Performance of OpenDPI in Identifying Sampled Network Traffic

Jawad Khalife and Amjad Hajjar
Lebanese University/Faculty of Engineering, IT department, Beirut, Lebanon
Email: jawad_khalife@hotmail.com, arhajjar@idm.net.lb

Jesús Díaz-Verdejo
University of Granada/Department of Signal Processing, Telematics and Communication, Granada, Spain
Email: jedv@ugr.es

***Abstract*—The identification of the nature of the traffic flowing through a TCP/IP network is a relevant target for traffic engineering and security related tasks. Despite the privacy concerns it arises, Deep Packet Inspection (DPI) is one of the most successful current techniques. Nevertheless, the performance of DPI is strongly limited by computational issues related to the huge amount of data it needs to handle, both in terms of number of packets and the length of the packets. One way to reduce the computational overhead with identification techniques is to sample the traffic being monitored. This paper addresses the sensitivity of OpenDPI, one of the most powerful freely available DPI systems, with sampled network traffic. Two sampling techniques are applied and compared: the per-packet payload sampling, and the per-flow packet sampling. Based on the obtained results, some conclusions are drawn to show how far DPI methods could be optimised through traffic sampling.**

***Index Terms*— network traffic identification, deep packet inspection, optimisation, payload truncation, flow truncation, traffic sampling**

I. INTRODUCTION

Network traffic identification aims to classify packets (packet-based identification) or flows (flow-based identification) in a given network according to the associated application protocol. Traditionally, this task has been considered quite simple as ports were assigned for many application protocols. In this scenario a simple inspection of transport layer header suffices to identify the underlying protocol. Nevertheless, this situation is changing, thus making traffic identification a hot research topic, as some Internet applications, such as P2P, are becoming more and more challenging to identification techniques by using port obfuscation, encryption, and/or tunnelling [1]. One of the most successful methods currently available to identify traffic is based on the examination of the payloads to find known protocol patterns or signatures (e.g. "GET * HTTP"). This is the so-called DPI (Deep Packet Inspection) [2].

However, in today's networks, performance and privacy issues are two important factors that are considered some of the weaknesses of DPI. On the other hand, DPI is not able to inspect ciphered payloads. This fact is pushing researchers for alternate solutions in which P2P identification is still considered a complex task, especially when DPI is not involved at all.

As such, one of the current research trends is to optimise current DPI based identification methods characterised by their high accuracy, while keeping at the same time an acceptable level of user privacy and performance.

One of the DPI optimisation means is to reduce the input size through traffic sampling. Although different sampling policies exist [3], in this work, we applied sampling techniques at two different levels:

- Per-packet sampling: (or payload truncation) this is performed on the packet level, through partially inspecting the payload of each packet.
- Per-flow sampling: (or flow truncation) this is performed on the flow level, through inspecting the full payloads of a subset of packets per flow.

While sampling obviously provides a significant impact on the processing times by reducing the size of the input to process, it may have an unexpected impact on the traffic classification.

However, what impact the traffic sampling process would have on DPI classification accuracy, and which is the preferred sampling technique to use in optimising DPI, are important questions that we try to answer through this work.

In an attempt to answer these questions, we present in this paper, a study on the effect of traffic sampling on identification accuracy by using one of the best DPI-based tools: OpenDPI [4]. Our conducted identification experiments were based on full payload dataset traffic as captured through an institution's Internet link. We tested OpenDPI accuracy with per-packet sampling (using incremental payload truncation lengths) and with Per-flow sampling (using different number of sampled packets per flow), keeping three goals in mind:

- To provide protocol oriented results for classification accuracy.
- To compare the effect of both traffic sampling techniques on OpenDPI accuracy and their required input.
- To draw conclusions on how far combined DPI methods could be optimised through traffic sampling.

The remaining of this paper is organised as follows. Section 2 provides an overview of payload based

doi: 10.4304/jnw.8.1.71-81

identification tools, methods and optimisations. Section 3 describes OpenDPI tool in the way it analyses and labels packets and flows. Section 4 provides a description of the testbed we used for the experiments. Our conducted experiments and the obtained results in running the OpenDPI tool, both with per-packet and per-flow sampling techniques, are shown in Sections 5 and 6. Section 7 compares the obtained results. Finally, Section 8 presents some conclusions and future work.

II. Identification Of Flows Based On Payloads

Deep Packet Inspection" (DPI) is defined in [2] as being "*…a computer networking term that refers to devices and technologies that inspect and take action based on the contents of the packet (commonly called the "payload") rather than just the packet header.*"

The most important parts of DPI are regular expression matching and signature based scanning. In this technique the payload of all the packets is checked against the set of known protocol signatures.

Some well-known DPI technology based tools are OpenDPI [4], an open source traffic classification tool, L7-filter [5], an open source application layer classifier for Linux's Netfilter, and Snort [6], an open source network intrusion prevention and detection system. In this paper, our choice was to use the OpenDPI tool since it includes the latest DPI technology combined with other techniques making it one of the most accurate classifiers.

Many authors attempted to enhance DPI accuracy by combining it with other methods, such as behavioural [7], statistical [8], port based [9] and DFI (Deep Flow Identification) based methods [10].

On the other hand, many recent works attempted to optimise DPI performance for high link speeds. Some of them apply software based optimisation focused on enhancing DPI algorithms, e.g. [11][12][13], while others use hardware based optimisation e.g. [14].

In this paper, we will focus on a software optimisation which consists on reducing the size of DPI input through partial payload inspection. In this context, different methods were proposed in the literature. For instance, ML (Machine Learning) identification methods [1] use the feature selection algorithm. On the other hand, sampling techniques are more general and easy to implement as they just try to reduce the size of the input data by simply taking samples or parts from the data according to a given criteria. This later approach could be jointly applied with DPI. In fact, this is the scenario considered in this work.

Sampling network traffic is the process of taking partial observations from the monitored traffic, and drawing conclusions about the behaviour of the system from these sampled observations. They are mainly used for network management and monitoring [15] although may also be used in classification tasks e.g. [9][16][17]. As many works [3][18][19] show, sampling techniques can be integrated within the traffic classification process.

Apparently, few works apply sampling to network traffic classification. A detailed taxonomy of sampling techniques according to the used method is provided in [15]. Another way of categorising sampling techniques is related to the target considered by the method. From this point of view, they can be classified as per-packet payload sampling [9][16], i.e. sampling bytes from within the packet payload, per-flow packet sampling [3][15][20][21], i.e. sampling a subset of packets from within the whole traffic flow, or a combination of both [17].

