEE Trans Neural Netw 2007;18:223–39. http://dx.doi.org/10.1109/TNN.2006.883010.

Crotti M, Dusi M, Gringoli F, Salgarelli L. Traffic classification through simple statistical fingerprinting. ACM SIGCOMM Comput Commun Rev 2007;37:5–16. http://dx.doi.org/10.1145/1198255.1198257.

Dainotti A, de Donato W, Pescapē A, Salvo Rossi P. Classification of network traffic via packet-level hidden markov models. In: Proceedings of the IEEE global telecommunications conference GLOBECOM; 2008. p. 1–5. http://dx.doi.org/10.1109/GLOCOM.2008.ECP.412.

Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B (Methodol) 1977;1(39):1–38.

Deep packet inspection-technology, applications and net neutrality ⟨http://www.ipoque.com⟩ (accessed on 9 April 2013).

Erman J, Mahanti A, Arlitt M, Williamson C. Identifying, discriminating between web, peer to peer traffic in the network core. In: Proceedings of the 16th international conference on world wide web WWW'07; 2007. p. 883–92. http://dx.doi.org/10.1145/1242572.1242692.

Erman J, Mahanti A, Arlitt M, Cohen I, Williamson C. Offline/realtime traffic classification using semi-supervised learning. Perform Eval 2007;64:1194–213. http://dx.doi.org/10.1016/j.peva.2007.06.014.

Gu C, Zhuang S. A novel P2P traffic classification approach using back propagation neural network. In: Proceedings of the 12th IEEE international conference on communication technology ICCT. IEEE; 2010. p. 52–5. http://dx.doi.org/10.1109/ICCT.2010.5689171.

Huang S, Chen K, Liu C, Liang A. A statistical-feature-based approach to internet traffic classification using machine learning. In: Proceedings of the international conference on ultra modern telecommunications & workshops ICUMT; 2009. p. 1–6. http://dx.doi.org/10.1109/ICUMT.2009.5345539.

Iacovazzi A, Baiocchi A. Internet traffic privacy enhancement with masking: optimization and tradeoffs. IEEE Trans Parallel Distrib Syst 2014;25:353–62. http://dx.doi.org/10.1109/TPDS.2013.42.

Jaber M, Cascella RG, Barakat C. Enhancing application identification by means of sequential testing. In: NETWORKING; 2009. p. 287–300. http://dx.doi.org/10.1007/978-3-642-01399-7_23.

Jaber M, Cascella RG, Barakat C. Can we trust the inter-packet time for traffic classification?, In: Proceedings of the international conference on communications (ICC); 2011. p. 1–15. http://dx.doi.org/10.1109/icc.2011.5963024.

Jaber M, Cascella RG, Barakat C. Using host profiling to refine statistical application identification. In: Proceedings of INFOCOM; 2012. p. 2746–50. http://dx.doi.org/10.1109/INFCOM.2012.6195692.

JinSong W, Yan Z, Qing W, DongYi W. Connection pattern-based P2P application identification characteristic, In: Proceedings of IFIP International Conference on Network and Parallel Computing; 2007. p. 437–41. http://dx.doi.org/10.1109/NPC.2007.62.

Karagiannis T, Papagiannaki K, Faloutsos M. BLINC: multilevel traffic classification in the dark. In: Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications (SigComm); 2005. p. 229–40. http://dx.doi.org/10.1145/1090191.1080119.

Khalife J, Hajjar A, Diaz-Verdejo J. A multilevel taxonomy and requirements for an optimal traffic-classification model. Int J Netw Manag 2014;24:101–20. http://dx.doi.org/10.1002/nem.1855.

Lacovazzi A, Baiocchi A. Optimum packet length masking. In: Proceedings of the 22nd international teletraffic congress (ITC); 2010. p. 1–8. http://dx.doi.org/10.1109/ITC.2010.5608728.

Li J, Zhang S, Lu Y, Yan J. Real-time P2P traffic identification. In: Proceedings of the IEEE global telecommunications conference GLOBECOM; 2008. p. 2474–8. http://dx.doi.org/10.1109/GLOCOM.2008.ECP.475.

Lu W, Ling X. A heuristic-based co-clustering algorithm for the internet traffic classification. In: Proceedings of the advanced information networking and applications workshops (WAINA); 2014. http://dx.doi.org/10.1109/WAINA.2014.16.

