On the Performance of OpenDPI in Identifying P2P Truncated Flows

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

Jesús Díaz-Verdejo
Dept. Signal Theory, Telematics and Commun.
University of Granada
Granada, Spain
jedv@ugr.es

***Abstract*—This paper aims to show the impact on classification accuracy and the level of computational gain that could be obtained in applying deep packet inspection on truncated peer to peer traffic flows instead of complete ones. Using one of the latest open source classifiers, experiments were conducted to evaluate classification performance on full and truncated network flows for different protocols, focusing on the detection of peer to peer. Despite minor exceptions, all the results show that with the latest deep packet inspection classifiers, which may incorporate different helper technologies, inspecting the first packets at the beginning of each flow, may still provide concrete computational gain while an acceptable level of classification accuracy is maintained. The present paper discusses this tradeoff and provides some recommendations on the number of packets to be inspected for the detection of peer to peer flows and some other common application protocols. As such, a new sampling approach is proposed, which accommodates samples to the stateful classifier's algorithm, taking into consideration the characteristics of the protocols being classified.**

Keywords-IP traffic classification; p2p; peer to peer; deep packet inspection; DPI optimization

I. INTRODUCTION

Traffic identification is a hot research topic, especially when it comes to complex Internet applications, such as P2P (peer to peer), using port obfuscation, encryption, and tunneling [1]. As they inspect the full packet payloads to match specific protocol patterns or signatures, DPI (Deep Packet Inspection) based methods [2], are characterized by the high level of classification accuracy they provide. However, the associated high computational cost and some user privacy issues arise some concerns regarding their use in real environments, especially in high-speed networks.

Optimizing DPI based methods, is thus becoming an important research trend attempting to enhance the classifier performance in terms of low computation, while maintaining an acceptable level of accuracy.

Reducing the input size required by the classifier through sampling can be considered one of DPI optimization means, which is not only supposed to reduce computational requirements, but also to ensure an acceptable level of user privacy. While different sampling policies exist, traffic sampling could be applied on two different levels: on the packet payload level, through partial packet payload inspection, and on the flow level, through inspecting only a few packets from within the complete traffic flow.

In [3], we studied how far DPI could be optimized through packet level sampling. Results showed that, unless just few bytes (not more than 128 Bytes) were truncated from the end of the packet payload, a sharp decrease in the accuracy will be apparent.

As part of our work in progress, this paper attempts to optimize DPI classifiers through flow sampling to which we will refer as flow truncation. It is important to note that we consider truncation as a particular case of the sampling technique, by inspecting a certain number of elements at the beginning of a stream (first bytes in a packet payload, first packets in a flow). We focus on P2P applications as we consider that identifying p2p traffic is one of the most complex classification tasks, especially when compared with other common application protocols. In fact, and as shown in [1], most works were emphasizing on their ability to detect p2p traffic as a key indicator of the quality of the classifier, and most importantly, were providing a better understanding of P2P traffic characteristics as part of the detection mechanism.

While flow sampling is supposed to decrease the classification time, it is still a lossy process, which would affect accuracy. *What impact would the flow truncation process have on DPI accuracy and at which computational gain?* This is the question we are trying to answer in this work through our conducted experiments and the obtained results.

Our goal(s) through this work can be defined as follows:

1) To determine a minimum number of packets to be inspected within a p2p flow to be classified with an acceptable level of accuracy. To generalize for other protocols.

2) To discover to what extent per-flow sampling would optimize DPI classifiers in the sense of decreasing computational costs while maintaining an acceptable level of classification accuracy.

The remaining of this paper is organized as follows: Section II provides an overview of DPI optimization means and parameters for their evaluation, focusing on flow sampling techniques and putting our work in perspective with previous works in the literature. Section III describes the dataset we used for the experiments, and the OpenDpi [4] tool in the way it analyzes and labels packets and flows. Section IV provides a description of the way we used to truncate the flows. Section V shows our conducted experiments' results in terms of accuracy and computational cost and highlights on some important results and special

 ISBN: 978-1-61208-173-1

cases. Finally, Section VI presents the conclusion and future work.