Per-packet sampling was shown in [16], where authors proposed a novel approach that brings the sampling idea to the regular expression field. Their approach, called payload sampling, allows skipping a large portion of the text in the payload, thus processing less bytes. Their results show that the sampling approach is faster than previous advanced solutions. However, the price to pay is a slight number of false alarms which require a confirmation stage.

Another example of per-packet byte sampling was shown in [9] which also combined the port-based method with the DPI approach. Using L7-Filter [5] DPI tool, one of the paper's targets was to study the amount of payload information actually relevant in successful DPI matches.

For each session, L7-Filter attempts to match its regular expression rules against the stream of payload every time a new packet is seen. Their experimental results showed that 72% of the total attempts happen at the first packet of a flow. Moreover, they computed the offset of matching regular expression's first character and last character from the beginning of the packets respectively containing them. They showed that almost all matching strings start (99.98%) and finish (90.77%) in the first 32 bytes of payload.

Per-flow sampling [3] for DPI classification is shown in many papers using different sampling techniques such as: sampled NetFlow [20], related sampling [21], Bloom filters [22][23], k-ary sketch [24], and mask-match sampling [25]. In [27], we studied the effect of per-flow sampling on DPI classification accuracy and showed that more than 90% of OpenDPI classification accuracy is maintained by sampling the first ten packets of each flow.

The combination of per-flow and per-packet sampling is addressed in [17]. In this work the authors combined both sampling methods through the so-called LW-DPI. Results showed that most flows can be classified with only their first 7 packets or a fraction of their payload.

Rather than presenting an exhaustive list of comparisons of existing per-packet or per-flow sampling policies, we preferred to compare at higher level, that is, by choosing one representative technique from each category for comparison purposes. The chosen techniques were designed to focus on sampling the first payload chunks: the first bytes of each packet's payload and the first packets of each flow. This is supposed to be an efficient yet distinguished sampling method yielding up to increased computational gain especially for large flows. In fact, most works in the literature used continuous sampling rates, which implies that the number of sampled packets will increase as long as the flow is under course, while it is fixed to a predefined number with the per-flow

sampling approach used in this work (as detailed in Section 6).

Comparison is based on two main criterions: the effect of sampling on the classification accuracy and the required input size. We consider both sampling techniques as eventual means of DPI optimisation as the size of input will be reduced by only inspecting the truncated part of the packet payload, with per-packet sampling, and a subset of packets per flow, with per-flow sampling.

III. OPENDPI

As previously stated, the tool of choice for the classification of traffic is openDPI [4], which is derived from the commercial PACE product from Ipoque [26]. On 2009, Ipoque announced that it succeeded to win a test of deep packet inspection (DPI) bandwidth management solutions for monitoring and regulating peer-to-peer (P2P) traffic conducted by the European Advanced Networking Tester Centre (EANTC). Test results yield up to 99% detection and regulation accuracy for all popular P2P protocols.

The core of OpenDPI is a software library designed to classify internet traffic according to application protocols. In [4] the authors explain that OpenDPI incorporates different techniques such as behavioural (by searching for known behavioural patterns of an application in the monitored traffic) and statistical analysis (by calculating some statistical indicators that can be used to identify transmission types, as mean, median and variation of values used in behavioural analysis and the entropy of a flow).

Therefore, OpenDPI is not a pure-DPI product as it is not only signature-based but also incorporates information from other sources. This way, the classification accuracy is improved (no false classification according to Ipoque's claims), although some packets and flows still remain unclassified. This, together with the availability and quality of the signatures, made us to select OpenDPI instead of any other similar product.

In its current version, up to 101 different protocols can be identified, including the most common ones as SIP (Session initiation protocol), DNS (Domain Name Service), HTTP (Hypertext Transfer Protocol), HTTPS (Secure HTTP), FTP (File Transfer Protocol), and P2P protocols such as eDonkey, DirectConnect, Bittorrent etc.

Nevertheless, and according to its functioning, the capabilities of OpenDPI are mainly limited by the need to analyse the whole payload of all the packets in a flow in search of signatures (DPI behaviour) and to extract the behavioural and statistical information from the flows. Therefore, it is a basically full payload / full flow analysis which imply a high computational cost. This way, it would be desirable to reduce the size of the explored data in order to reduce this computational cost, but without degrading the performance of the classifier.

In this context, the target of this paper can be stated as analysing how sensitive are the mechanisms involved in OpenDPI to the per-packet and per-flow sampling techniques.

IV. TESTBED

In order to evaluate the effects of truncating the payloads in the traffic identification task, we have developed an experimental setup built from two main components. These components are a database of real traffic captured in an academic network, and a tool to automatically classify packets and flows according to their payloads by primarily using Deep Packet Inspection (DPI) which is based in OpenDPI.

Therefore, we have built a tool based on the OpenDPI library which is able not only to identify the application protocols but also to follow and differentiate the packets in each flow. To be able to handle UDP packets, we have generalized the concept of flow through the use of sessions. Sessions are considered as defined by the exchange of information associated to a tuple (IP addresses, ports and transport protocol) [4]. Nevertheless, throughout this paper, we will use the term flow to refer to a session, unless explicitly stated. It was convenient not only to apply sampling techniques on TCP sessions, but on UDP sessions as well. In fact, experiments show that application signatures are detected within UDP flows and that the classification accuracy is affected accordingly. Similarly to TCP based applications, classification results for UDP based applications (such as DNS and SIP) are protocol dependant, as shown in section 6.

As the output of the tool, two levels of classifications are provided: flow-based (each flow is labelled) and packet-based (each packet is also labelled). The tool operates in batch mode.

On the other hand, the traffic database contains the data captured during 3 working days at the access link of a medium size institution. The network consists of one head office to which more than 50 branches are connected according to a star topology. Local application services such as Email, Web and DNS are hosted in the head office which aggregates and controls all traffic flows, generated by different branches, through one central firewall. Through this network, around 4000 concurrent users are usually connected and generating approximately 40000 concurrent flows.

The data acquisition was carried out at a border router in the head office in order to be able to monitor all incoming and outgoing traffic. Therefore, apart from the boundaries of the caption, flows are captured complete and in both directions. Table I highlights some figures of the database.

