DPI Solutions in Practice: Benchmark and Comparison

DPI Solutions in Practice: Benchmark and Comparison

Tommaso Rescio, Thomas Favale, Francesca Soro, Marco Mellia
Politecnico di Torino, Italy
name.surname@polito.it

Idilio Drago
University of Turin, Italy
idilio.drago@unito.it

***Abstract*—Having a clear insight on the protocols carrying traffic is crucial for network applications. Deep Packet Inspection (DPI) has been a key technique to provide visibility into traffic. DPI has proven effective in various scenarios, and indeed several open source DPI solutions are maintained by the community. Yet, these solutions provide different classifications, and it is hard to establish a common ground truth. Independent works approaching the question of the quality of DPI are already aged and rely on limited datasets. Here, we test if open source DPI solutions can provide useful information in practical scenarios, e.g., supporting security applications. We provide an evaluation of the performance of four open-source DPI solutions, namely nDPI, Libprotoident, Tstat and Zeek. We use datasets covering various traffic scenarios, including operational networks, IoT scenarios and malware. As no ground truth is available, we study the consistency of classification across the solutions, investigating root-causes of conflicts. Important for on-line security applications, we check whether DPI solutions provide reliable classification with a limited number of packets per flow. All in all, we confirm that DPI solutions still perform satisfactorily for well-known protocols. They however struggle with some P2P traffic and security scenarios (e.g., with malware traffic). All tested solutions reach a final classification after observing few packets with payload, showing adequacy for on-line applications.**

***Index Terms*—DPI, Protocol Recognition, Traffic Analysis**

I. Introduction

The internet is a continuously growing ecosystem composed by diverse protocols and applications. The rise and spread of smart devices, video-conference platforms as well as the continuous appearance of sophisticate cyber-attacks keeps changing the characteristics of traffic observed in the network. Understanding protocols that are carrying specific flows in the middle of such a variety of traffic has always been essential for multiple applications, in particular for those supporting network security like firewalls and IDS.

Deep Packet Inspection (DPI in short) has been the dominant approach to perform protocol recognition, showing effectiveness in several traffic monitoring scenarios. DPI parses traffic payload searching for signatures that characterize the protocols. Indeed, many DPI solutions do exist and still find important applications, despite the increasing usage of encrypted protocols. DPI is particularly useful in cyber-security scenarios, such as for intrusion detection systems, firewalls and other tools supporting security (e.g., flexible honeypots). The timely identification of a broad range of protocols remains a key first step in the security use case, calling for accurate, efficient and up-to-date DPI solutions. Yet, previous efforts providing an independent evaluation of DPI are already aged [1] or leverage on restrict traffic traces, which questions the applicability of such results to practical scenarios.

We revisit the question on the quality of DPI-based protocol identification. We select and evaluate four popular, open source projects implementing DPI, namely nDPI [2], Libprotoident [3], Tstat [4] and Zeek [5]. We first study their classification using passively captured traces, covering a wide range of scenarios, i.e., traffic produced by IoT devices, collaborative platforms/video-calls, malware, as well as production internet traffic. Establishing a ground-truth is challenging when dealing with such diverse traces composed by dozens of protocols. We here evaluate the *consistency* of the classification provided by the tools, relying on heuristics and domain knowledge to validate the decision of each tool when finding conflicting cases.

After that, we investigate whether the DPI solutions operate consistently when exposed to a limited number of packets per flow. Indeed, network applications usually perform protocol identification on-the-fly using the initial packets of each flow, in order to take timely decisions. For this, we investigate the number of packets per flow each solution needs to reach a decision, as well as the consistency of such decisions as more traffic is observed.

Our results show that:

- All tested solutions perform well when facing traces with well-established protocols. This is particularly true for popular protocols that account for the majority of production traffic;
- Some DPI solutions struggle when facing unusual events, such as massive scans or malware traffic;
- All tested tools reach a final decision already after observing the first packets with payload in a flow;
- nDPI outputs labels more often than others, and it usually agrees with the majority when tools diverge about the protocol of a flow.

