Is our Ground-Truth for Traffic Classification Reliable?*

Valentín Carela-Español[1], Tomasz Bujlow[2], and Pere Barlet-Ros[1]

[1] UPC BarcelonaTech, Spain
[2] Aalborg University, Denmark

Abstract. The validation of the different proposals in the traffic classification literature is a controversial issue. Usually, these works base their results on a ground-truth built from private datasets and labeled by techniques of unknown reliability. This makes the validation and comparison with other solutions an extremely difficult task.
This paper aims to be a first step towards addressing the validation and trustworthiness problem of network traffic classifiers. We perform a comparison between 6 well-known DPI-based techniques, which are frequently used in the literature for ground-truth generation. In order to evaluate these tools we have carefully built a labeled dataset of more than 500 000 flows, which contains traffic from popular applications. Our results present *PACE*, a commercial tool, as the most reliable solution for ground-truth generation. However, among the open-source tools available, *NDPI* and especially *Libprotoident*, also achieve very high precision, while other, more frequently used tools (e.g., *L7-filter*) are not reliable enough and should not be used for ground-truth generation in their current form.

1 Introduction and Related Work

During the last decade, traffic classification has considerably increased its relevance, becoming a key aspect for many network related tasks. The explosion of new applications and techniques to avoid detection (e.g., encryption, protocol obfuscation) have substantially increased the difficulty of traffic classification. The research community have thrown itself into this problem by proposing many different solutions. However, this problem is still far from being solved [1].

Most traffic classification solutions proposed in the literature report very high accuracy. However, these solutions mostly base their results on a private ground-truth (i.e., dataset), usually labeled by techniques of unknown reliability (e.g., ports-based or DPI-based techniques [2–5]). That makes it very difficult to compare and validate the different proposals. The use of private datasets is derived from the lack of publicly available datasets with payload. Mainly because of privacy issues, researchers and practitioners are not allowed to share their datasets

* This research was funded by the Spanish Ministry of Economy and Competitiveness under contract TEC2011-27474 (NOMADS project) and by the Comissionat per a Universitats i Recerca del DIUE de la Generalitat de Catalunya (ref. 2009SGR-1140).

Table 1. DPI-based techniques evaluated

Name	Version	Applications
PACE	1.41 (June 2012)	1000
OpenDPI	1.3.0 (June 2011)	100
NDPI	rev. 6391 (March 2013)	170
L7-filter	2009.05.28 (May 2009)	110
Libprotoident	2.0.6 (Nov 2012)	250
NBAR	15.2(4)M2 (Nov 2012)	85

with the research community. To the best of our knowledge, just one work has tackled this problem. Gringoli et al. in [6] published anonymized traces without payload, but accurately labeled using GT. This dataset is very interesting to evaluate Machine Learning-based classifiers, but the lack of payload makes it unsuitable for DPI-based evaluation.

Another crucial problem is the reliability of the techniques used to set the ground-truth. Most papers show that researchers usually obtain their ground-truth through port-based or DPI-based techniques [2–5]. The poor reliability of port-based techniques is already well known, given the use of dynamic ports or well-known ports of other applications [7, 8]. Although the reliability of DPI-based techniques is still unknown, according to conventional wisdom they are, in principle, one of the most accurate techniques.

Some previous works evaluated the accuracy of DPI-based techniques [3, 5, 9, 10]. These studies rely on a ground-truth generated by another DPI-based tool [5], port-based technique [3] or a methodology of unknown reliability [9,10], making their comparison and validation very difficult. Recently, a concomitant study to ours [10] compared the performance of four DPI-based techniques (i.e., *L7-filter*, *Tstat*, *NDPI* and *Libprotoident*). This parallel study confirms some of the findings of our work presenting *NDPI* and *Libprotoident* as the most accurate open-source DPI-based techniques. In [11] the reliability of L7-filter and a port-based technique was compared using a dataset obtained by GT [6] showing that both techniques present severe problems to accurately classify the traffic.

