Improving cost and accuracy of DPI traffic classifiers

Improving Cost and Accuracy of DPI Traffic Classifiers

Niccolò Cascarano
Politecnico di Torino
C.so Duca degli Abruzzi 24
Torino, Italy
niccolo.cascarano@polito.it

Luigi Ciminiera
Politecnico di Torino
C.so Duca degli Abruzzi 24
Torino, Italy
luigi.ciminiera@polito.it

Fulvio Risso
Politecnico di Torino
C.so Duca degli Abruzzi 24
Torino, Italy
fulvio.risso@polito.it

ABSTRACT

Traffic classification through Deep Packet Inspection (DPI) is considered extremely expensive in terms of processing costs, leading to the conclusion that this technique is not suitable for DPI analysis on high speed networks. However, we believe that performance can be improved by exploiting some common characteristics of the network traffic. In this paper we present and evaluate some optimizations that can definitely decrease the processing cost and can even improve the classification precision.

Categories and Subject Descriptors

C.2.3 [**Computer-communication networks**]: Network monitoring

Keywords

Deep Packet Inspection, Traffic Classification

1. INTRODUCTION

Deep packet inspection (DPI) is perhaps the most common technique for traffic classification. Among the reasons, its (relatively) straightforward hardware implementation and the fact that other techniques are still in the research domain. However, DPI classifiers are considered expensive in terms of CPU and memory consumption.

This mis-conception is probably due to two reasons. The first is that several flavors of DPI are possible and the most advanced techniques (e.g. the ones that reconstruct the entire stream at the application level) may be extremely costly. The second is that DPI is often associated to security applications such as Intrusion Detection Systems (IDS) or firewalls, which may include thousand of rules (the ruleset of the November 2007 release of Snort includes 8536 rules, 5549 of them requiring application-level content inspection [1]), which tend to stress the capabilities of DPI engines.

We believe that traffic classification is different from the other applications cited above because the number of protocols to be classified is definitely lower than the number of rules present in an IDS (tens or at most hundreds against several thousands) and because a limited number of misclassifications are acceptable. For instance, misclassified traffic may update the byte counter of the wrong protocol but it does not usually pose any security threat such as a misclassification into an IDS. Hence, a traffic classifier can take into consideration the average characteristics of the traffic in order to optimize the processing path while an IDS must prevent a single session from damaging network infrastructure, therefore leading to different architectural choices.

This paper describes some optimizations that are used by some real implementations and that are able to substantially reduce the cost of a DPI classifier. Optimizations are based on limiting the amount of data that are analyzed by the DPI engine, which is possible by taking into account some common characteristics of the traffic present in nowadays networks. However, no experimental data is available so far that evaluate the goodness of these techniques in terms of costs/benefits, i.e. at which extent the gain in processing complexity can be balanced with a loss of precision.

This paper aims at filling this gap by evaluating the impact of each optimization on the processing load and on the accuracy of the classification process. Results demonstrate that these optimizations alone are able to decrease the processing cost of more than one order of magnitude, while even improving classification precision. Our analysis is based on real traffic traces, some of them containing also a relevant portion of peer-to-peer traffic that is known to be challenging for DPI classifiers because of its frequent use of encryption and content hiding techniques to avoid the identification. Moreover, two of the three traffic traces used have been collected using the GT suite [2], which provides the ground truth for evaluating the classification accuracy.

This paper is organized as follows. Section 2 presents the existing works on DPI optimizations, while Section 3 introduces the general architecture of a DPI classifier. Proposed optimizations are presented in Section 4 and the procedures used in their evaluation are summarized in Section 5. Section 6 evaluates the impact of the optimizations in terms of processing load and classification precision and Section 7 concludes the paper.

2. RELATED WORKS

Most papers assume that that DPI is a very expensive traffic classification technique without any further investigation. A (non exhaustive) list can be papers [3–6]. Only recently a paper appeared [7] that demonstrates how the

SAC'10 March 22-26, 2010, Sierre, Switzerland.

complexity of a well-known traffic classifier based on Support Virtual Machine may be comparable to the DPI one.