II. STATE OF ART IN DPI OPTIMIZATION

DPI [2] based identification methods have no restrictions on inspecting the full contents of the payloads. As such, there are some concerns related to both user privacy and computational performance.

In this paper, we will focus on software based optimizations which concept is to reduce the size of DPI input through sampling techniques [5] that are considered as a general mean of input reduction. As many works [6] [7] [8] show, sampling techniques can be integrated within the traffic classification process.

Amongst the ways of categorizing sampling techniques we have the *per-packet* payload sampling [3] [9] [10], i.e., sampling bytes from within the packet payload, and the *per-flow* packet sampling [7] [8] [11] [12], i.e., sampling a subset of packets from within the whole traffic flow or a combination of both [13].

Per-flow packet sampling for DPI classification is shown in many papers with different sampling policies: Bloom filters in [8], Deep Packet Inspection using Parallel Bloom Filters [14] [15], k-ary sketch [16], Related sampling [12], and Mask-match sampling in [17]. Chen et al. [7] suggested six sampling strategies and showed how they affect DPI identification systems.

In [3], we tested *per-packet* payload sampling. In this paper, we consider *per-flow* packet sampling. However, we did not follow any specific sampling policy. Instead, given an accuracy level to be maintained, *we simply recommend to only inspect a predetermined number of packets N_{min} at the beginning of a traffic flow (p2p or other protocols).* Thus, the sampling rate will be *N_{min} packets per flow.*

Although an in-depth comparison of different sampling methods is beyond the scope of this paper, we will briefly compare the most relevant ones to our work. Our optimization concept is detailed in Subsection II.C.

Some of the sampling modes discussed in [7] can timely investigate the traffic load conditions of the links. However, their results were difficult to generalize since they were affected by many factors as they concluded.

Sampled NetFlow was mentioned in [11], where Carela-Español et al. find that packet sampling has a severe impact on the performance of the classification method. They were able to achieve an overall flow classification accuracy of 85% for a sampling rate of 1/100.

RelSamp (Related Sampling) [12] proposes that flows, parts of the same application session, are given higher probability. However, in [12] RelSamp was compared to Sampled NetFlow, which is a sampling technique for network monitoring rather than traffic classification.

Mask-match sampling method discussed in [17], provided 94% accuracy for UDP flows for a sampling rate of 0.1. However, this method focuses mainly on long flows, and the validity of the samples is related to the randomness of the ID field of the IP packets headers.

Similar to our work, Canini et al. [8] used Bloom filter to sample the first 10 packets of each flow while a negligible loss in the accuracy was encountered (0.0047%). However, they used L7 filter and they encountered some false positive figures due to Bloom Filters. Fernandes et al. [13] also proposed a similar work, where a combination of per-flow and per-packet sampling was used to capture only few packets (7 packets) per flow and a fraction of its payload without a significant impact on accuracy. However, the analysis in [13] did not provide protocol oriented results (such as p2p) and experiments were based on L7-filter tool.

III. OPTIMIZATION THROUGH FLOW TRUNCATION

Our goal in optimizing DPI is to maintain classification accuracy level as high as possible while trying to decrease the required computational cost.

In this subsection, we will focus on both the theoretical concept of optimizing the processing time through flow truncation, and on the quasi-theoretical estimation for this parameter as well.

Theoretically, the model of the DPI classifier proposed in [18] includes 5 processing blocks. Cascarano et al. studied the cost of each block and concluded that the most important is the cost of the pattern-matching block, which according to the study, has a linear dependence on the number of input characters.

For simplicity reasons, and according to the target of the experiments, we will model individual packet classification as a process composed of just two modules or steps:

- A packet-handling module, mainly devoted to preprocess the packet and identify the flow the packet belongs to (reading the packet, getting flow identifiers, searching for the flow in active flow list, etc.). This processing is mainly related to packet header.
- A packet-inspection module (referred to as "pattern matching block" in [18]), devoted to the classification of the packet. This module handles both the header and the payload of the packet.

Thus, the time cost related to individual packet classification t_{pc}, can be approximated as:

$$t_{pc}=t_h + t_i \quad (1)$$

where t_h is the packet handling time, and t_i is the packet inspection time.