This paper presents two main contributions. First, we publish a reliable labeled dataset with full packet payloads [12]. The dataset has been artificially built in order to allow us its publication. However, we have manually simulated different behaviours to make it as representative as possible. We used VBS [13] to guarantee the reliability of the labeling process. This tool can label the flows with the name of the process that created them. This allowed us to carefully create a reliable ground-truth that can be used as a reference benchmark for the research community. Second, using this dataset, we evaluated the performance and compared the results of 6 well-known DPI-based techniques, presented in Table 1, which are widely used for the ground-truth generation in the traffic classification literature.

These contributions pretend to be a first step towards the impartial validation of network traffic classifiers. They also provide to the research community some insights about the reliability of different DPI-based techniques commonly used in the literature for ground-truth generation.

2 Methodology

The Testbed. Our testbed is based on VMWare virtual machines (VM). We installed three VM for our data generating stations and we equipped them with Windows 7 (W7), Windows XP (XP), and Ubuntu 12.04 (LX). Additionally, we installed a server VM for data storage. To collect and accurately label the flows, we adapted Volunteer-Based System (VBS) developed at Aalborg University [13]. The task of VBS is to collect information about Internet traffic flows (i.e., start time of the flow, number of packets contained by the flow, local and remote IP addresses, local and remote ports, transport layer protocol) together with detailed information about each packet (i.e., direction, size, TCP flags, and relative timestamp to the previous packet in the flow). For each flow, the system also collects the process name associated with that flow. The process name is obtained from the system sockets. This way, we can ensure the application associated to a particular traffic. Additionally, the system collects some information about the HTTP content type (e.g., *text/html*, *video/x-flv*). The captured information is transmitted to the VBS server, which stores the data in a MySQL database. The design of VBS was initially described in [13]. On every data generating VM, we installed a modified version of VBS. The source code of the modified version was published in [14] under a *GPL license*. The modified version of the VBS client captures full Ethernet frames for each packet, extracts HTTP *URL* and *Referer* fields. We added a module called *pcapBuilder*, which is responsible for dumping the packets from the database to PCAP files. At the same time, INFO files are generated to provide detailed information about each flow, which allows us to assign each packet from the PCAP file to an individual flow. We also added a module called *logAnalyzer*, which is responsible for analyzing the logs generated by the different DPI tools, and assigning the results of the classification to the flows stored in the database.

Selection of the Data. The process of building a representative dataset, which characterizes a typical user behavior, is a challenging task, crucial on testing and comparing different traffic classifiers. Therefore, to ensure the proper diversity and amount of the included data, we decided to combine the data on a multidimensional level. Based on w3schools statistics, we selected Windows 7 (55.3 %) of all users), Windows XP (19.9 %), and Linux (4.8 %) - state for January 2013. Apple computers (9.3 % of overall traffic) and mobile devices (2.2 %) were left as future work. The selected applications are shown below.

- Web browsers: based on w3schools statistics: Chrome and Firefox (W7, XP, LX), Internet Explorer (W7, XP).
- BitTorrent clients: based on CNET ranking: uTorrent and Bittorrent (W7, XP), Frostwire and Vuze (W7, XP, LX)
- eDonkey clients: based on CNET ranking: eMule (W7, XP), aMule (LX)
- FTP clients: based on CNET ranking: FileZilla (W7, XP, LX), SmartFTP Client (W7, XP), CuteFTP (W7, XP), WinSCP (W7, XP)
- Remote Desktop servers: built-in (W7, XP), xrdp (LX)

– SSH servers: sshd (LX)
– Background traffic: DNS and NTP (W7, XP, LX), NETBIOS (W7, XP)

The list of visited websites was based on the top 500 websites according to Alexa statistics. We chose several of them taking into account their rank and the nature of the website (e.g., search engines, social medias, national portals, video websites) to assure the variety of produced traffic. These websites include: Google, Facebook, YouTube, Yahoo!, Wikipedia, Java, and Justin.tv. For most websites we performed several random clicks to linked external websites, which should better characterize the real behavior of the real users and include also other websites not included in the top 500 ranking. This also concerns search engines, from which we manually generated random clicks to the destination web sites. Each of the chosen websites was processed by each browser. In case it was required to log into the website, we created fake accounts. In order to make the dataset as representative as possible we have simulated different human behaviors when using these websites. For instance, on Facebook, we log in, interact with friends (e.g., chat, send messages, write in their walls), upload pictures, create events or play games. On YouTube, we watched the 10 most popular videos, which we randomly paused, resumed, and rewound backward and forward. Also, we randomly made some comments and clicked *Like* or *Not like* buttons. The detailed description of actions performed with the services is listed in our technical report [15]. We tested the P2P (BitTorrent and eDonkey) clients by downloading files of different sizes and then leaving the files to be seeded for some time, in order to obtain enough of traffic in both directions. We tried to test every FTP client using both the active transfer mode (PORT) and passive transfer mode (PASV), if the client supports such mode.