So far, most of the work focused on the analysis or the definition of new techniques for performing fast and scalable regular expression matching [8–10], which are mostly appropriate when thousands of rules are present or when the characteristics of the ruleset in use require a careful representation in memory in order to avoid state explosions [1], which are common conditions especially in case of IDS.

However, we believe that traffic classification has different characteristics than the applications that operate in network security. In our vision, the complexity of the DPI technique applied to traffic classification can be definitely decreased by accepting a minor worsening in the classification accuracy and by optimizing the whole process of DPI classification and not just the engine in charge of the regular expression matching.

Along this line, Moore et al. [11] proposes different algorithms that basically increase the amount of data to be analyzed until a positive match on a known protocol is returned, but it limits the analysis on the classification precision and does not investigate the impact in terms of processing cost. Furthermore, they use old traffic traces that do not contain peer-to-peer traffic.

A (relatively) widespread technique to reduce the processing cost of traffic classification consists in using only the first portion of the session data for classification. For instance, BRO IDS stops the pattern matching after examining 1KBytes of application payload, while Snort can specify (for each signature) the amount of application data to be analyzed before stopping the search. However, this value has been chosen as "reasonable" according to the users experience. Also in this case no scientific analysis has been performed to justify this choice and to evaluate its impact on classification precision and computational load.

Another technique which seems to be in use in some commercial device consists in stopping the analysis of a session if it is still unclassified after a given number of packets. Also in this case no public results are available that can confirm the goodness of this technique.

3. BASELINE DPI CLASSIFIER

A DPI classifier relies on the observation that each application uses specific *protocol headers* to initiate and/or control the information transfer. DPI uses mostly regular expressions (commonly called *signatures*) to identify the peculiar sequence of data of an application protocol. A DPI engine receives all the packets that belong to a TCP/IP session[1] not yet associated to a known application-level protocol and it compares the packet payload to a given set of signatures. In case a match is found, the corresponding TCP/IP session is placed in a session table and all the packets belonging to that session are no further analyzed by the DPI engine.

Among the several flavours of DPI classifiers, the most common categories [12] are *(i) packet-based, per-flow state* which analyzes data on a packet-by-packet basis as soon as packets are received by the classifier, and *(ii) message-based, per-flow state* that analyzes application-level payload as a unique stream of data, after TCP/IP normalization[2].

Our "baseline DPI classifier" follows the packet-based approach that, as pointed out in [11, 12], is usually enough for traffic classification. In fact, the loss of precision of the packet-based approach is usually limited and tolerable in case of traffic classification, while the additional precision of the message-based approach makes it more appropriate for network security environments, which are outside the scope of this paper. We use the signatures contained in the protocol database available on NetPDL website [13], which includes some processing tools that can be easily customized for our objectives. The current version of the NetPDL database (as of July 2009) includes 72 application-level protocols (39 TCP, 25 UDP and 8 that operate with both TCP and UDP), whose signature are partially derived from the L7-filter [14] project.

4. DPI OPTIMIZATIONS

This Section presents the optimizations that can improve the behavior of the DPI traffic classifier under evaluation.

4.1 Snapshot-based classification

An inspection of the protocol signatures contained in our database [13] shows that most of them require a limited number of bytes for identifying a protocol, and these bytes are usually placed at the beginning of the application data. This makes sense because application signatures usually describe the handshaking phase of application protocols or some kind of data used for synchronization, which are usually placed at the beginning of the application payload; the rest is usually application-related data that is useless for classification. This applies particularly to UDP traffic, whose datagrams travel independently on the network and therefore each packet requires its own header (e.g. for sequence numbering). For the above considerations, we can reasonably conclude that the portion of data useful for classification should be in the first bytes transmitted over the wire.