To foster further research and contribute to the community, we share our code and the instructions to build the complete datasets used in our experiments.[1]

The research leading to these results has been funded by the Huawei R&D Center (France) and the SmartData@PoliTO center for Big Data technologies.

[1] https://smartdata.polito.it/dpi-in-practice/

Next, Sect. II summarizes the related work. Sect. III introduces our datasets and methodology. Sect. IV describes the results, and finally Sect. V concludes the paper.

II. Related Work

DPI has been applied to protocol identification since the early 2000s, when the usage of well-known ports for traffic identification turned out to be unreliable. Multiple approaches have been proposed. Some works rely on "shallow" packet inspection [6], i.e., they parse only packet headers in the search for protocol fingerprints. Such techniques still find practical applications, as encryption protects protocol payloads. Others propose efficient approaches for DPI, e.g., using pattern matching [7] or finely-tailored DPI algorithms [8]. Finally, some works rely on stateful information from multiple flows to label traffic, e.g., leveraging the DNS to obtain the labels used to classify encrypted traffic [4].

Many DPI tools have been introduced implementing such techniques. Here we consider four alternatives, which have been evaluated by original authors in [2], [3], [5], [9]. In contrast to them, we perform an independent evaluation of the tools, thus providing also a validation of the authors' results.

Past works compare DPI solutions. Authors of [10] perform an extensive benchmark covering port-based classification, packet signature algorithms etc. In [11], authors survey approaches to overcome the lack of ground truth in such studies. In some cases manual labelling of packet captures is used for DPI comparisons [12], while other works rely on active measurements to enrich captures with information about underneath applications [13]–[15].

Closer to our analysis is the work presented in [1], where authors also provide an independent comparison of DPI solutions. In contrast to [1], we leave out of our evaluation proprietary tools and libraries, since the lack of source code makes it hard to explore and explain discrepant results. We also refrain from evaluating tools no longer maintained. More important, we provide an updated comparison of DPI tools considering recent and real traces, thus covering scenarios not evaluated in the previous work, with a particular focus on modern security applications.

III. Datasets and Methodology

Fig. 1 summarizes our methodology. We describe the DPI tools selected for testing (Sect. III-A). Then, we build up a set of traces covering different traffic scenarios (Sect. III-B). Next we process the traces with the DPI tools. As matching the obtained labels requires ingenuity, we perform several steps and build up heuristics to find discrepancies on the final classifications (Sect. III-C).

A. Selection of DPI Tools

We restrict our analysis to DPI tools that perform *protocol* identification (e.g., HTTP, TLS, SSH etc.), ignoring those aiming at the identification of the services generating traffic (e.g., Google, Facebook etc.) [16], [17]. Namely, we focus on the following four alternatives:

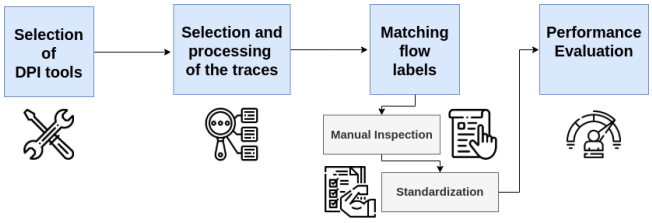


Fig. 1. Testing methodology.

- *nDPI* [2] is an open-source DPI library written in *C* and based on dissectors, i.e., functions that detect the given protocols. It is an *OpenDPI* [18] fork optimized for performance and supports more than 100 protocols.
- *Libprotoident* [3] is a *C++* library that focuses on L7 protocols. It applies a lightweight approach that uses just the first 4 bytes of payload. The idea is to overcome drawbacks of DPI, i.e., computational complexity and privacy risks. The library combines pattern matching with algorithms based on payload sizes, port numbers and IP matching. It supports over 200 protocols.
- *Zeek*[2] – formerly *Bro* [5] – is a complete framework for traffic analysis that also allows L7 protocol recognition. It exploits a combination of protocol fingerprint matching and *protocol analyzers*. It currently supports more than 70 protocols.
- *Tstat* [4] is a passive traffic monitoring tool that classifies traffic flows. It identifies a set of L7 protocols using payload fingerprint matching. It supports over 40 protocols.