In our concept of optimization through flow truncation, we emphasize on t_i as we consider it to be the only sensitive term to flow truncation. Through flow truncation, we intend only to classify packets which ordinal number inside the flow is lower than a predefined threshold (N_{min}) within each flow. However, this does not mean that packets over N_{min} are not be parsed at all. On the contrary, these packets still have to be handled by the classifier simply for determining to which flow they belong. The difference is that for these packets the inspection part is to be omitted. So, it is important to note that, for these packets, t_h cannot be avoided by the classifier, as it is evidently impossible to know if a given packet belongs to an existing flow without parsing its header at least. The decision to inspect the packet or not

 ISBN: 978-1-61208-173-1

depends on its ordinal number being lower or higher than N_{min}.

As a result, with flow truncation, we are supposed to eliminate packet inspection time t_i for packets which position in the flow is over N_{min}. This is supposed to decrease the global classification time for the whole traffic according to the following simple approximations:

- With *no flow truncation*, the time for analyzing a flow t_{fc} can be approximated as:

$$t_{fc} = N_p \cdot E[t_h + t_i] \quad (2)$$

where N_p is the number of packets in the flow and $E[t_h + t_i] = E[t_{pc}]$ is the average packet classification time. $E[t_{pc}]$ depends on the flow size and the length of the payloads, and on the protocol to which the flow belongs.

- With *flow truncation*, assuming $N_p > N_{min}$, the time for analyzing a flow, t'_{fc}, becomes:

$$t'_{fc} = N_p \cdot E[t_h] + N_{min} \cdot E[t_i] \quad (3)$$

Our proposal, based on N_{min}, and to the best of our knowledge, is different from the other sampling methods in the following points:

- Most sampling policies were general with no specific attention on the detection of a particular protocol (or class of protocols), while our aim is to find the number of packets N_{min} necessary for the detection of each particular protocol. This approach fits with our long-term objective, which is the detection of P2P flows.
- Most sampling policies provide sampling rates, which implies that the number of sampled packets will increase as long as the flow is under course while in our proposal, it is fixed to N_{min} *per flow*, which is an efficient method that yields higher computational gains for large flows.
- Most sampling policies neglect the effect of non sampled packets, while in our proposal we have to parse all packets.
- In some papers, experiments were restricted to DPI as implemented by L7-filter tool while in the latest classifiers, such as OpenDPI, the DPI technology is being enhanced through integration with other helper methods such as behavioral and statistical analysis.

To conclude with common features for all sampling methods, Guo et al. [6] showed that as the packet sampling probability decreases, the false negative rates become higher. Therefore, it can be stated that: the maximum value of sampling rate is limited by the affordable computational cost and the minimum value of sampling rate is limited by the acceptable accuracy value.

As our sampling rate is defined as: "*N_{min} packets per flow*", what is the recommended value for N_{min} to identify p2p and other protocols? The following sections will help in answering this question through experimental results.

IV. OPENDPI AND TESTBED

For the experiments, we used OpenDPI, which is derived from the commercial PACE product [19]. The core of OpenDPI is a software library designed to classify internet traffic according to application protocols. In its current version, up to 101 different protocols can be identified, including some P2P protocols. In addition to pattern matching, OpenDPI incorporates different techniques such as behavioral (by searching for known behavioral 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 behavioral analysis and the entropy of a flow). Our experimental setup is described next.

First, we use a customized tool based on the OpenDPI library, which is able to follow and differentiate the packets in each flow and to provide both flow and packet based outputs. Second, we used a dataset of real traffic captured at the access link of a medium size institution over 3 days. Complete flows in both directions were captured at a border router. We have chosen a subset of randomly selected files (totaling 3 GB) from our original dataset on which we run classification experiments. Then, by using the customized OpenDPI tool over the database subset and using complete flows with full packet payloads, we have built the "ground truth", i.e., the set of correctly labeled flows and packets (without truncation) that will be used as the reference for the analysis of flow truncation as described in the following section.