Although in principle we should stay with the longest input (i.e. full payload) in order to preserve the accuracy, a better approach is to limit the number of bytes provided to the pattern search block and to evaluate the possible loss in classification accuracy, in order to determine the best tradeoff between accuracy and processing cost. We speculate that forcing the DPI classifier to analyze only a snapshot of network packets can provide an advantage in terms of cost, without a significant impact on the classification accuracy.

4.2 Limiting classification attempts

A DPI classifier might waste a huge amount of computational resources while performing pattern searches on sessions that cannot be identified anyway. Table 3 demonstrates that traces with higher percentage of unknown traffic are associated with higher computational costs, suggesting a direct correlation between unclassified traffic and cost. For instance, a protocol cannot be classified when *(i)* the data transported by the session is encrypted, *(ii)* the classifier does not know the signature that describe the protocol transported, or *(iii)* the signature does not match because it is split across two packets.

[1]A TCP/IP session includes all the network packets that share the same source/destination address, transport protocol and source/destination ports.

[2]TCP/IP normalization is a term commonly used to indicate the capability to handle IP fragments and TCP segment reordering/duplication, which can be used to rebuild the original application-level stream as seen by the receiver.

Table 1: Average number of pattern searches in case of unclassified sessions

Data set	$\bar{x}$	$\sigma(x)$
UNIBS-GT (tcp)	654	4619
POLITO-GT (tcp)	563	3659
POLITO (tcp)	67.6	1879
UNIBS-GT (udp)	2.62	0.71
POLITO-GT (udp)	6.05	26.4
POLITO (udp)	9.17	476

Apart from the case (iii) that is outside the scope of this paper (a packet-based approach is currently considered), in the other two cases the DPI classifier should stop its analysis over a given session after a reasonable number of classification attempts and leave the session as "unclassified". This could save processing power since we avoid to analyze all the packets belonging to the session, while at the same time we decrease the possibility that a random payload matches a signature and brings to a misclassification. Particularly, Table 1 shows the mean and standard deviation of the number of classification attempts executed over unclassified sessions; the number of searches is extremely high especially in case of TCP sessions, which are usually longer than UDP ones.

This suggests that a reasonable limit on the number of classification attempts per session should not affect the amount of traffic correctly classified, while it should reduce the number of misclassifications and the processing cost per packet.

5. EVALUATION METHODOLOGY

5.1 Parameters under evaluation

We define three parameters for the evaluation of each optimization: *(i)* the performance speedup of the new classifier compared to the baseline classifier presented in Section 3 (measured when processing the same traffic traces) and the accuracy in terms of *(ii)* percentage of unclassified and *(iii)* misclassified traffic. The "unclassified traffic" is the traffic that does not match any known protocol signature. The "misclassified traffic" is the traffic that matches a protocol signature that does not correspond to the application that generated it. Obviously, the last parameter can be calculated only on traffic traces captured with the GT suite. The traffic correctly classified can be derived by complementing the unclassified and misclassified traffic and we tolerated a maximum worsening of 1% compared to the traffic correctly classified without optimizations.

Processing cost has been evaluated by running the classifier on our traces and measuring the average processing cost per packet. Measurements have been done using the RDTSC assembly instruction available on Pentium-compatible processors and include only the time spent in the DPI classifier, excluding all the other conpanion functions (e.g. loading packets from disk). The measurement platform was an Intel Dual Xeon 5160 at 3GHz, 4GB RAM and Ubuntu 8.04 32bit; the code under examination was compiled with GCC v4.2.4 and always executed on the same CPU core.

5.2 Traffic traces

Table 2 summarizes the most important characteristics of the three full-payload traffic traces that we used to evaluate the goodness of the proposed optimizations. All of them were collected through the well-known *tcpdump* tool at the border routers of Politecnico di Torino and University of Brescia campuses and then properly anonymized. The "POLITO-GT" and "UNIBS-GT" traces were obtained using the GT suite [2] thus we know exactly which application generated each session; this information allows to evaluate the absolute accuracy of the classification process.