Extracting the Data for Processing. Each DPI tool can have different requirements and features, so the extracting tool must handle all these issues. The PCAP files provided to PACE, OpenDPI, L7-filter, NDPI, and Libprotoident are accompanied by INFO files, which contain the information about the start and end of each flow, together with the flow identifier. Because of that, the software, which uses the DPI libraries, can create and terminate the flows appropriately, as well as to provide the classification results together with the flow identifier. Preparing the data for NBAR classification is more complicated. There are no separate INFO files describing the flows, since the classification is made directly on the router. We needed to extract the packets in a way that allows the router to process and correctly group them into flows. We achieved that by changing both the source and destination MAC addresses during the extraction process. The destination MAC address of every packet must match up with the MAC address of the interface of the router, because the router cannot process any packet which is not directed to its interface on the MAC layer. The source MAC address was set up to contain the identifier of the flow to which it belongs, so the flows were recognized by the router according to our demands. To the best of our knowledge, this is the first work to present a scientific performance evaluation of NBAR.

Table 2. Application classes in the dataset

Application	No. of flows	No. of Megabytes
Edonkey	176581	2823.88
BitTorrent	62845	2621.37
FTP	876	3089.06
DNS	6600	1.74
NTP	27786	4.03
RDP	132907	13218.47
NETBIOS	9445	5.17
SSH	26219	91.80
Browser HTTP	46669	5757.32
Browser RTMP	427	3026.57
Unclassified	771667	5907.15

The Classification Process. We designed a tool, called *dpi_ benchmark*, which can read the PCAP files and provide the packets one-by-one to *PACE, OpenDPI, L7-filter, NDPI* and *Libprotoident*. All the flows are started and terminated based on the information from the INFO files. After the last packet of the flow is sent to the classifier, the tool obtains the classification label associated with that flow. The labels are written to the log files together with the flow identifier, which makes us later able to relate the classification results to the original flows in the database. A brief description of the DPI-tools used in this study is presented in Table 1. Although some of the evaluated tools have multiple configuration parameters, we have used in our evaluation the default configuration for most of them. A detailed description of the evaluated DPI-tools and their configurations can be found in [15].

Classification by *NBAR* required us to set up a full working environment. We used GNS3 - a graphical framework, which uses Dynamips to emulate our Cisco hardware. We emulated the 7200 platform, since only for this platform supported by GNS3 was available the newest version of Cisco IOS (version 15), which contains Flexible NetFlow. The router was configured by us to use Flexible NetFlow with *NBAR* on the created interface. Flexible NetFlow was set up to create the flows taking into account the same parameters as are used to create the flow by VBS. On the computer, we used *tcpreplay* to replay the PCAP files to the router with the maximal speed, which did not cause packet loss. At the same time, we used *nfacctd*, which is a part of PMACCT tools, to capture the Flexible NetFlow records sent by the router to the computer. The records, which contain the flow identifier (encoded as source MAC address) and the name of the application recognized by *NBAR*, were saved into text log files. This process is broadly elaborated in our technical report [15].