V. FLOW TRUNCATION RESULTS AND ANALYSIS

To be able to generate accuracy results without truncating flows, we customized OpenDPI to output the packet ordinal number inside the flow the packet belongs to at which detection is achieved. As described in OpenDPI documentation, the flow is classified according to the first recognized packet. In what follows, we will refer to this number as packet detection number or flow detection number. Then, to truncate flows, we have customized the code to be able to classify, within each flow, only packets with numbers less than N_{min}, and as mentioned in Section II, to solely handle remaining packets just for determining to which flow they belong. Note that with this customization, we could obtain computational measurements and validate accuracy results obtained previously through the flow detection number.

A. Accuracy Results

As shown in [1], a common framework is not yet defined for traffic classification methods. Therefore, we referred to existing works in the literature most of which commonly considered values above 90% as acceptable levels of classification accuracy. The distribution (histogram) of the flow detection number for both P2P and non-P2P protocols is depicted in Figure 1 for the full dataset, which contains 40340 P2P flows out of 4859208. As shown, there is a big number of flows for which the detection is achieved with just a few packets. Proportionally, the number of flows with high

 ISBN: 978-1-61208-173-1

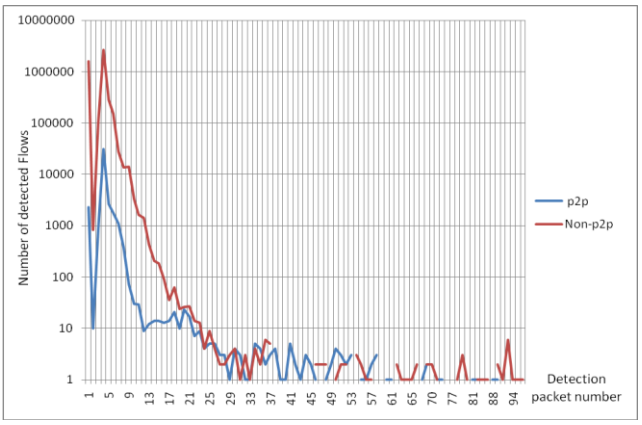

Figure 1. Histogram showing the number of detected p2p and non p2p flows in function of the flow detection number. Data for detection packet number over 100 is negligible and is not shown.

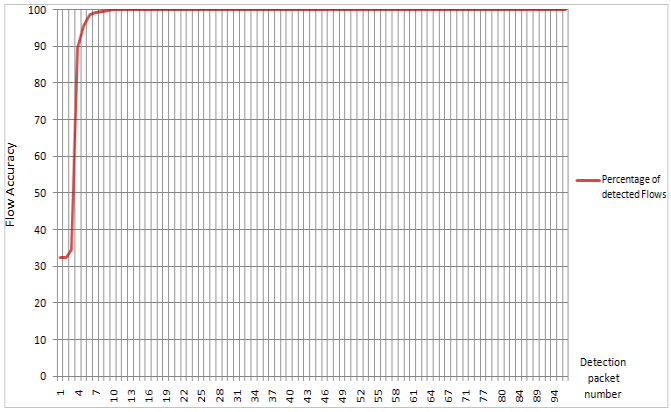

Figure 2. Global flow accuracy as a function of the packet detection number.

(i.e., greater than 50) flow detection number is almost negligible.

In fact, if we represent the percentage of flows that have been classified vs. the flow detection number (Figure 2), we can see that an important fraction of the flows can be classified with N_{min} below 20. Moreover, all protocols are mainly being detected within the first ten packets with a flow accuracy value of 99.90%.

As depicted also in Figure 2, for N_{min}=4, flow accuracy degrades to 84.35% while it jumps to 99.15%, for N_{min}=10. It is noticed that for N_{min} values greater than 10, only a very slight increase in accuracy is obtained. For example, for N_{min}=20, accuracy becomes 99.54%, with an increase of 0.39% compared to N_{min}=10.