Table 2: Traffic traces used in the evaluation

Data set	Date and Duration	Bytes	Packets
POLITO-GT	December 10, 2008 68 hours	202 GB 76.8% TCP	330M 69.8% TCP
UNIBS-GT	December 17, 2008 56 hours	3.5 GB 99.4% TCP	4.72M 67.6% TCP
POLITO	Dec. 20, 2007 12 hours	419 GB 94.7 % TCP	579M 92.3% TCP

Table 3: Classification cost and precision of the baseline classifier

Data set	Avg. cost (ticks/pkt)	Unknown (bytes)	Misclass. (bytes)
POLITO-GT (tcp)	6681	72.7%	16.9%
UNIBS-GT (tcp)	2503	29.2%	7.73%
POLITO (tcp)	743	5.67%	N/A
POLITO-GT (udp)	242	0.43%	57.7%
UNIBS-GT (udp)	758	14.7%	1.39%
POLITO (udp)	709	15.3%	N/A

POLITO-GT contains mainly peer-to-peer and WebTV traffic, generated by 10 virtual machines running Edonkey, Bittorrent, Skype, PPlive, TVAnts and SopCast on Windows XP, plus four real machines running Linux, Windows Vista and MacOS X with default settings and used by regular users. WebTV applications were executed with an automatic turnover of 1 hour, while P2P application were downloading and seeding some popular resources for the entire duration of the capture. The UNIBS-GT trace contains the traffic generated in a research laboratory by 20 PhD students doing their normal activities. This trace is smaller than POLITO-GT in term of volume but it contains more normal users activity, including P2P file sharing.

Since POLITO-GT and UNIBS-GT include traffic generated by a limited number of hosts due to the difficulties to deploy the GT suite over many clients, we decided to use also the POLITO trace that includes traffic generated by about 6000 hosts during an entire working day in order to extend the evaluation scenario. Although the ground truth is not available on this trace (hence the amount of misclassified traffic cannot be derived and the result in terms of unclassified traffic cannot be verified), this trace is interesting for evaluating the impact of proposed optimizations at least in terms of processing costs in a more realistic scenario.

5.3 Performance of the baseline classifier

Table 3 reports the classification results (in terms of average processing cost per packet and unclassified / misclassified traffic) obtained on the three traffic traces by the baseline DPI classifier. POLITO-GT trace has a very high percentage of unknown TCP traffic. This is expected because the signatures for the WebTV protocols, which represent the largest part of the traffic captured, are partially unknown and partially derived with reverse engineering (and not very precise). With respect to the UDP portion, the result is

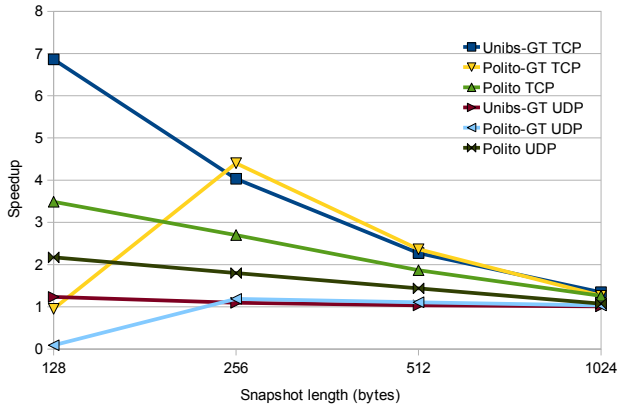

Figure 1: Performance speedup at different snapshot length.