The Dataset. Our dataset contains 1 262 022 flows captured during 66 days, between February 25, 2013 and May 1, 2013, which account for 35.69 GB of pure packet data. The application name tag was present for 520 993 flows (41.28 % of all the flows), which account for 32.33 GB (90.59 %) of the data volume. Additionally, 14 445 flows (1.14 % of all the flows), accounting for 0.28 GB (0.78 %) of data volume, could be identified based on the HTTP *content-type* field extracted from the packets. Therefore, we were able to successfully establish the ground truth for 535 438 flows (42.43 % of all the flows), accounting for 32.61 GB (91.37 %) of data volume. The remaining flows are unlabeled due to their short lifetime (below ~1 s), which made VBS incapable to reliably establish the corresponding sockets. Only these successfully classified flows will be taken into account during the evaluation of the classifiers. However, all the flows are included in the publicly available traces. This ensures data integrity and the proper work of the classifiers, which may rely on coexistence of different flows. We isolated several application classes based on the information stored in the database (e.g., application labels, HTTP *content-type* field). The classes together with the number of flows and the data volume are shown in Table 2. We have published this labeled dataset with full packet payloads in [12]. Therefore, it can be used by the research community as a reference benchmark for the validation and comparison of network traffic classifiers.

3 Performance Comparison

This section provides a detailed insight into the classification results of different types of traffic by each of the classifiers. All these results are summarized in Table 3, where the ratio of correctly classified flows (i.e., precision or true positives), incorrectly classified flows (i.e., errors or false positives) and unclassified flows (i.e., unknowns) are respectively presented. The complete confusion matrix can be found in our technical report [15].

Regarding the classification of P2P traffic, *Edonkey* is the first application studied. Only *PACE*, and especially *Libprotoident*, can properly classify it (precision over 94 %). *NDPI* and *OpenDPI* (that use the same pattern), as well as *NBAR*, can classify almost no *Edonkey* traffic (precision below 1 %). *L7-filter* classifies 1/3 of the flows, but it also produces many false positives by classifying more than 13 % of the flows as *Skype*, *NTP*, and *finger*. The wrongly classified flows in *NDPI* were labeled as *Skype*, *RTP* and *RTCP*, and in *NBAR* as *Skype*. The classification of *BitTorrent* traffic, the second P2P application studied, is not completely achieved by any of the classifiers. *PACE* and *Libprotoident* achieve again the highest precision (over 77 %). The rest of the classifiers present severe problems to identify this type of traffic. When misclassified, the *BitTorrent* traffic is usually classified as *Skype*.

The performance of most DPI tools with more traditional applications is significantly higher. FTP traffic is usually correctly classified. Only *L7-filter* and *NBAR* present problems to label it. The false positives produced by *L7-filter* are because the traffic is classified as *SOCKS*. Table 3 also shows that all the classifiers can properly classify *DNS* traffic. Similar results are obtained for *NTP*, which almost all the classifiers can correctly classify it. However, *NBAR* completely miss the classification of this traffic. *SSH* was evaluated in its *Linux* version. Table 3 shows that *NBAR* almost classified all the flows while the rest of classifiers labeled more than 95 % of them.

Similar performance is also obtained with *RDP*, usually employed by VoIP applications, as shown in Table 3. Again, *L7-filter* and *NBAR* can not classify this application at all. The false positives for *L7-filter*, *Libprotoident*, and *NBAR* are mainly due to *Skype*, *RTMP*, and *H323*, respectively.

Unlike previous applications, the results for *NETBIOS* are quite different. Surprisingly, *NBAR* and *NDPI* are the only classifiers that correctly label *NETBIOS* traffic. *PACE* can classify 2/3 of this traffic and *OpenDPI* only 1/4. On the other hand, the patterns from *L7-filter* and *Libprotoident* do not properly detect this traffic. The wrongly classified flows in *Libprotoident* are labeled as *RTP* and *Skype*, and in *L7-filter* as *Edonkey*, *NTP*, and *RTP*.

We also evaluated *RTMP* traffic, a common protocol used by browsers and plugins for playing *FLASH* content. It is important to note that only *Libprotoident* has a specific pattern for *RTMP*. Because of that, we have also counted as correct the *RTMP* traffic classified as *FLASH* although that classification is not as precise as the one obtained by *Libprotoident*. *L7-filter* and *NBAR* can not classify this type of traffic. The rest of the classifiers achieve a similar precision, around 80 %. The surprising amount of false positives by *NDPI* is because some traffic is classified as *H323*. *L7-filter* errors are due to wrongly classified traffic as *Skype* and *TSP*.