Recall that we ignore projects no longer active. In particular, we leave *L7-filter* out since it has been shown to produce unreliable results in more recent scenarios [12]. Equally, we ignore proprietary alternatives, given the intrinsic difficulty to evaluate the root-causes of conflicting results without access to source codes [2]. Finally, we do not evaluate *tshark*[3] as it has proved much slower than the alternatives.

B. Selection and pre-processing of traces

We consider four scenarios to compare the DPI alternatives, including not only common internet protocols, but also protocols encountered by security applications.

We select 421 different PCAP traces that are aggregated in four macro-categories: (i) *User*, which includes ordinary browsing activity of ISP users while at home; (ii) *Media & Games* [19]–[21] that includes conference-calls, RTC applications, multimedia and gaming traffic; (iii) *Malware* [22], which aggregates several samples of malware[4] and security experiments;[5] and *IoT* [23], [24], captured in different labs hosting a variety of IoT devices. We include both traces captured in our premises and third-party traces available on public repositories. Traces cover multiple years, and total more than

[2]https://zeek.org
[3]https://www.wireshark.org/docs/man-pages/tshark.html
[4]https://www.malware-traffic-analysis.net
[5]https://www.netresec.com/?page=PcapFiles

TABLE I
FLOWS EXPORTED BY THE DIFFERENT TOOLS BEFORE THE PRE-PROCESSING.

Macrotrace	Tool	Flows	
		TCP	UDP
User Traffic	Tstat	681 k	1.1 M
	Libprotoident	678 k	1.1 M
	nDPI	543 k	1.1 M
	Zeek	804 k	1.2 M
Media & Games	Tstat	15 k	16 k
	Libprotoident	15 k	14 k
	nDPI	10 k	21 k
	Zeek	17 k	16 k
Malware	Tstat	858 k	979 k
	Libprotoident	858 k	993 k
	nDPI	891 k	1 M
	Zeek	1242 k	971 k
IoT	Tstat	118 k	50 k
	Libprotoident	118 k	51 k
	nDPI	120 k	62 k
	Zeek	119 k	52 k

TABLE II
MACROTRACES CHARACTERISTICS WITH PRE-PROCESSING RESULTS.

Macrotrace	Flows			Packets	
	TCP		UDP		
	Complete	Ignored		Original	Filtered
User	440 k	241 k	1.1 M	118 M	10.1 M
Media&Games	11 k	4 k	16 k	81 M	2 M
Malware	392 k	466 k	979 k	33 M	26 M
IoT	39 k	79 k	50 k	5 M	2 M

143 GB of PCAP files. For brevity, we do not provide details of each PCAP file here, instead describing only the aggregated *macrotraces*. To allow others reproduce our results, we link the public PCAP files in our website.[6]

We need to match flows as defined by each DPI tool for comparing their performance.[7] However, tools employ different rules for defining and exporting *flow records*. For example, each tool uses various timeouts to terminate flows that become inactive. Equally, traffic flags (e.g., TCP `FIN` and `RST` flags) are possibly used to identify the end of flows, releasing memory in the traffic monitor. The way such rules are implemented differs and, as a consequence, tools identify and report different numbers of flows. Thus, we need ingenuity to compare results.

Table I summarizes the number of flows reported by each tool. We see major differences, e.g., Zeek usually identifies more flows than Tstat, even when configured with similar timeouts. This happens because of the way midstream traffic and incomplete flows are processed by the tools.