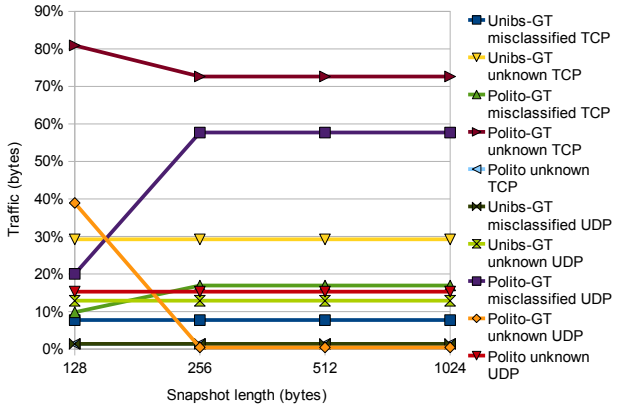

Figure 2: Classification accuracy at different snapshot length (thick lines refer to TCP).

even more problematic because of the high percentage of misclassified traffic. Trace UNIBS-GT is less critical than POLITO-GT since the percentage of misclassified traffic is reasonably low; we still have a large portion of unknown traffic due to the use of P2P file sharing applications. For trace POLITO we have only the information of unknown traffic that results to be lower than the other two traces for the TCP case (most hosts on the network use only "standard" applications such as web and email). Vice versa, the unknown UDP traffic is rather high, probably for the presence of traffic coming from P2P clients that adopt hiding techniques for avoiding classification (e.g. Emule).

The average processing cost for TCP and UDP is different because we implemented the pattern matching module as a Deterministic Finite Automata (DFA), using algorithm provided by the well-known `flex` tool. The cost of a DFA matching depends on many parameters (i.e., average packet size, average session length, presence of unclassifiable sessions, type of regular expressions used, etc.), which are different from trace to trace and even from TCP and UDP traffic.

While a DFA does not seem appropriate for traffic classification, [1] demonstrated that a careful implementation is able to address the requirements of many regular expressions even with the Kleene closure (which represents the worst operating conditions for DFA because it could lead to state explosion).

6. EXPERIMENTAL EVALUATION

6.1 Snapshot-based classification

We evaluated the impact of the snapshot length optimization by varying the number of bytes analyzed for each packet and by observing the corresponding variation in accuracy and processing cost. Results, shown in Figure 1, are encouraging. For instance, the best trace (UNIBS-GT) shows that the processing cost related to the TCP traffic is 7 times better than the one of the baseline classifier.

The POLITO-GT trace is a tough trace for our DPI classifier because of the amount of unclassified traffic present (72.7% of the TCP traffic, as reported in Table 3). For this trace we can observe an interesting phenomenon: processing cost obtained with a 128 bytes snapshot is higher than the one related to the 256 bytes case, and a similar phenomenon can be seen for the classification accuracy. In fact, Figure 2 shows that the 128 bytes snapshot leads to an higher amount of unknown traffic while the misclassified traffic is lower. The reason is that a smaller snapshot triggers less false positive matches on that trace, but this implies the growth of the number of classification attempts per session, lowering the classification speedup consequently. Another observation reveals that UDP traffic shows a limited improvement in terms of processing cost compared to the TCP case; this is mainly due to the average payload size of UDP traffic that is usually smaller than TCP and often below 250 bytes.

Figure 2 shows the corresponding variation in classification accuracy. Results are in some sense similar to the previous graph, showing that in most cases even a snapshot length of 128 bytes does not affect significantly the accuracy, and that the result achieved with a snapshot of 256 bytes is almost indistinguishable from the result obtained with full payload. The worst trace in these conditions appears to be the POLITO, which shows an imperceptible increase in the unknown TCP traffic (0.04% in bytes) with respect to the baseline classifier. With respect to UDP, results are even more interesting. In the POLITO-GT trace there are no differences at all between 256 bytes and the full payload, while the POLITO trace the difference is limited to 0.001% in terms of bytes. Interesting, the traffic classified differently includes also some sessions that are misclassified with full payload and that remain unclassified with the snapshot.