Table 3 also presents the results regarding the *HTTP* protocol. All of them but *L7-filter* can properly classify most of the *HTTP* traffic. *L7-filter* labels all the traffic as *finger* or *Skype*. *NDPI* classifies some *HTTP* traffic as *iMessage_Facetime*. The amount of errors from PACE is surprising, as this tool is usually characterized by very low false positive ratio. All the wrong classifications are labeled as *Meebo* traffic. The older *Meebo* pattern available in *OpenDPI* and the newer from *NDPI* seems not to have this problem.

Most incorrect classifications for all the tools are due to patterns that easily match random traffic. This problem especially affects L7-filter and, in particular, with the patterns used to match *Skype*, *finger* and *ntp* traffic. The deactivation of those patterns would considerably decrease the false positive ratio but it would disable the classification of those applications. In [4], the authors use a tailor-made configuration and post-processing of the L7-filter output in order to minimize this overmatching problem.

Table 3. DPI evaluation

Application	Classifier	% correct	% wrong	% uncl.	Application	Classifier	% correct	% wrong	% uncl.
	PACE	94.80	0.02	5.18		PACE	95.57	0.00	4.43
	OpenDPI	0.45	0.00	99.55		OpenDPI	95.59	0.00	4.41
Edonkey	L7-filter	34.21	13.70	52.09	SSH	L7-filter	95.71	0.00	4.29
	NDPI	0.45	6.72	92.83		NDPI	95.59	0.00	4.41
	Libprotoident	98.39	0.00	1.60		Libprotoident	95.71	0.00	4.30
	NBAR	0.38	10.81	88.81		NBAR	99.24	0.05	0.70
	PACE	81.44	0.01	18.54		PACE	99.04	0.02	0.94
	OpenDPI	27.23	0.00	72.77		OpenDPI	99.07	0.02	0.91
BitTorrent	L7-filter	42.17	8.78	49.05	RDP	L7-filter	0.00	91.21	8.79
	NDPI	56.00	0.43	43.58		NDPI	99.05	0.08	0.87
	Libprotoident	77.24	0.06	22.71		Libprotoident	98.83	0.16	1.01
	NBAR	27.44	1.49	71.07		NBAR	0.00	0.66	99.34
	PACE	95.92	0.00	4.08		PACE	66.66	0.08	33.26
	OpenDPI	96.15	0.00	3.85		OpenDPI	24.63	0.00	75.37
FTP	L7-filter	6.11	93.31	0.57	NETBIOS	L7-filter	0.00	8.45	91.55
	NDPI	95.69	0.45	3.85		NDPI	100.00	0.00	0.00
	Libprotoident	95.58	0.00	4.42		Libprotoident	0.00	5.03	94.97
	NBAR	40.59	0.00	59.41		NBAR	100.00	0.00	0.00
	PACE	99.97	0.00	0.03		PACE	80.56	0.00	19.44
	OpenDPI	99.97	0.00	0.03		OpenDPI	82.44	0.00	17.56
DNS	L7-filter	98.95	0.13	0.92	RTMP	L7-filter	0.00	24.12	75.88
	NDPI	99.88	0.09	0.03		NDPI	78.92	8.90	12.18
	Libprotoident	99.97	0.00	0.04		Libprotoident	77.28	0.47	22.25
	NBAR	99.97	0.02	0.02		NBAR	0.23	0.23	99.53
	PACE	100.00	0.00	0.00		PACE	96.16	1.85	1.99
	OpenDPI	100.00	0.00	0.00		OpenDPI	98.01	0.00	1.99
NTP	L7-filter	99.83	0.15	0.02	HTTP	L7-filter	4.31	95.67	0.02
	NDPI	100.00	0.00	0.00		NDPI	99.18	0.76	0.06
	Libprotoident	100.00	0.00	0.00		Libprotoident	98.66	0.00	1.34
	NBAR	0.40	0.00	99.60		NBAR	99.58	0.00	0.42

3.1 Sub-classification of HTTP traffic

Our dataset also allows the study of *HTTP* traffic at different granularity (e.g., identify different services running over *HTTP*). However, only *NDPI* can sub-classify some applications at this granularity (e.g., *Youtube*, *Facebook*). Newer versions of *PACE* also provide this feature but we had no access to it for this study. Table 4 presents the results for four applications running over *HTTP* identified by *NDPI*. Unlike the rest of tools that basically classify this traffic as *HTTP*, *NDPI* can correctly give the specific label with precision higher than 97 %. Furthermore, the classification errors are caused by traffic that *NDPI* classifies as *HTTP* without providing the lower level label.