By using the customized OpenDPI tool over the whole database we have built the "ground truth", that is, the set of correctly labelled flows and packets that will be used as the reference when evaluating traffic sampling techniques, in the following sections, where the evaluation of the identification is measured in terms of accuracy [1], that is, the percentage of detected packets/flows in regard to the full payload case. This procedure is adopted under the assumption that DPI is the

best currently available method for traffic classification and that the number of errors is negligible. This is a common approach in the traffic identification field, the number of packets and flows that DPI is not able to classify being its major limitation. In fact, some flows are not classified by OpenDPI (labelled as *unknown*), even when inspecting complete flows with full packet payloads, i.e. without sampling. Evidently, when sampling techniques are applied, these flows will remain unclassified by OpenDPI. Nevertheless, when sampling is applied, some of the flows, classified at the ground truth, become unclassified. Consequently, in order to highlight the effect of sampling on classification accuracy, unknown flows at the ground truth level are not counted when evaluating the flow accuracy under sampling.

The results provided by the classification tool show up to 42 protocols, including known web protocols such as HTTP and FTP, voice over IP protocols, such as SIP, and P2P applications such as Bittorrent, etc. Most of these protocols have been identified in the database, being HTTP the most frequent one, while an important part of the flows and packets remain unclassified. The relative distributions of flows and packets for most relevant protocols are shown in Fig. 1. A first inspection evidences big differences among the properties or frequencies at flow and packet levels. Therefore, the results can be different depending on whether we focus at flow or packet levels.

In this work, we will evaluate sampling techniques and their accuracy from the point of view of flow classification, not individual packets. This is because classifying flows is semantically more significant and more adequate to most traffic engineering tasks. Furthermore, flow classification is more efficient as all the packets in a flow will be classified including even those that do not contain any application-specific signatures or patterns.

V. Truncation Of The Payloads

In this section, we will show the conducted experiments in running OpenDPI on partially truncated packet payloads using the per-packet sampling technique.

Our main targets at this level are, as mentioned in Section 1: *To provide protocol oriented results for accuracy as a function of the sampled input (truncation length), and to show to what extent could the payload truncation affect OpenDPI accuracy.*

Through packet truncation, we intend to partially inspect each packet's payload. The studied sampling technique is very simple: the classifier must parse only a specified length of bytes (called payload truncation length or S) within each packet's payload. This is supposed to decrease the global classification time for the whole traffic.

As such, the per-packet sampling scheme we used is defined as: "*First S Bytes per packet*"

A. Methodology for Truncation Experiments

To achieve our targets, we customized OpenDPI tool to be able to parse only a specified length of bytes within each packet's payload. In order to obtain granular results, our choice was to iterate with incremental truncation length values with step of D Bytes, ranging from 0 Bytes (no payload) to 1500 Bytes (full payload). We have chosen D=128 Bytes.

TABLE I.
FIGURES FOR THE CAPTURED TRAFFIC DATABASE.

Size of the database	~180 GB
Number of IP packet	278 Mpackets
Number of different IPs	822519
Number of flows	6.3 Mflows
Number of identified protocols	42

The complete dataset we captured is very huge (63 pcap files totalling 177 GB) to use for the classification experiments, joint with sampling. In fact, we needed to run the customized OpenDPI up to 15 times on the same set of capture files. Since the customized OpenDPI requires around 50 minutes in classifying 1GB of capture data, the tool was run on a subset of only 17 randomly selected files (totalling 45 GB, i.e. 25% of the number of pcap files) due to time constraint.

On the other hand, those packets and flows that were not classified by OpenDPI when using the whole payload are dismissed and not considered in the figures and percentages that will be shown.

For this section, accuracy results are shown as a function of the truncation lengths and grouped according to three different sets: per protocol, per protocol group, and for all the protocols. For this purpose, the protocols were categorised into 12 groups that were defined according to [26].

B. Global results

The results obtained for all the protocols are shown in Fig. 2, where we show the number of successfully classified packet (in red) and flows (in blue) as a function of the length of the sample from each packet payload. At packet level, a sudden drop in the accuracy for truncation lengths lower than 1408 Bytes is observed. For 1280 Bytes, 47% of the packets were correctly classified, while for 1408 Bytes, 99% of all the packets were identified. On the other hand, the results at flow level show that for truncation length equal to 512 Bytes, 57% of total flows

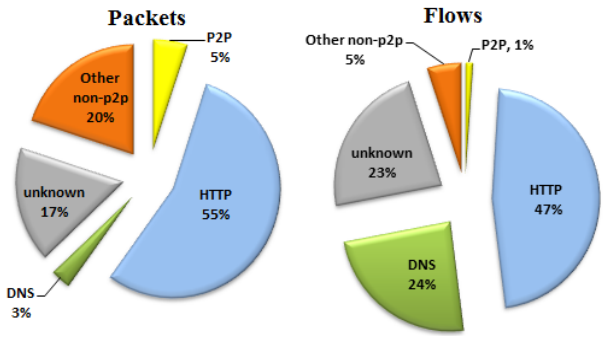


Figure 1. Distribution of packets (left) and flows (right) for most relevant protocols or groups of protocols.

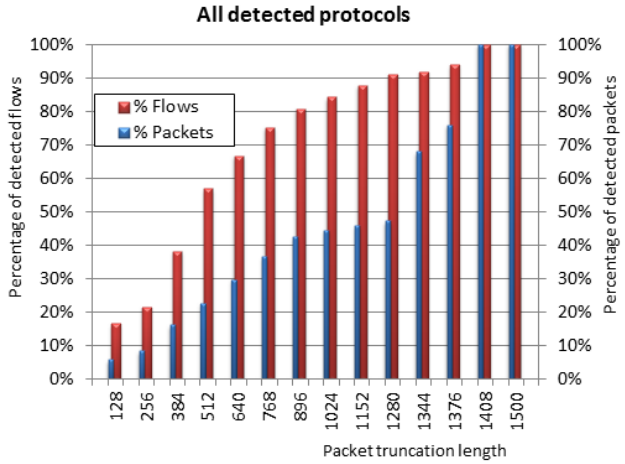


Figure 2. Global results for classification accuracy as a function of the truncation length of the payloads.

were detected, while for 1280 Bytes, 91% of flows were detected. Therefore, the analysis is more tolerant to payload truncation at flow levels than at packet levels.

Thus, truncation length must be at least 1280 Bytes to reach 50% of both flow and packet accuracy. This is not a very encouraging result for DPI optimisation through payload truncation as reducing only 15% of payload input would lead to a 50% drop in OpenDPI packet accuracy. However, results are encouraging if only flow accuracy is the main concern since still 57% of flows can be detected for 512 bytes of truncation.

From a macroscopic point of view, OpenDPI showed a common behaviour for all protocols:

- The number of detected packets/flows is increasing as the truncation length increases.
- For truncation length equal to 512 Bytes, 57% of flows were detected while only 22% of packets were detected.