Most of the cases creating discrepancies are however not interesting for our analysis, since they usually refer to flows that carry no payload. Indeed, a lot of flows without payload is present in particular for the Malware traces due to internet scanning traffic. These flows cannot be evaluated with DPI. As such, we perform a *pre-processing step* using Tstat as reference to keep in the final macrotraces only complete flows, i.e., UDP flows with payload and TCP flows with complete three-way handshake. All remaining flows are discarded. Whenever possible, we set the tools with similar timeout parameters for the experiments that will follow. We next normalize results ignoring the small percentage of flows that are not revealed by tools other than Tstat to avoid artifacts related to the way flow are expired or terminated. At last, we keep only the first 20 packets per flow in the final macrotraces to speed-up the analysis (see column "Filtered"). We will show later that all tested tools achieve a final protocol classification using a small number of packets per flow. As such, this pre-processing step does not impact results.

We report a summary of the final macrotraces in Table II. We show the number of packets and flows reported by Tstat, with the latter split as TCP and UDP. For TCP flows, we detail the number of *complete* and *ignored* flows.

In total, our final macrotraces include more than 3 M flows, and 40 M packets after all pre-processing steps are applied.

[6]https://smartdata.polito.it/dpi-in-practice/

[7]We use the classic 5-tuple definition for a flow: Source IP address, destination IP address, source port, destination port and transport protocol.

TABLE III
LABEL STANDARDIZATION

Standardized Label	Original Label
p2p	p2p, edonkey, emule, ed2k, cacaoweb, kademlia, bittorrent, torrent
netbiosSmb	netbios, smb, smb2, nbns
krb	krb, kerberos, spnego-krb5spnego
dns	dns, llmnr, mdns
sslTls	ssl, tls
skype	skype, skypetcp
ldap	ldap, cldap
quic	quic, gquic

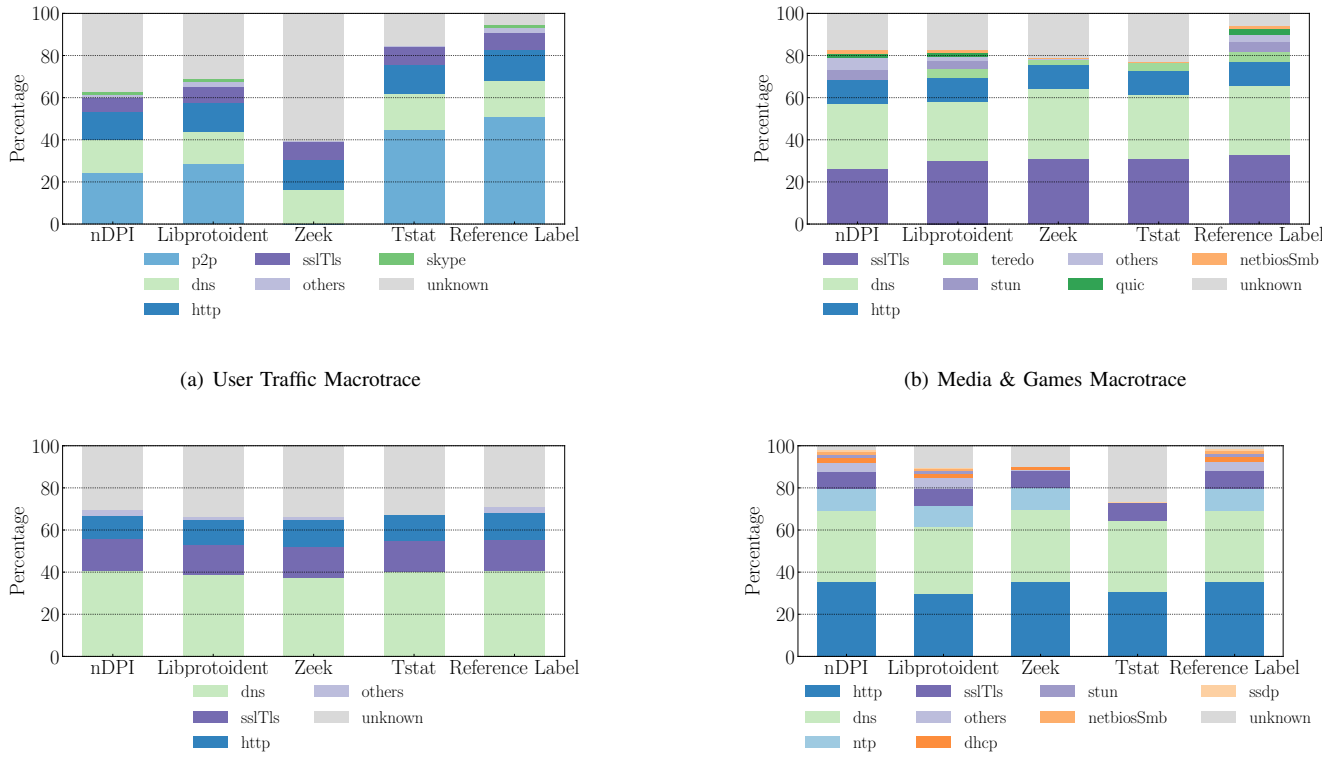
TABLE IV
EXAMPLE OF FLOW LABEL CONSTISTENCY AND SCORE.