Table 4. HTTP sub-classification by *NDPI*

Application	% correct	% wrong	% unclassified
Google	97.28	2.72	0.00
Facebook	100.00	0.00	0.00
Youtube	98.65	0.45	0.90
Twitter	99.75	0.00	0.25

Another sub-classification that can be studied with our dataset is the *FLASH* traffic over *HTTP*. However, the classification of this application is different for each tool making its comparison very difficult. *PACE*, *OpenDPI* and *NDPI* have a specific pattern for this application. At the same time, these tools (as well as *L7-filter*) have specific patterns for video traffic, which may or may not run over *HTTP*. In addition, *NDPI* has specific labels for *Google*, *Youtube* and *Facebook* that can also carry *FLASH* traffic. *Libprotoident* and *NBAR* do not provide any pattern to classify *FLASH* traffic over *HTTP*. Table 5 shows that *NDPI* can correctly classify 99.48 % of this traffic, 25.48 % of which is classified as *Google*, *Youtube* or *Facebook*. *PACE* and *OpenDPI* can properly classify around 86 % of the traffic. The errors produced in the classification are almost always related to traffic classified as *HTTP* with the exception of *L7-filter* that classifies 86.49 % of the traffic as *finger*.

Table 5. FLASH evaluation

Classifier	% correct	% wrong	% unclassified
PACE	86.27	13.18	0.55
OpenDPI	86.34	13.15	0.51
L7-filter	0.07	99.67	0.26
NDPI	99.48	0.26	0.26
Libprotoident	0.00	98.07	1.93
NBAR	0.00	100.00	0.00

4 Discussion

This section extracts the outcomes from the results obtained during the performance comparison. Also, we discuss the limitations of our study. Table 6 presents the summary of the results from Section 3. The *Precision* (i.e., first column) is computed similarly to Section 3, but we take into account all the applications together (i.e., 100 * # correctly classified flows / # total flows). However, this metric is dependent on the distribution of the dataset. Because of that, we also compute a second metric, the *Average Precision*. This statistic is independent from the distribution and is calculated as follow:

$$Avg.\ Precision = \frac{\sum_{i=1}^{N} \frac{correctly\ classified\ i\ flows}{total\ i\ flows}}{N}$$

where N is the number of applications studied (i.e., N = 10).

As it can be seen in Table 6, *PACE* is the best classifier. Even while we were not using the last version of the software, PACE was able to properly classify 94 % of our dataset. Surprisingly for us, *Libprotoident* achieves similar results, although this tool only inspect the first four bytes of payload for each direction. On the other hand, *L7-filter* and *NBAR* perform poorly in classifying the traffic from our dataset. The more fair metric, *Avg. Precision*, presents similar results. *PACE* is still the best classifier, however, it has increased the difference by several points to the second best classifier, *Libprotoident*. Unlike before, *NDPI* is almost as precise as *Libprotoident* with this metric. *L7-filter* and *NBAR* are still the tools that present the worst performance.

Table 6. Summary

Classifier	% Precision	% Avg. Precision
PACE	94.22	91.01
OpenDPI	52.67	72.35
L7-filter	30.26	38.13
NDPI	57.91	82.48
Libprotoident	93.86	84.16
NBAR	21.79	46.72

Nonetheless, the previous conclusions are obviously tied to our dataset. Although we have tried our best to emulate the real behavior of the users, many applications, behaviors and configurations are not represented on it. Because of that, it has some limitations. In our study we have evaluated 10 well-known applications, however adding more applications as *Skype* or *Spotify* is part of our ongoing future work. The results obtained from the different classifiers are directly related to those applications. Thus, the introduction of different applications could arise different outcomes. The traffic generated for building the dataset, although has been manually and realistically created, is artificial. The backbone traffic would carry different behaviors of the applications that are not fully represented in our dataset (e.g., P2P clients running on port 80). Therefore, the performance of the tools studied could not be directly extrapolated from the current results, but it gives an idea of their precision in the evaluated set of applications. At the same time, the artificially created traffic allowed us to publish the dataset with full packet payloads.