C. Results per protocol group

When varying the truncation length, OpenDPI shows different behaviour for different protocol groups.

As an example, results for web group packets and flows are shown in Fig. 3.a. Web group results show that truncation, though differently, is affecting both packet and flow accuracy. In addition, web packet accuracy seems to be more affected by truncation than flow accuracy. It's noticeable that packet classification accuracy drops to around 50% for 1280 Bytes while for flow accuracy it drops to 50% only if less than 512 Bytes are truncated.

A different behaviour is observed for other groups. For example, if we consider the IM (Internet Messaging protocols) group –Fig. 3.b– or DNS group –Fig. 3.c– the classification accuracy is only slightly affected by truncation. In fact, for a truncation length equal to 256 Bytes, more than 50% of both packets and flows are detected. The same applies for DNS packets and flows.

The results for P2P protocols exhibit a mixed behaviour –Fig. 3.d– as they are similar to those from the web group at packet level and to those from IM and DNS groups at flow level. In fact, packet accuracy drops to around 50% for 1280 Bytes while flow accuracy stays above 92% even for 128 Bytes only.

In summary, at a granular level, the experimental results showed different behaviour for OpenDPI with truncation for different protocols. This in fact could be based on two main factors: the stateful behaviour of some protocols combined with the detection algorithm used by OpenDPI which considers some behavioural and statistical information for the whole flow.

We can evidence this assertion if we examine the obtained results for the web and DNS protocol groups. Since DNS is a stateless protocol, flows with truncated packets can still be detected. On the other hand, as web is a stateful protocol, the detection of web flows drops for truncated packets. Though not shown, FTP results also were different since FTP protocol has a special behaviour.

Therefore, we can conclude that stateless protocols are less sensitive to payload truncation than stateful ones. Thus, optimising DPI/DFI methods through payload truncation could be more effective for stateless and P2P protocols.

For interpreting the differences between flow and packet results for the same protocol, flow results are considered more significant since undetected flows may contain a huge number of packets thus affecting packet accuracy. We also noticed that flows detected at higher truncation length mostly contain a huge number of packets.

As a result for per-packet sampling, studied in this section, unless just a few bytes (not more than 128 Bytes) were omitted from the end of the packet payload, payload truncation with combined DPI/DFI will lead to many unknown flows and packets. For instance, by inspecting the full packet payload and omitting the last 512 bytes, only 57% of flow accuracy can be maintained.

VI. Truncation of The Flows

In this section, we will show the conducted experiments in running OpenDPI on sampled flows using the per-flow sampling technique.

Our main targets at this level are, as mentioned in Section: *To provide protocol oriented results for accuracy as a function of the sampled input (number of inspected packets per flow) and to show to what extent could the flow truncation affect OpenDPI accuracy.*

For comparison purposes with per-packet sampling, we conducted per-flow sampling experiments to obtain results for the same protocol groups, shown in Figures 2 and 3 of the previous section.

The methodology we used for flow truncation is described in [27], where we intend only to inspect, within each flow, the packets whose ordinal number inside the flow is lower than a predefined threshold (N_{min}). With this sampling scheme, while inspecting only the first N_{min} packets of the flow for the purpose of classification, the classifier will still handle the remaining packets for the purpose of assigning them to the flow. The difference is that for these packets the inspection part is to be omitted, and this is where the concept of optimisation comes: Through flow-sampling, we emphasize on the inspection time as we consider it to be the only sensitive term to flow truncation. In other words, the sampling speed-up in terms of CPU processing time comes only from speeding-up the flow classification itself, but there is no gain in the operation of mapping packets to flows, as this operation is independent and untouched.

As such, the per-flow sampling scheme we used is defined as: "*N_{min} packets per flow*"

A. Methodology for Truncation Experiments

In this Section, we used the same OpenDPI customization we performed in [27], on which we run on a subset of randomly selected files from our original dataset, since the main dataset is very huge. This customization allowed us to output the packet ordinal number inside the flow the packet belongs to at which

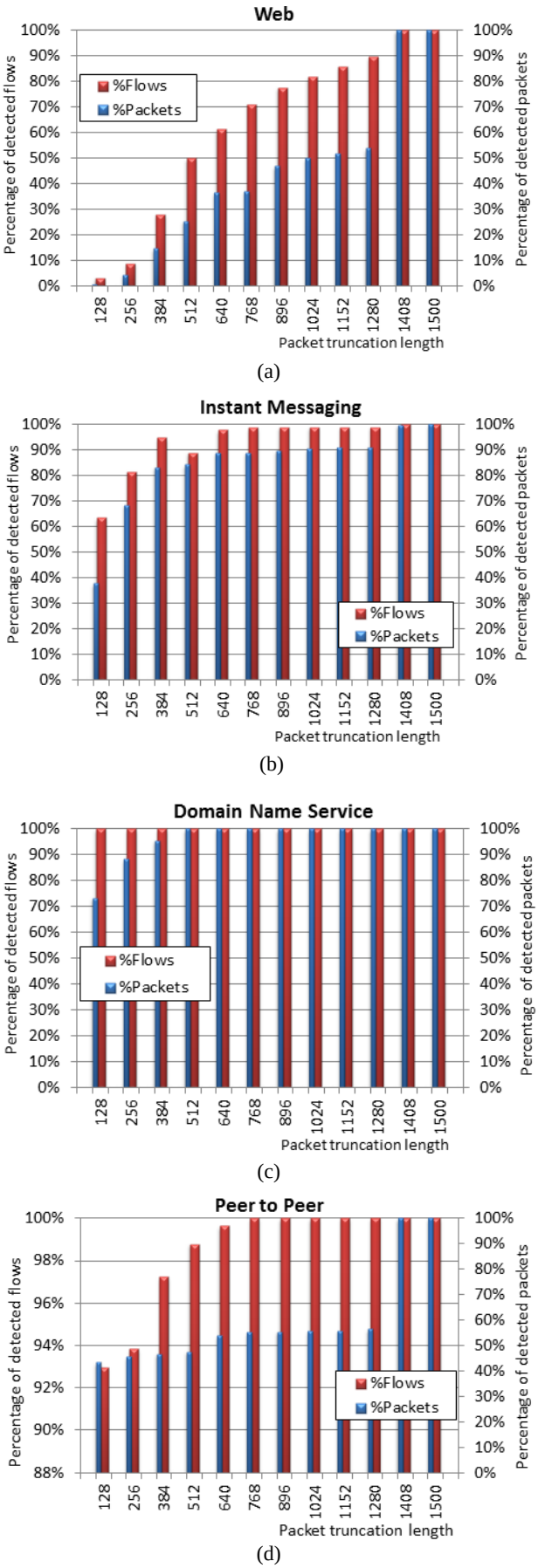


Figure 3. Results for various protocols/groups as a function of the truncation length for packets and flows. a) Web; b) Instant messaging; c) DNS; and d) P2P.

detection is achieved, referred to as *packet detection number* or *flow detection number*. In addition, we were able to generate accuracy results for different numbers of sampled packets per flow (N_{min}), without effectively truncating the flows.