Flow ID	Tool				Reference Label	Score
	Tstat	Libprotoident	nDPI	Zeek		
1	krb	krb	krb	krb	krb	1
2	unk	unk	unk	unk	unk	1
3	krb	unk	krb	krb	krb	0.75
4	unk	unk	krb	krb	krb	0.5
5	unk	unk	unk	krb	krb	0.25
6	unk	sip	unk	p2p	conflict	0
7	krb	krb	p2p	p2p	conflict	0

C. Matching flow labels

We need some ingenuity to normalize the output of the tools and compare their classifications. First, we normalize all labels, e.g., using always lower case and removing special characters. Then, we manually verify the output strings to identify possible synonyms used across tools. Table III reports a subset of labels that require manual standardization. In total we manually evaluated 225 labels, replacing cases such as those in the right column of Table III by a single common label (left column).

Next, we face the question on how to determine the label for each flow in absence of ground truth. Indeed, the lack of ground truth has pushed most of previous works to resort to



(a) User Traffic Macrotrace

(b) Media & Games Macrotrace

(c) Malware Macrotrace

(d) IoT Macrotrace

Fig. 2. Percentage of labelled flows for each tool. The last bar in the plots reports percentages for our reference label.

testbeds or emulated traffic that we want to avoid [11]. We thus decide to focus on the *consistency* of different tools, i.e., we assume that the most common normalized label assigned to a flow is the *reference label* for such flow, and calculate a *confidence score* for each decision. In case of conflicts, we manually verify each case.

Table IV reports examples of classification, along with the per-flow *confidence score*. The easiest cases happen when there is an unanimous decision towards the same protocol (e.g., Flow 1) or towards the unknown label (e.g., Flow 2). Both decisions result in a score equals to 1. When at least one tool is able to recognize the protocol, we ignore the unknown labels and pick the recognized label as reference label. Yet, our *confidence score* is lower in this case, e.g., see Flow 5. It rarely happens (e.g., Flow 6) that all tools recognize a different protocol, or there is a draw (e.g., Flow 7). Some of these cases have been solved by inspecting the source code of the DPI tools, e.g., giving preference to labels found by pattern matching over those guessed based on port numbers or other heuristics. The few cases we could not resolve are ignored, with confidence score equals to zero.

Finally, once the reference labels are defined, we calculate performance metrics for each tool. We consider the following metrics: (i) accuracy, the percentage of flows with label matching the reference; (ii) precision (per protocol), the percentage of such flows that match with the reference; and (iii) recall (per protocol), the percentage of such flows the tool has classified as the given protocol.

IV. Results

We show a summary of the identified flows per tool and we summarize the classification performance in the several scenarios. Next, we discuss the performance in terms of the number of packets required to reach a steady classification, and briefly discuss computational performance of tools.