Regarding the behavior on a per protocol basis, Figure 3 shows the average flow detection number for most common protocols in the dataset. Although some protocols like iMESH and Bitorrent show higher average values, most protocols averages were below 10 packets. If we further analyze the results on a per protocol basis, similar results could also be obtained. As an example, Figure 4 shows the histogram of the flow detection number for two relevant protocols: Bittorent (Figure 4.a) and http (Figure 4.b). Thus,

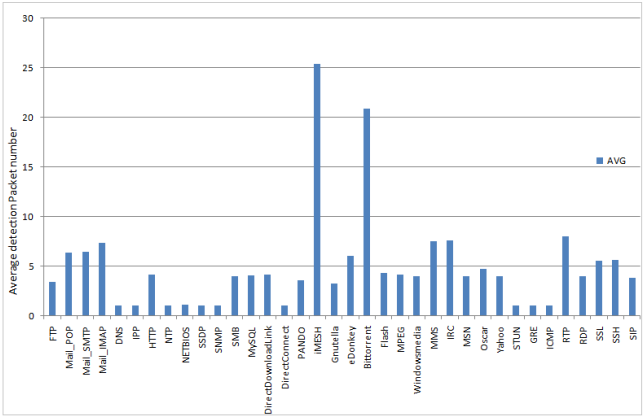

Figure 3. Average detection packet number for different protocols in the dataset.

an accuracy value of 99.91% for http and 99.18% for Bittorrent can be achieved if the 10 packets rule is still being respected for both protocols.

It is worthy to note that according to [19], OpenDPI in bandwidth management systems only scans for patterns in the first 1-3 packets for unencrypted and 3-20 packets for encrypted communication protocols: A fact which does not apply to OpenDPI in traffic classification systems, as shown in our experiments.

As it was clearly noticed from the previous results, in order to reach 99.15% of p2p flow accuracy, the classifier should inspect at least 10 packets. The same applies for remaining protocol. As a result, one optimal value to be recommended for N_{min} is 10. However, this is not mandatory as it is related to the required level of accuracy.

B. Computational Cost Results

Different set of evaluation tools and parameters were proposed for comparing classifiers, such as, Netramark [20] TIE [21] and perfprofiling for Snort [6]. However, according to [22], a commonly agreed upon workload for the evaluation of deep packet inspection architectures is still missing. In our case, we needed higher level evaluation parameters regardless of the pattern matching technique, therefore, we have chosen simple processing time as in [7] [18] using Linux monitoring tools [23]. For this purpose, we evoked the insertion into the classifier code at the proper places, of time related function calls providing granular results at the microseconds' level. We compiled the classifier code with GCC v4.4.3 with -O3 optimization level, on the testing server having the hardware specifications of 8 GB of memory, 2 Intel(R) Xeon(R) 2.66GHz processors with 4 cores each.

Tests were performed over one of the captured files for N_{min}=20. The most relevant features are shown in Table I, while Table II shows the results.

Flow and packet classification times were calculated by dividing the classification time for all the flows respectively by the number of flows for t_{fc} and by the number of packets for t_{pc}. Consequently, the gain in processing time when using truncated flows is 9.63%. .

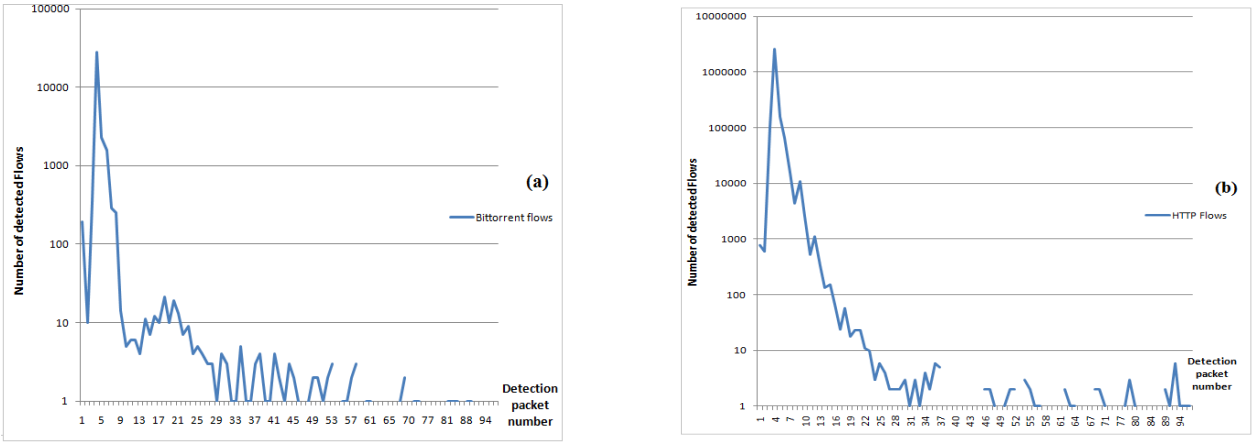

Figure 4. Histogram showing the number of detected flows in function of packet detection number for: (a) Bittorrent protocol (b) HTTP protocol.