As in the previous section, flow accuracy results are shown in terms of number of successfully classified flows as a function of the number of sampled packets from the beginning of the flow. Again, these results are grouped according to three different sets: per protocol, per protocol group, and for all the protocols.

B. Global Results

Fig. 4 shows the percentage of flows that have been classified vs. the number of sampled packets. As we can see, most flows are detected by inspecting the few first packets. Specifically, within the first ten packets (N_{min}=10), most protocols are being detected with an accuracy value of 99%. As also depicted in Fig. 4, flow accuracy is near 90% for N_{min}=4. Only a slight increase in accuracy is obtained for N_{min} values greater than 10. For these reasons, N_{min}=4 or N_{min}=10 could be considered critical values for the per-flow sampling scheme, according to the required level of accuracy and the required level of classification speed-up.

C. Results per Protocol Group

Results for the same protocol groups tested in the previous section are now shown in Fig. 5. The DNS group in Fig. 5.c seems to be the less sensitive protocol to flow truncation, as it's being classified by OpenDPI by inspecting solely the first packet with a 99% of classification accuracy. Other protocol groups are shown as well, like Web in Fig. 5.a, and Instant Messaging in Fig. 5.b. The same result as seen globally for all protocols persists: at least four packets are required to be inspected to reach an accuracy level of 90% and above. Results for P2P are shown in Fig. 5.d where 84% of accuracy can still be reached for N_{min} =4. If the classifier inspects the first ten packets of a P2P flow, 99.15% of classification accuracy can be reached as well.

D. Results per Protocol

The average packet detection number in the dataset is

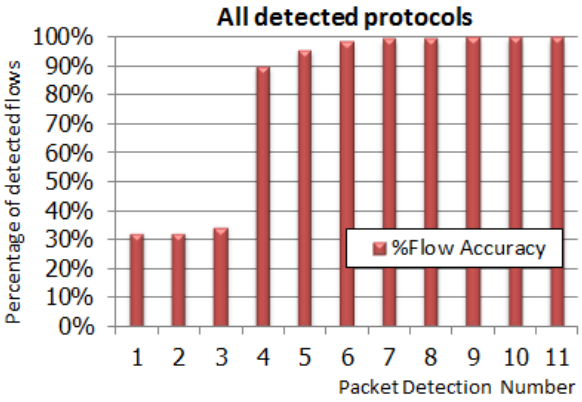
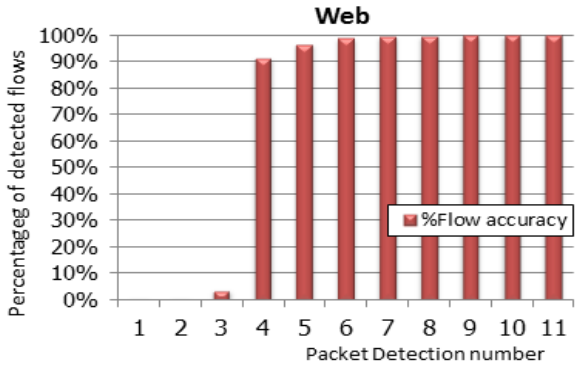
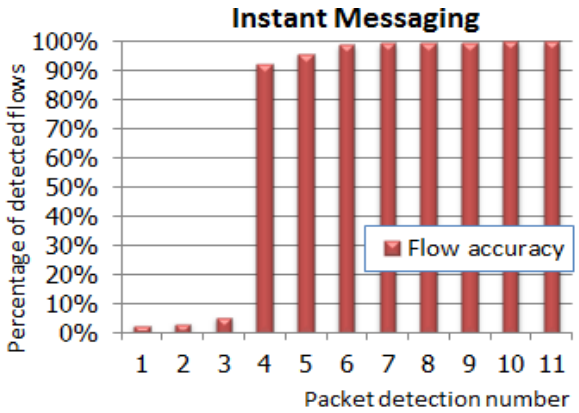


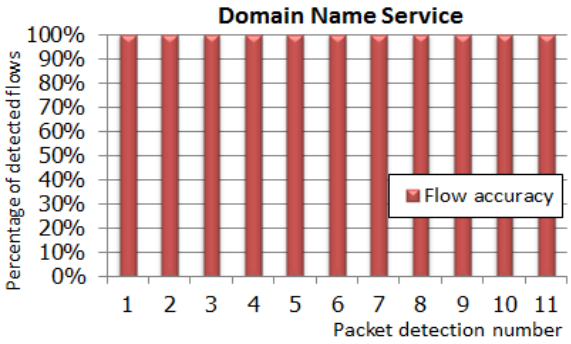
Figure 4. Global results for flow accuracy as a function of the packet detection number.



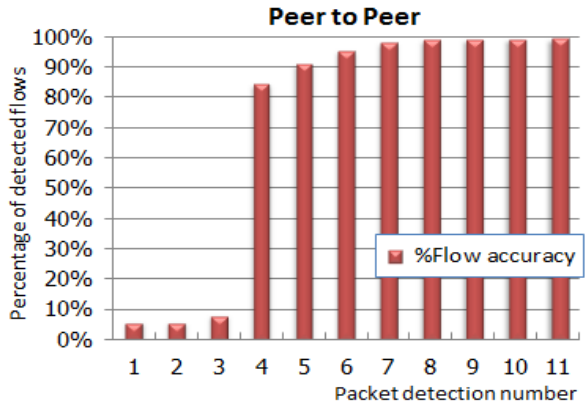
(a)



(b)



(c)



(d)

Figure 5. Flow accuracy results for various protocols/groups as a function of the packet detection number. a) Web; b) Instant messaging; c) DNS; and d) P2P.

shown in Fig. 6 for most common protocols.

Some protocols like iMESH and Bittorrent, show higher values than other protocols. We validated the fact that the presence of most deviation is due to flows that were under course during the start of the capture. Most protocols averages were below 10 packets. To validate this fact, we generated results per individual protocol. For instance, Fig. 7 shows the histogram of flow accuracy for some selected protocols like SIP (Fig. 7.a), FTP (Fig. 7.b), and HTTPS (Fig. 7.c). It can be noticed that about 90% of flow accuracy is reached by inspecting the first 6 packets.