A. Labelled flows per protocol

Fig. 2 shows a break-down of the number of labelled flows reported by each tool. Four plots depict results for the different macrotraces. The last bar on each plot reports the percentage of flows given by our *reference label*, i.e., the label selected by the majority of tools. Each figure reports the most common lables in order of popularity.

In the User Traffic case (top-left plot), Tstat shows the best performance, reporting labels for around 85% of the flows. All the libraries recognize popular protocols (e.g., HTTP, DNS and TLS), but Libprotoident, nDPI and Zeek fail to recognize some P2P traffic, thus leaving a larger number of flows marked as unknown. Yet, notice how the number of unknown flows is small for the reference label – i.e., flows marked as unknown by Tstat are recognized by others.

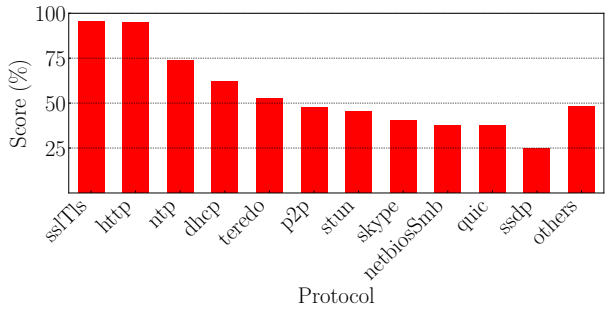


Fig. 3. Average per flow confidence score for the top reference labels.

In the Media & Games case – Fig. 2(b) – all tools recognize close to 80% of the flows. This trace is mostly composed by HTTP, DNS and TLS traffic, which are well recognized by all tools. The reference label reports again a lower percentage of unknown than each single tool, showing potential for achieving higher classifications by merging the output of different tools.

The analysis of the Malware macrotrace – Fig. 2(c) – leads to worse numbers for all cases. The percentage of labelled flows ranges from 66% to 70%. Here the presence of UDP scans towards multiple ports impact results. Manual inspection shows the presence of payload that matches the fingerprints of scan UDP attacks against certain IoT devices. None of the tools is able to identify the protocol of this malicious traffic, calling for specialized DPI approach in security use cases.

In the IoT case – Fig. 2(d) – nDPI is the best performing, labelling almost all flows. Tstat is penalized by the lack of fingerprints for NTP, STUN and SSDP. All in all, most flows in this trace are labelled by at least one tool (see the reference label bar).

Finally, we evaluate the average confidence scores for different protocols. With this analysis, we aim at identifying protocols for which the tools demonstrate high consistency. Fig. 3 shows the average scores for flows labeled with one of the top-20 protocols considering all four macrotraces. Common protocols such as TLS, HTTP and NTP are recognized with an average score higher or equal to 75% (left side of the figure). That is, such protocols are consistently identified by at least three tools on average. As we move to less popular labels, the confidence scores reduce significantly. Indeed, the score is reduced to around 25% for Netbios, QUIC and SSDP (right side of the figure). In other words, only one tool outputs a label for flows carrying these protocols, with others marking flows as unknown.

B. Classification performance

We next quantify the percentage of flows classified by each tool as well as their classification performance in respect to the reference labels. Results are presented in Tab. V. We highlight in bold the best performing tool per trace and metric.

Consider the first row group in the table. It reports the percentage of labelled flows, summarizing the results presented in the previous section. As said, Tstat reports more labels for the User Traffic scenario, thanks to its abilities to spot P2P flows. nDPI instead reaches the largest percentages in the other scenarios, thanks to its capabilities to guess labels based on multiple heuristics.

TABLE V
SUMMARY OF CLASSIFICATION RESULTS.