5 Conclusions

This paper presents the first step towards validating the reliability of the accuracy of the network traffic classifiers. We have compared the performance of six tools (i.e., *PACE*, *OpenDPI*, *L7-filter*, *NDPI*, *Libprotoident*, and *NBAR*), which are usually used for the traffic classification. The results obtained in Section 3 and further discussed in Section 4 show that *PACE* is, on our dataset, the most reliable solution for traffic classification. Among the open-source tools, *NDPI* and especially *Libprotoident* present the best results. On the other hand, *NBAR* and *L7-filter* present several inaccuracies that make them not recommendable as a ground-truth generator.

In order to make the study trustworthy, we have created a dataset using VBS [13]. This tool associates the name of the process to each flow making its labeling totally reliable. The dataset of more than 500 K flows contains traffic from popular applications like *HTTP*, *Edonkey*, *BitTorrent*, *FTP*, *DNS*, *NTP*, *RDP*, *NETBIOS*, *SSH*, and *RDP*. The total amount of data properly labeled is 32.61 GB. Furthermore, and more important, we release to the research community this dataset with full payload, so it can be used as a common reference for the comparison and validation of network traffic classifiers.

As the future work, we plan to extend this work by adding new applications to the dataset (e.g., Skype, Games) and especially focus on HTTP-based applications. We also plan to introduce new tools to the study (e.g., NBAR2).

References

1. Dainotti, A. et al.: Issues and future directions in traffic classification. IEEE Network **26**(1) (2012) 35–40
2. Valenti, S. et al.: Reviewing Traffic Classification. In: Data Traffic Monitoring and Analysis, Springer (2013) 123–147
3. Fukuda, K.: Difficulties of identifying application type in backbone traffic. In: Int. Conf. on Network and Service Management (CNSM), IEEE (2010) 358–361
4. Carela-Español, V. et al.: Analysis of the impact of sampling on NetFlow traffic classification. Computer Networks **55** (2011) 1083–1099
5. Alcock, S. et al.: Libprotoident: Traffic Classification Using Lightweight Packet Inspection. Technical report, University of Waikato (2012)
6. Gringoli, F. et al.: Gt: picking up the truth from the ground for internet traffic. ACM SIGCOMM Computer Communication Review **39**(5) (2009) 12–18
7. Dainotti, A. et al.: Identification of traffic flows hiding behind TCP port 80. In: IEEE Int. Conf. on Communications (ICC). (2010) 1–6
8. Karagiannis, T. et al.: Transport layer identification of P2P traffic. In: 4th ACM Internet Measurement Conf. (IMC). (2004) 121–134
9. Shen, C. et al.: On detection accuracy of L7-filter and OpenDPI. In: 3th Int. Conf. on Networking and Distributed Computing (ICNDC), IEEE (2012) 119–123
10. Alcock, Shane and Nelson, Richard: Measuring the Accuracy of Open-Source Payload-Based Traffic Classifiers Using Popular Internet Applications. In: IEEE Workshop on Network Measurements. (2013)
11. Dusi, M. et al.: Quantifying the accuracy of the ground truth associated with Internet traffic traces. Computer Networks **55**(5) (2011) 1158–1167
12. [Online]: Traffic classification at the Universitat Politècnica de Catalunya (UPC). (2013) URL: `http://monitoring.ccaba.upc.edu/traffic_classification`.
13. Bujlow, T. et al.: Volunteer-Based System for classification of traffic in computer networks. In: 19th Telecommunications Forum TELFOR, IEEE (2011) 210–213
14. [Online]: Volunteer-Based System for Research on the Internet (2012) URL: `http://vbsi.sourceforge.net/`.
15. Bujlow, T. et al.: Comparison of Deep Packet Inspection (DPI) Tools for Traffic Classification. Technical report, UPC BarcelonaTech (2013)