As a result for per-flow sampling, studied in this section, inspecting the first 4 to 10 packets of a flow (as DPI input for inspection) could maintain the flow classification accuracy at high levels ranging from 90% to 99%.

In choosing the appropriate value of N_{min} for the classifier, two situations should be distinguished according to the classification target:

If the target is to classify only one specific protocol, N_{min} could be easily specified according to Fig. 6 (e.g. for HTTP, N_{min}=4). In this case, the classifier would inspect only the minimum number of packets, necessary for flow classification. However, if the target is to classify all protocols, which is the most common situation, N_{min} should be assigned the maximum value of the average packet detection number (N_{min}=10) in order to classify most protocols. In this case, and for protocols whose average packet detection number is lower than N_{min}, the classifier would inspect more packets than necessary.

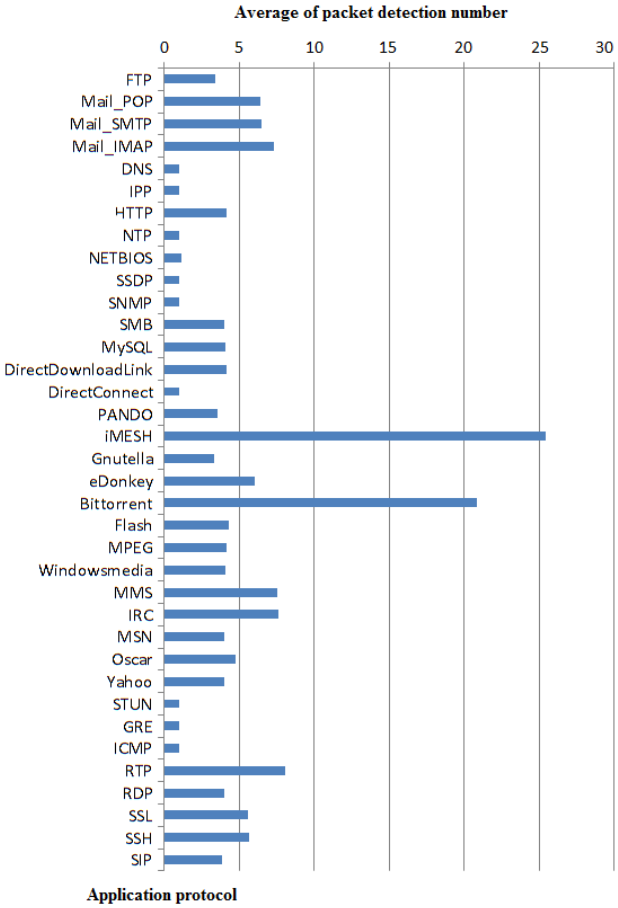


Figure 6. Average detection packet number for each individual protocol.

E. Computational Measurement

To highlight the optimisation aspect of sampling approaches, we choose to measure the computational gain in processing time for the per-flow sampling technique. Specifically, we measure the processing time consumed by the classification modules inside the classifier's code. As mentioned previously, through the flow sampling process, only the inspection time is optimised and not the packet handling time.

Experiments show that compared to full flow sampling, the per-flow sampling approach can provide 9% of computational time gain and 99% of classification accuracy, when only the first 20 packets (N_{min}=20) are inspected.

In comparing to EIM (Equidistant Invariable Mode) [3] having a sampling rate of 7/13, 36% of classification time can be saved due to inspecting less packets with the per-flow sampling approach (for N_{min}=20).

VII. Results Comparison And Analysis

A. Results Comparison

Table II shows the comparison results as summarized for both sampling techniques, according to the provided flow classification accuracy and the required DPI input. The percentage of input reduction is not shown in this table since it is dataset-dependent and can be simply estimated according to the average number of packets per flow, and the average packet size.

Results shown in Table II indicate that in order to obtain around 90% of flow accuracy, it's mandatory to inspect the first 1280 payload bytes of each packet (as DPI input for inspection), while inspecting the first 4 packets with full payload per flow is sufficient to obtain the same accuracy level. For higher accuracy results at 99%, the first 1408 of payload bytes are required per packet compared to the first 10 packets with full payload per flow.

For the dataset we used, the average number of packets

TABLE II.
Comparison between per-flow and per-packet sampling schemes used for DPI optimisation

Sampling Scheme used for DPI Optimisation	Required Input	Flow classification accuracy
Per-packet payload sampling	First 1280 bytes of payload, per packet	91%
	First 1408 bytes of payload, per packet	99%
Per-flow packet sampling	First 4 packets with full payload, per flow	90 %
	First 10 packets with full payload, per flow	99%

per flow is 45 packets and the average packet size is 233 Bytes. Thus, in order to obtain at least 90% of flow accuracy, 932 Bytes will be required for inspection with the per-flow sampling compared to 10,485 Bytes with the per-packet sampling.

Thus, compared to per-packet sampling, the per-flow sampling technique will provide higher flow classification accuracy at a cost of less input. Apparently, this applies only to traffic having, on average, more than 4 packets per flow and less than 1400 Bytes per packet, which is common for traffic of most known protocols, as shown in the dataset we used.

Moreover, per-packet sampling leads to many unknown packets, and the packet accuracy drops to 47% when the packet truncation length is 1280 bytes, as depicted in Fig. 2. Thus, optimising DPI/DFI methods through payload truncation could not be considered generally effective (especially when the set includes stateful protocols, which are the more affected).

Obviously, the best trade-off between the required DPI input size and the provided classification accuracy in Table II is with the per-flow sampling when only the first 4 packets with full payload are inspected per flow.

As a result, according to the sampling schemes we defined and regardless of the traffic dataset being tested, the following result can be generalized:

For DPI optimisation, the per–flow sampling technique is more convenient than the per-packet sampling technique, in terms of the required input and the provided classification accuracy.