Metric	Library	Macrotrace			
		User Traffic	Games & Media	Malware	IoT
Labelled Flows	Tstat	**0.85**	0.77	0.67	0.73
	Libprotoident	0.69	**0.86**	0.66	0.89
	nDPI	0.63	**0.86**	**0.70**	**0.98**
	Zeek	0.40	0.78	0.66	0.89
Accuracy	Tstat	**0.85**	0.77	0.67	0.73
	Libprotoident	0.69	**0.82**	0.66	0.85
	nDPI	0.62	0.79	**0.70**	**0.98**
	Zeek	0.40	0.78	0.66	0.89
Average Precision	Tstat	0.99	0.87	0.98	**1**
	Libprotoident	0.96	0.91	0.99	0.80
	nDPI	0.93	0.89	**1**	0.99
	Zeek	**1**	**0.97**	**1**	**1**
Average Recall	Tstat	0.71	0.62	**1**	1
	Libprotoident	**1**	**0.89**	**1**	0.94
	nDPI	0.82	0.78	**1**	**1**
	Zeek	0.66	0.62	0.97	0.79

Considering accuracy (second row group), we see numbers similar to those for labelled flows across all scenarios. That is, the overall accuracy (with regards to the reference labels) is driven by the percentage of unknown flows reported by each tool. Yet, some particular cases can be noticed, such as minor differences between nDPI and Libprotoident in the Media & Games Macrotrace. These minor mismatches arise from cases in which one of the tools, although capable to label the given flow, disagree with the label given by the majority. As we see in the table, these cases are rare and indeed confirm that once tools labels a flow, the provided label is usually reliable.

Zeek wins when it comes to the average precision per protocol (third row group), almost always reaching 100%. That is, when Zeek recognizes a protocol, its label matches the reference. Yet, Zeek suffers in terms of average recall (fourth row group), due to its limited set of labels. Libprotoident, on the other hand, reaches the highest average recall per protocol in most scenarios, which can be explained by its large set of labels, with over 200 protocols. nDPI shows balanced numbers for both precision and recall per protocol. nDPI find a good number of labels (high recall) that usually match with the reference (high precision).

C. How many packets are needed for DPI?

We analyze the performance of tools while limiting the number of packets per flow. This test has been performed by cutting off each flow after observing its n first packets *with payload*, i.e., ignoring initial TCP handshake packets. Flows composed by n or less packets with payload are kept untouched. The goal is to evaluate the number of packets needed to reach a final classification, and whether labels change as more packets are observed.

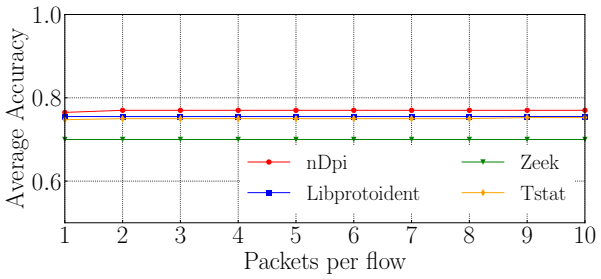


Fig. 4. Average accuracy when increasing the number of packets per flow. Tools reach a final classification already in the first packet with payload.

Fig. 4 shows the resulting average accuracy among all macrotraces. Clearly results do not change when increasing the number of packets, and all tools reach an almost steady classification after just one packet. Some tools (e.g., nDPI) increase accuracy further after observing the second packet with payload, but gains are marginal. This result is particularly relevant, as DPI tools are often used for real-time identification of protocols on security applications. Note that nDPI has average accuracy slightly superior than others, with Libprotoident and Tstat coming next.

Finally, we also controlled the performance of the tools in terms of memory fingerprint and processing time. Here a general conclusion is hard to be reached, since the tools are delivered for different target scenarios. For example, the basic installation of Zeek runs as multiple processes, prepared to handle several Gbps. Libprotoident and nDPI are libraries that can also be integrated in simple demonstration programs. In our tests, all tool, but Zeek, present similar performance figures when processing a single PCAP at a time.