As shown through the previous results, a *computational time gain* of 9 % can be obtained by classifying the first only 20 packets while maintaining an accuracy level of more than 99 %.

In Table I, the percentage of flows with N_p less than N_{min} is important for the effectiveness of the truncation. In fact, the smaller this value is, the bigger the time gain would be and the flow truncation will become more effective. In showing the importance of this parameter, we have chosen N_{min}=20 instead of N_{min}=10 as the accuracy is slightly different between these values, as depicted in Figure2. Then, we referred to comparing the quasi-theoretical and experimental measured values of the time gain. We measured the average for t_i=5.35µs, N_p=42, with N_{min}=20, then, when multiplied by the total number of flows, (3) gives the total processing time for all the flows. The obtained result is a quasi-theoretical value of 11.8 % for the time gain instead of 9.63%, which is the measured one. Effectively, this difference is due to the assumption we took that all of the flows should have theoretically an average number of packets (N_p=42) more than N_{min}=20, which is not exactly the case for the capture file we tested, having 30% flows with N_p less than N_{min} as shown in Table I.

C. Comparison with other sampling schemes

In comparing our approach (for N_{min}=20), with EIM (Equidistant Invariable Mode) [7] having a sampling rate of 7/13, 164084 packet samples are to be inspected, which means 36% increase in the classification time due inspecting additional 140,634 packets. Figure 1, shows that, for N_{min}=7, 97.96% of accuracy could be obtained, which applies to EIM scheme only if the first 7 packets were sampled, otherwise, EIM accuracy will drop to an unacceptable value, due to the stateful inspection of OpenDPI, as explained next.

TABLE I. CHARACTERISTICS OF THE CAPTURE FILE USED FOR TEST

Total # flows	**# P2P flows**	**# Flows $N_p < N_{min}$**	***Number of packets***	
			Under N_{min}	***Over N_{min}***
7337	188	30%	104678	200040

TABLE II. THE OBTAINED COMPUTATIONAL RESULTS

File N_{min}=20	**Classification time** (in µs)		
	All flows	***Flow (t_{fc})***	***Packet (t_{pc})***
Full flows	7300301	995	24
Truncated flows	6596881	899	21

D. Classifying the DPI Classifier

Although a concrete gain in classification time was obtained through flow truncation, still the 9.63% value did not meet with our expectations. In fact, if we assume that OpenDPI is inspecting all packets over N_{min} the gain has to be theoretically higher. However, this fact helped in reverse engineering an important aspect of the classifier itself, which in turn could interpret the moderate gain we obtained. In fact, OpenDPI seems to be incorporating DFI (Deep flow inspection) beside the DPI classification mechanism. Specifically, OpenDPI shows a stateful or PBFS like (Packet Based per Flow State) classification behavior. As per the taxonomy presented in [24], PBFS based DPI classifiers focus on the first packets of each session. Thus, they have a built-in feature of requiring less input than other classifiers.

In this context, the sampling scheme we proposed seems to be more convenient to PBFS classifiers as it focuses on packets where the PBFS classification decision is being made, whereas in most sampling methods, packets have to be continuously inspected as long as the flow is under course. In this regard, it may be required that all sampling works joint with classification should be reconsidered, especially when used with PBFS based classifiers.

E. Special cases

During experiments, some exceptional cases were noticed. Though these cases had no significant impact on the presented results, it is worthy to provide a preliminary interpretation while detailed explanations should be left for future work.

1) Deviators Flows

As shown in Figure 2, we calculated the average *detection packet number* for each protocol. However, we found some flows which detection number is much deviated

 ISBN: 978-1-61208-173-1

from the calculated average, we call these flows *deviators*. We validated the fact that the presence of most deviators is due to flows that were under course during the start of the capture. In addition, this highlights the importance of the first flow packets to OpenDPI, which when lost, will cause flows to be detected at packet numbers deviated from the average.