According to these results, an hard limit of 256 bytes seems to be a good tradeoff between the improvements in processing costs (especially for TCP traffic, which varies between 2.8 and 4.4 against the baseline classifier), and the impact on accuracy. For UDP, the improvement is smaller and varies (in our traces) between 1.1 and 1.8 but there are almost no effects in terms of accuracy. Although this number depends on the traffic traces, we believe our dataset is sufficiently representative of real network traffic and therefore we do not expect may differences in other environments. Further analysis on different dataset are planned, but this is an hard task due to the difficulties of getting traces that include the real payload (i.e. not randomized data) and the proper ground truth associated with each session.

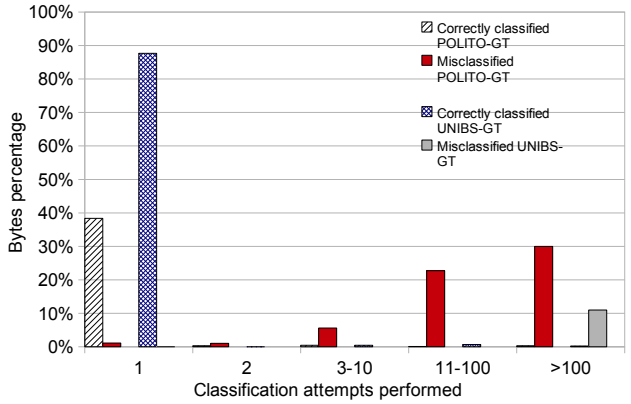

Figure 3: Distribution of positive matches vs. number of classification attempts performed.

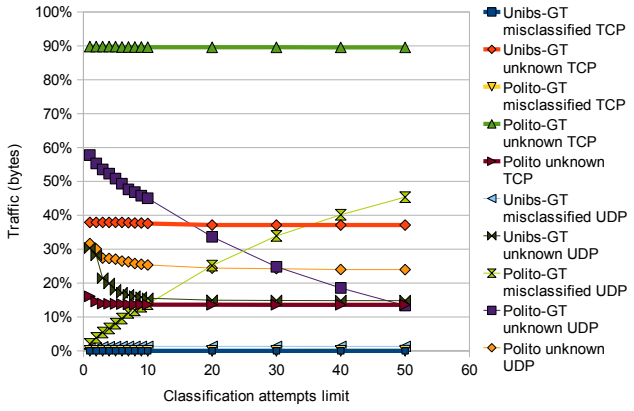

Figure 4: Classification accuracy at different classification attempts limit (thick lines refer to TCP).

6.2 Limiting classification attempts

Figure 3 shows the percentage of traffic that is either correctly classified or misclassified by the baseline classifier while examining the first N packets of each session. The most part of the traffic is correctly classified by examining only the first packet, while the sessions that are classified when examining the N^{th} packet (with $N \geq 2$) is definitely limited. Besides, inspecting more packets has the side effect of increasing the amount of misclassifications, because the randomness of application data transported leads to incidentally return a positive match on some "weak" signatures.

Figure 4 adds more details and shows that TCP traffic is classified almost entirely at the first packet, in both UNIBS-GT and POLITO-GT traces (the curves of unclassified and misclassified traffic does not change sensibly with $N \leq 50$). Considering a limit of $N = 2$, the correctly classified traffic is reduced of 0.95% in the UNIBS-GT trace without almost no misclassifications, which can account up to 7.73% in absence of limits. Results on the POLITO-GT are even better, with a loss of 0.20% in terms of correct traffic and almost no misclassifications.