V. Conclusions

We presented an evaluation of DPI solutions in several traffic scenarios, comparing the consistency of their classifications. The tools are practically equivalent when the input traffic is composed by popular and well-known protocols (e.g., HTTP, DNS and TLS). When applied to complex scenarios, such as to traffic generated by Malware scans, DPI tools struggle. We also observed discrepancies on the classification of less popular protocols, with some protocols being supported by only one of the tools. In sum, there is space for improving these DPI tools by extending their label sets. Interestingly, tools reach steady-state classification after one packet, suggesting they can be exploited in online scenarios.

References

[1] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular dpi tools for traffic classification," *Computer Networks*, vol. 76, pp. 75–89, 2015.

[2] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "ndpi: Open-source high-speed deep packet inspection," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2014, pp. 617–622.

[3] S. Alcock and R. Nelson, "Libprotoident: Traffic classification using lightweight packet inspection."

[4] M. Trevisan, A. Finamore, M. Mellia, M. Munafo, and D. Rossi, "Traffic Analysis with Off-the-Shelf Hardware: Challenges and Lessons Learned," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 163–169, 2017.

[5] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[6] A. Chaudhary and A. Sardana, "Software based implementation methodologies for deep packet inspection," in *2011 international conference on information science and applications*. IEEE, 2011, pp. 1–10.

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

[8] S. Kumar, J. Turner, and J. Williams, "Advanced algorithms for fast and scalable deep packet inspection," in *2006 Symposium on Architecture For Networking And Communications Systems*. IEEE, 2006, pp. 81–92.

[9] M. Mellia, R. L. Cigno, and F. Neri, "Measuring ip and tcp behavior on edge nodes with tstat," *Computer Networks*, vol. 47, no. 1, pp. 1–21, 2005.

[10] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *International Workshop on Passive and Active Network Measurement*. Springer, 2005, pp. 41–54.

[11] J. Yan, "A survey of traffic classification validation and ground truth collection," in *2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. IEEE, 2018, pp. 255–259.

[12] S. Alcock and R. Nelson, "Measuring the accuracy of open-source payload-based traffic classifiers using popular internet applications," in *38th Annual IEEE Conference on Local Computer Networks - Workshops*, 2013, pp. 956–963.

[13] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the validation of traffic classification algorithms," in *International Conference on Passive and Active Network Measurement*. Springer, 2008, pp. 72–81.

[14] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. Claffy, "Gt: picking up the truth from the ground for internet traffic," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 12–18, 2009.

[15] P. Lizhi, Z. Hongli, Y. Bo, C. Yuehui, and W. Tong, "Traffic labeller: collecting internet traffic samples with accurate application information," *China Communications*, vol. 11, no. 1, pp. 69–78, 2014.

[16] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proceedings of the 2008 ACM CoNEXT Conference*, ser. CoNEXT '08. New York, NY, USA: Association for Computing Machinery, 2008. [Online]. Available: https://doi.org/10.1145/1544012.1544023

[17] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "Portload: Taking the best of two worlds in traffic classification," in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, 2010, pp. 1–5.

[18] J. Khalife, A. Hajjar, and J. Díaz-Verdejo, "Performance of opendpi in identifying sampled network traffic," *Journal of Networks*, vol. 8, no. 1, p. 71, 2013.

[19] A. Nisticò, D. Markudova, M. Trevisan, M. Meo, and G. Carofiglio, "A comparative study of rtc applications," *To appear in the Proceedings of the 22nd IEEE International Symposium on Multimedia*, 2020.

[20] A. Dainotti, A. Pescapé, and G. Ventre, "A packet-level characterization of network traffic," in *2006 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*. IEEE, 2006, pp. 38–45.

[21] A. Di Domenico, G. Perna, M. Trevisan, L. Vassio, and D. Giordano, "A network analysis on cloud gaming: Stadia, geforce now and psnow," *arXiv preprint arXiv:2012.06774*, 2020.

[22] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[23] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[24] A. Parmisano, S. Garcia, and M. Erquiaga, "A labeled dataset with malicious and benign iot network traffic," *Stratosphere Laboratory: Praha, Czech Republic*, 2020.