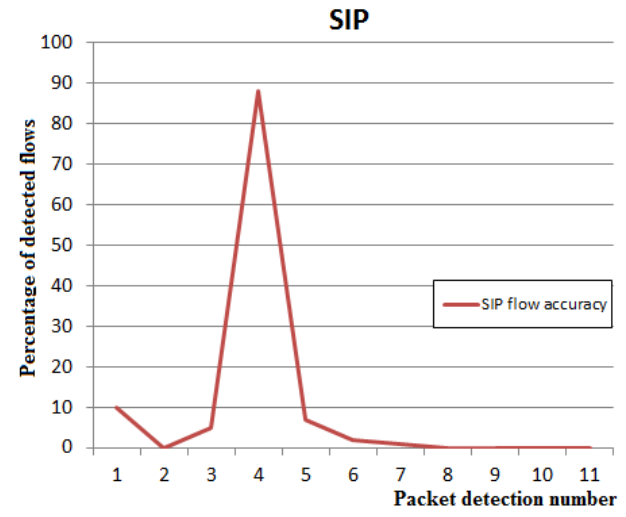
B. Results Analysis

One interpretation for the obtained result is that by combining DPI with other technologies (such as behavioural and statistical modelling), the task of DPI optimisation through per-packet sampling or payload truncation may render the identification method itself inefficient since the non-parsed part of the data may still be needed for the other added technology. However, to validate this assertion, further experiments are required, in which the pure DPI technique is to be separated from other helper technologies. The per-packet payload truncation can still be useful as an optimisation if, instead of classifying all the traffic, the target is to select some of them based on the application content and depending on the nature of the associated protocol.

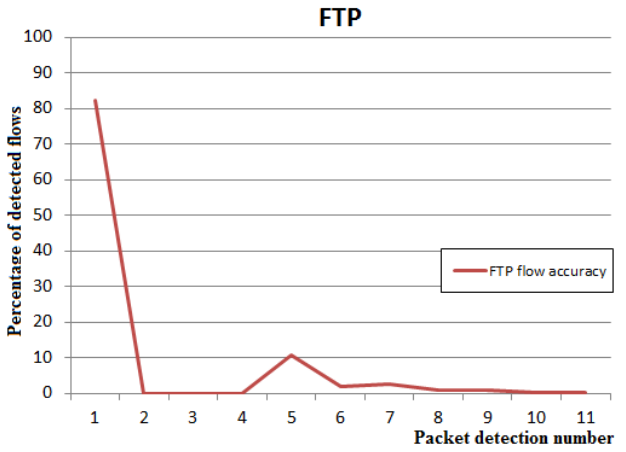
With the per-flow sampling, better results are obtained only if the first packets were sampled. This can be interpreted by the fact that the first packets usually signal the application protocol in use, during the first phase of flow set-up, and are thus of high importance for the classification process.

On the other hand, the default behaviour for PBFS (Packet Based Per Flow State) classifiers [28] is that the entire session must be “marked” as soon as one packet is detected to be holding an application signature.”

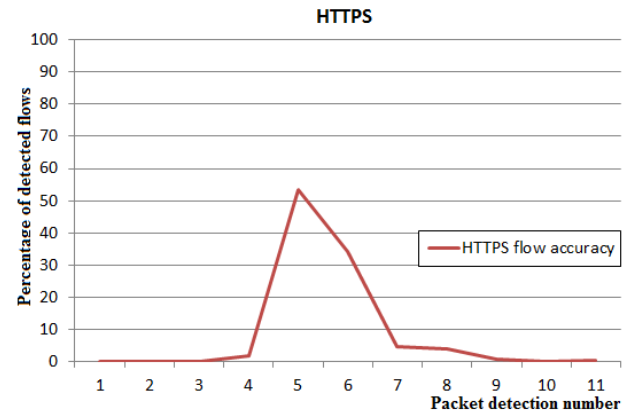
However, not all DPI classifiers are PBFS based, and for some particular cases, not all flows belonging to the same application protocol should be necessarily detected at the same packet detection number. For instance, when



(a)



(b)



(c)

Figure 7. Flow accuracy results for two selected protocols as a function of the packet detection number. a) SIP; b) FTP and c) HTTPS.

the packet holding the signature is delayed or lost, more packets will be inspected until the flow protocol is detected at higher packet numbers or simply marked as unknown. As for the OpenDPI tool, we were able to prove its PBFS like behaviour both theoretically, by interpreting the classifier’s code in Section 6.A, and

practically, by interpreting results for the average packet detection number (being nearly the same per protocol, in Fig. 6) and for the classification time gain (being moderate, in Section 6.E).

VIII. Conclusions and Future Work

In this paper we explored the effects of sampling on traffic classification accuracy using OpenDPI. Traffic sampling is considered one of the means of DPI optimisation as it reduces the required input size for the classifier. Two sampling techniques were tested and compared: per-packet sampling (through partial payload inspection) and per-flow sampling (through inspecting only a few packets per flow). Comparison is accomplished according to the reduction in the input size and the maintained classification accuracy.

Results show that flow accuracy is less sensitive to flow truncation than to packet payload truncation. With per-packet payload sampling, unless just few bytes (not more than the last 128 Bytes) were omitted during the packet payload inspection, per-packet sampling (or payload truncation) will lead to many unknown packets and flows. With per-flow packet sampling, inspecting the first 4 to 10 packets per flow could maintain flow accuracy at higher levels, ranging from 90% to 99%.

As a result, the per–flow sampling technique is more convenient than the per-packet sampling technique for DPI optimisation.

To provide more richness to this work, future enhancements may include comparing existing sampling methods within the same category (per-flow and per-packet), in which computational gain is evaluated in terms of both processing time and memory usage. Enhancements may involve as well other advanced DPI tools to discard any possible bias to OpenDPI, our best tool of choice.

Finally, in seeking for DPI optimisation through sampling, future enhancements should involve additional experiments which should integrate different techniques in order to define the optimal sampling scheme for the DPI classification process. Once the required input for the DPI classifier is optimally reduced, other DPI optimisation means, as found in the literature, could be used jointly to complete the job of DPI optimisation.

Acknowledgment

This work has been supported by Spanish MICINN under project TEC2008-06663-C03-02.

References

[1] T. Nguyen, G. Armitage, “A Survey of Techniques for Internet Traffic Classification using Machine Learning”, IEEE Communications Surveys & Tutorials, 2007, vol. 10, pp. 56-76, doi: 10.1109/SURV.2008.080406.

[2] Allot Communications, 2007. Digging Deeper Into Deep Packet Inspection (DPI). White paper. Available at http://www.dpacket.org 20.01.2012

[3] H. Chen, F. You, X. Zhou, and C. Wang, "The study of DPI identification technology based on sampling", ICIECS 2009, 2009, pp. 1-4, doi: 10.1109/ICIECS.2009.5363202.