Unfortunately, the analysis of UDP traffic is less encouraging. While in the POLITO-GT trace the correctly classified traffic drops only 0.57% when using $N = 2$ instead of $N = \infty$, the UNIBS-GT shows a decrease of 12.87% in the same conditions. The gap reduces when inspecting more packets (i.e. it becomes 0.76% when $N = 10$) but this seems to suggest that the definition of a "fair" limit such as in the TCP case is more complicated. With respect to the misclassified traffic, both traces share the same behavior. In the POLITO-GT trace, in which this phenomenon is more evident, the unclassified traffic drops from 55.29% ($N = 2$) to 13.37% ($N = 50$) and leads almost entirely to misclassifications, whose value grows from 3.84% ($N = 2$) to 45.35% ($N = 50$). This phenomenon is less evident in the UNIBS-GT trace, but it is expected because the misclassified traffic in this trace is already very low.

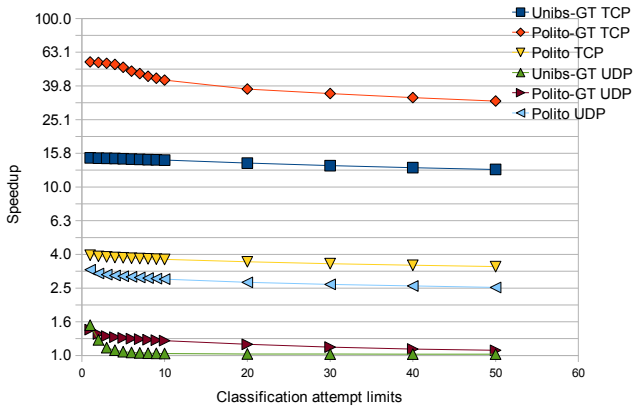

Figure 5: Performance speedup at different classification attempts limits (Y scale is logarithmic).

The reduction of misclassified traffic is a clear by-product of this technique and it is particularly evident on traces with high amount of encrypted traffic or P2P applications using obfuscation techniques. For instance, Table 3 reports a 72.7% of unknown TCP traffic for POLITO-GT trace, which becomes almost 90% in Figure 4 because it includes also all the previously misclassified traffic (was 16.9% in Table 3) that remains unclassified with the new optimization.

Clearly, this optimization can lead to a gain in term of processing cost. Figure 5 shows the performance speedup obtained with different limits of classification attempts, which can reach more than a 50-fold increase on some traces (e.g. POLITO-GT with $N = 1$). The gain is less evident for UDP traffic, mostly due to the fact that UDP sessions are usually shorter (e.g. several DNS queries are present in the traces) and some "weak" signatures may return a positive result after a small number of packets, which are in most part misclassifications. For instance, Table 1 shows that the average number of packets required to classify a UDP session is always less than 10, although the standard deviation may be rather high (POLITO trace) showing that a non-negligible percentage of long UDP sessions may exist.

A limit of $N = 2$ for TCP traffic seems acceptable in terms of classification accuracy[3], with a corresponding improvement of the processing cost that ranges from about 4 to more than 50 times, depending on the traces (3.9, 14.8, 54.8

[3]This value is probably due to the fact that some signatures match over the message coming from the server, which is usually the *second* packet of the session.

respectively for POLITO, UNIBS-GT and POLITO-GT). For UDP traffic, results are less interesting. For instance, the speedup in terms of processing cost is limited (from 1.2 to 2.9 times) even with $N = 2$, at the expense of a substantial loss in classification accuracy (e.g., in the UNIBS-GT trace). On the other side, the improvement with higher values of N becomes negligible, e.g. the cost speedup of the UNIBS-GT trace when $N = 10$ is limited to 1.03.

We can conclude that limiting the number of classification attempts of a DPI classifier is a good strategy for limiting the computational cost introduced by unclassifiable traffic while preserving, and in some cases improving, the classification accuracy in case of TCP traffic, and a limit of $N = 2$ seems to be a good tradeoff. With respect to the UDP traffic, this optimization does not seem to guarantee sensible improvements in terms of processing costs even in case of very small values of N; furthermore the impact on the classification accuracy is unclear because the misclassifications are definitely reduced, but the amount of correctly classified traffic may suffer. This may be due to the poor quality of the signatures we use, but this point will surely require some further investigations.