2) Packets Changing Protocol

Some minor flows were noticed to be changing their protocol even after the first time detection. This may lead to an error in classification, and can be considered as one of the weaknesses of our approach to be dealt with in future enhancements.

VI. CONCLUSION AND FUTURE WORK

This paper aims to optimize DPI-based classifier by decreasing the required computational cost while maintaining acceptable levels of flow classification accuracy for p2p and other application protocols. As opposed to many sampling works, our approach is to reduce input requirement through inspecting only a fixed number of packets from within the flow beginning. Our conducted experiments show that when inspecting the first 10 packets for all protocols, including p2p, more than 9% of classification time can be saved while a flow accuracy level over 99% can be still maintained. However, future enhancements may have to deal with some weaknesses and special cases detected within our approach such as, deviators flows, packets changing protocol within the same flow, studying the effect of modifying the first packets, distinguishing between packets in the uplink and downlink directions and their contribution in the classification process. Finally, it is important to note that when used jointly with stateful and PBFS based classifiers, sampling methods should accommodate to the importance of the first packets in classifying the whole flow. This fact highlights the importance of our sampling approach which accommodates samples to the stateful classifier's algorithm by focusing on the first packets, and takes into consideration as well, the characteristics of application protocols being classified by sampling a convenient number of 10 packets sufficient for identifying most application protocols including p2p. In this context, accommodating enhancement means to the classifier's algorithm from one hand, and to the classified traffic characteristics, on the other, would be a good practice for any related future work.

ACKNOWLEDGMENT

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

REFERENCES

[1] T. Nguyen and G. Armitage, " A Survey of Techniques for Internet Traffic Classification using Machine Learning", IEEE Communications Surveys & Tutorials, v. 10, pp. 56-76, 2007.

[2] Allot Communications "Digging Deeper Into Deep Packet Inspection (DPI)." White paper. Available at https://www.dpacket.org 14.09.2011

[3] J. Khalife, A. Hajjar, and J. Díaz-Verdejo, "Performance of Opendpi To Identify Truncated Network Traffic", In Proc. DCNET 2011, pp. 51-56, Seville.

[4] http://www.opendpi.org 14.09.2011

[5] R. Jurga and M. Hulbój, "Packet Sampling for Network Monitoring", Technical Report, CERN | HP Procurve openlab project. Available at http://www.zdnetasia.com 14.09.2011

[6] 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.

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

[8] M. Canini, D. Fay, D. Miller, A. Moore, and R. Bolla, "Per Flow Packet Sampling for High-Speed NetworkMonitoring", In Proc. COMSNETS'09, 2009, pp. 1-10.

[9] 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.

[10] 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.

[11] 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 .

[12] 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.

[13] 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.

[14] 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.

[15] Y. Li "Memory Efficient Parallel Bloom Filters for String Matching", In Proc. NSWCTC 2009, 2009, pp.485-488.

[16] 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.

[17] R. Cong, J. Yang and G. Cheng, "Research of Sampling Method Applied To Traffic Classification", In Proc. ICCT 2010, 2010, pp. 112-115.

[18] N. Cascarano, A. Este, F. Gringoli, F. Risso, and L. Salgarelli, "An Experimental Evaluation of the Computational Cost of a DPI Traffic Classifier", Proc. GLOBECOM'09, 2009, pp. 1-8.

[19] http://www.ipoque.com 14.09.2011

[20] S. Lee, H. Kim, D. Barman, S. Lee, C. Kim, and T. Kwon, "NeTraMark: A Network Traffic Classification Benchmark", ACM SIGCOMM Computer Communication Review, Volume 41 Issue 1, January 2011.

[21] http://www.grid.unina.it 14.09.2011

[22] M. Becchi, M. Franklin, and P. Crowley, "A Workload for Evaluating Deep Packet Inspection Architectures", In Proc. IISWC 2008, pp.79-89.

[23] http://www.tldp.org 14.09.2011

[24] 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.

 ISBN: 978-1-61208-173-1