[4] Opendpi. http://www.opendpi.org/ 20.01.2012

[5] L7filter. http://l7-filter.clearfoundation.com/ 20.01.2012

[6] Snort. http://www.snort.org 20.01.2012

[7] L. Zhang, D. Li, J. Shi and J. Wang, “P2P-based Weighted Behavioral Characteristics Of Deep Packet Inspection Algorithm, In Proc. of CMCE 2010, 2010, pp. 468-470, doi: 10.1109/CMCE.2010.5610457.

[8] F. Dehghani, N. Movahhedinia, M. Khayyambashi, and S. Kianian, “Real-time Traffic Classification Based on Statistical and Payload Content Features”, In Proc. IWISA 2010, 2010. pp. 1-4, doi: 10.1109/IWISA.2010.5473467.

[9] G. Aceto, A. Dainotti, W. de Donato, and A. Pescapé, "PortLoad: taking the best of two worlds in traffic classification”, In Proc. of INFOCOM 2010, 2010, pp. 1-5, doi: 10.1109/INFCOMW.2010.5466645.

[10] C. Wang, X. Zhou, F. You, and H. Chen, "Design of P2P Traffic Identification Based on DPI and DFI”, In Proc. of CNMT2009, 2009, pp. 1-4, doi: 10.1109/CNMT.2009.5374577.

[11] Y. Yang, H. Le, and V. Prasanna, “High Performance Dictionary-Based String Matching for Deep Packet Inspection”, In Proc. of INFOCOM 2010, 2010, pp. 1-5, doi: 10.1109/INFCOM.2010.5462268.

[12] P. Lin, Y. Lin, T. Lee, and Y. Lai, “Using String Matching for Deep Packet Inspection”, IEEE Computer, 2008, vol. 41, pp. 23-28, doi: 10.1109/MC.2008.138.

[13] G. La Mantia, D. Rossi, A. Finamore, M. Mellia, and M. Meo, “Stochastic Packet Inspection for TCP Traffic”, In Proc. ICC2010, 2010, pp. 1-6, doi: 10.1109/ICC.2010.5502280.

[14] A. Rao and P. Udupa, “A Hardware Accelerated System For Deep Packet Inspection”, In Proc. MEMOCODE’10, 2010, pp. 89-92, doi: 10.1109/MEMCOD.2010.5558646.

[15] R. Jurga and M. Hulbój, "Packet Sampling for Network Monitoring”, Technical Report, CERN | HP Procurve openlab project. Available at http://www.zdnetasia.com 20.1.2012

[16] D. Ficara, G. Antichi, A. Di Pietro, S. Giordano, G. Procissi, and F. Vitucci "Sampling Techniques to Accelerate Pattern Matching in Network Intrusion Detection Systems”, In Proc. ICC2010, 2010, pp. 1-5, doi: 10.1109/ICC.2010.5501751.

[17] S. Fernandes, R. Antonello, T. Lacerda, A. Santos, D. Sadok, and T. Westholm, "Slimming Down Deep Packet Inspection Systems”, In Proc. INFOCOM Workshops 2009, 2009, pp. 1-6, doi: 10.1109/INFCOMW.2009.5072188 .

[18] Z. Guo and Z. Qiu, "Identification Peer-to-Peer Traffic for High Speed Networks Using Packet Sampling and Application Signatures", In Proc. ICSP2008, pp. 2013-2019, doi: 10.1109/ICOSP.2008.4697540.

[19] M. Canini, D. Fay, D. Miller, A. Moore, and R. Bolla, "Per Flow Packet Sampling for High-Speed Network Monitoring”, In Proc. COMSNETS'09, 2009, pp. 1-10, doi: 10.1109/COMSNETS.2009.4808888.

[20] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification”, Computer Networks, Volume 55, Issue 5, 1 April 2011, pp. 1083-1099.

[21] M. Lee, M. Hajjat, R. Kompella, and S. Rao, "RelSamp: Preserving Application Structure in Sampled Flow Measurements”, In Proc. INFOCOM 2011, 2011, pp. 2354-2362, doi: 10.1109/INFCOM.2011.5935054.

[22] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep Packet Inspection using Parallel Bloom Filters", In Proc. High Performance Interconnects 2003, 2003, pp. 44-51, doi: 10.1109/CONECT.2003.1231477.

[23] Y. Li "Memory Efficient Parallel Bloom Filters for String Matching", In Proc. NSWCTC 2009, 2009, pp.485-488, doi: 10.1109/NSWCTC.2009.280.
[24] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, “Sketch-based change detection: methods, evaluation, and applications”, In Proc. of ACM SIGCOMM Internet Measurement Conference IMC’03, October 2003, doi: .
[25] R. Cong, J. Yang and G. Cheng, "Research of Sampling Method Applied To Traffic Classification", In Proc. ICCT 2010, 2010, pp. 112-115, doi: 10.1109/ICCT.2010.5689208.
[26] Ipoque. http://www.ipoque.com/ 20.01.2012
[27] J. Khalife, J. Verdejo, and A. Hajjar, “On the Performance of OpenDPI in Identifying P2P Truncated Flows”, AP2PS 2011, Lisbon, Portugal, 2011.
[28] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus "Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation", In Proc. ICC 2008, 2008, pp. 5869-5875.

Jawad khalife born in Lebanon, 1980, holds a master’s degree in telecommunications networks, University of Saint-Joseph, Beirut, Lebanon, 2003, and an engineering degree in computer and communication, the Lebanese University, Beirut, Lebanon, 2002. Since 2004, his teaching activities covered mainly network-related topics including network administration under Linux, Cisco and security courses in well-known faculties and institutions in Lebanon. Since 2002, he works as a Network Engineer in the central administration of the Lebanese university, Beirut, Lebanon. His current research interests covers enhancing traffic classification methods and their use in the security field, especially for intrusion detection systems. Eng. Khalife is a student member of IEEE.

Amjad S. Hajjar was born on May 12, 1964 in Chehim, Lebanon. He obtained his engineering diploma in electricity and electronics from the Lebanese university in 1986, and his Ph.D in computer-aided design (CAD) from the university of Paris-VI in 1992. He is currently assistant professor at the faculty of engineering of the Lebanese University, where he teaches computer networks, operations research and operating systems. His fields of interest in research are peer-to-peer (P2P) networks, traffic analysis and P2P activity detection.

Jesús Díaz Verdejo is a professor in the Department of Signal Theory, Telematics and Communications of the University of Granada. He received his B.Sc. in physics in 1989 and a Ph.D. degree in physics in 1995 from the University of Granada.

His initial research interest was related to speech technologies, especially automatic speech recognition. He is currently working on computer and network security, mainly focused in intrusion detection systems and traffic engineering from the point of view of security. He has also developed some work in telematics applications and e-learning systems.