7. CONCLUSIONS

This paper presents two optimizations of a packet-based DPI traffic classifier that are based on reducing the amount of data fed to the pattern matching engine. These optimizations are possible because we focus on traffic classification in which we can accept a limited reduction on the accuracy, while network security (e.g., IDS, firewalls) is out of scope. For each optimization we analyzed the trade-off between the improvement in terms of processing cost and the worsening in terms of classification accuracy.

According to our analysis, the first optimization based on the snapshot limit is particularly appropriate for UDP traffic, which shows no differences in the accuracy when reducing the payload analysis to the first 256 bytes, with a fair decrease in processing complexity. The second optimization based on limiting the number of packets examined is particularly useful for TCP traffic, in which a speedup of more than one order of magnitude and a negligible loss in terms of accuracy can be achieved when examining at most the first two packets of each session.

The second optimization has an interesting side effect in that it can reduce considerably the amount of misclassifications because "late" classifications appears mostly false positives. In that sense, this technique is beneficial also for UDP traffic, but we were unable to identify a clear threshold; first results seem to suggest that 10 packets may be a reasonable limit, but further investigations are required on this topic.

Although we worked with a limited number of traces, we are reasonably confident that our datasets are sufficiently representative of the current network traffic. Further investigations with different data sets are planned, but we are experiencing many difficulties in getting full-payload traces with ground truth information over a large number of clients.

8. ACKNOWLEDGEMENT

We would like to thank Luca Salgarelli and Francesco Gringoli at University of Brescia who gave us many suggestions in the earlier part of this work and who contributed to the evaluation of the results presented in this paper.

9. REFERENCES

[1] M. Becchi, M. A. Franklin, and P. Crowley, "A workload for evaluating deep packet inspection architectures," in *Proceedings of the IEEE International Symposium on Workload Characterization*, pp. 79–89, IEEE, Sept. 2008.

[2] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. Claffy, "Gt: picking up the truth from the ground for internet traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 207–218, October 2009.

[3] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," in *Proceedings of the 16th International Conference on World Wide Web*, (New York, NY, USA), pp. 883–892, ACM, 2007.

[4] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, 2006.

[5] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 5–16, 2007.

[6] S. Zander, T. T. T. Nguyen, and G. J. Armitage, "Self-learning ip traffic classification based on statistical flow characteristics.," in *Passive and Active Measurement Workshop*, vol. 3431 of *Lecture Notes in Computer Science*, pp. 325–328, Springer, 2005.

[7] N. Cascarano, A. Este, F. Gringoli, F. Risso, and L. Salgarelli, "An experimental evaluation of the computational cost of a dpi traffic classifier," in *Proceedings of IEEE Globecom 2009, Next-Generation Networking and Internet Symposium*, (New York, NY, USA), pp. 50–59, IEEE, November 2009.

[8] R. Smith, C. Estan, S. Jha, and S. Kong, "Deflating the big bang: fast and scalable deep packet inspection with extended finite automata," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 207–218, 2008.

[9] M. Becchi and P. Crowley, "Efficient regular expression evaluation: theory to practice," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '08)*, (New York, NY, USA), pp. 50–59, ACM, 2008.

[10] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," in *SIGCOMM 2006*, (New York, NY, USA), pp. 339–350, ACM, 2006.

[11] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proceedings of the Passive and Active Measurements Workshop*, pp. 41–54, 2005.

[12] F. Risso, M. Baldi, O. Morandi, A. Baldini, and P. Monclus, "Lightweight, payload-based traffic classification: An experimental evaluation," in *IEEE International Conference on, International Conference on Communications (ICC)*, pp. 5869–5875, May 2008.

[13] Netgroup Research Group, "NetBee library Protocol description database," `http://www.nbee.org/netpdl`.

[14] l7-filter, "Application Layer Packet Classifier for Linux," `http://l7-filter.sourceforge.net/`.