Moore A, Zuev D. Internet traffic classification using bayesian analysis techniques. In: Proceedings of the international conference on measurement and modelling of computer systems SIGMETRICS; 2005. p. 50–60. http://dx.doi.org/10.1145/1064212.1064220.

Moore A, Zuev D, Crogan M. Discriminators for use in flow-based classification. Technical report. Queen Mary University of London ⟨http://www.eecs.qmul.ac.uk/tech_reports/RR-05-13.pdf⟩ (last accessed on 23 July 2015).

Mujtaba G, Parish DJ. Detection of applications within encrypted tunnels using packet size distributions. In: Proceedings of the international conference for internet technology and secured transactions, ICITST; 2009. p. 1–6. http://dx.doi.org/10.1109/ICITST.2009.5402624.

The 2009-M57-Patents Scenario Pcap Dataset ⟨http://digitalcorpora.org/corpora/scenarios/m57-patents-scenario⟩ (accessed 28 January 2014).

Pironti Alfredo. Monitoring message size to break privacy. In: Proceedings of the STRINT workshop, 2014.

Rizzi A, Colabrese S, Baiocchi A. Low complexity, high performance neuro-fuzzy system for internet traffic flows early classification. In: Proceedings of the wireless communications and mobile computing conference (IWCMC); 2013. http://dx.doi.org/10.1109/IWCMC.2013.6583538.

Shannon C, Moore D, Keys K. Internet measurement data catalog. In: ACM computer communications review ⟨http://imdc.datcat.org/collection/1-070W-6=tcrsg-collection⟩ (last accessed on 24 July 2013).

Tabatabaei T, Karray F, Kamel M. Early internet traffic recognition based on machine learning methods. In: Proceedings of the 25th IEEE Canadian Conference on Electrical & Computer Engineering CCECE; 2012. p. 1–5. http://dx.doi.org/10.1109/CCECE.2012.6335034.

Verbeek JJ, Vlassis N, Kröse B. Efficient greedy learning of gaussian mixture models. Neural Comput 2003;15:469–85. http://dx.doi.org/10.1162/089976603762553004.

Waizumi Y, Tsukabe Y, Tsunoda H, Nemoto Y, Tanaka K. Network application identification based on communication characteristics of application messages. J Commun Comput 2011;8:111–9.

Wang X, Parish D. Optimised multi-stage TCP traffic classifier based on packet size distributions. In: Proceedings of the third international conference on communication theory, reliability, and quality of service; 2010. p. 98–103. http://dx.doi.org/10.1109/CTRQ.2010.24.

Wang P, Guan X, Qin T. P2P Traffic identification based on the signatures of key packets. In: Proceedings of the IEEE 14th international workshop on computer aided modeling and design of communication links and networks, CAMAD; 2009. p. 1–5. http://dx.doi.org/10.1109/CAMAD.2009.5161471.

Wu X, Li W, Liu F, Yu H. Packet size distribution of typical internet applications. In: Proceedings of the international conference on wavelet active media technology and information processing (ICWAMTIP); 2012. p. 276–81. http://dx.doi.org/10.1109/ICWAMTIP.2012.6413493.

Yang Z, Li L, Ji Q, Zhu Y. Cocktail method for BitTorrent traffic identification in real time. J Comput 2012;7:85–95. http://dx.doi.org/10.4304/jcp.7.1.85-95.

Yildirim T, Radcliffe P. A framework for tunneled traffic analysis. In: Proc. of the 12th international conference on advanced communication technology (ICACT), vol. 2; 2010. p. 1029–34. http://dx.doi.org/10.2307/2984875.

Zander S, Nguyen T, Armitage G. Automated traffic classification, application identification using machine learning. In: Proceedings of the IEEE conference on local computer networks (LCN 05); 2005. p. 250–7. http://dx.doi.org/10.1109/LCN.2005.35.

Zhenxiang L, Mingbo H, Song L, Xin W. Research of P2P traffic comprehensive identification method. In: Proceedings of the 2011 international conference on network computing and information security (NCIS), vol. 1; 2011. p. 307–10. http://dx.doi.org/10.1109/NCIS.2011.69.

Zink T, Waldvogel M. Bittorrent traffic obfuscation: a chase towards semantic traffic identification. In: Proceedings of the IEEE 12th international conference on peer-to-peer computing; 2012. p. 126–37. http://dx.doi.org/10.1109/P2P.2012.6335792.

Please cite this article as: Hajjar A, et al. Network traffic application identification based on message size analysis. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.10.003