Empirical analysis of traffic to establish a profiled flow termination timeout

Juan Molina Rodríguez
Telecommunication and Electronic Eng.
UPC BarcelonaTech
Barcelona, Spain
juan.molina@alu-etsetb.upc.edu

Valentín Carela Español
and Pere Barlet Ros
UPC BarcelonaTech
Barcelona, Spain
{vcarela, pbarlet}@ac.upc.edu

Ralf Hoffmann
and Klaus Degner
ipoque GmbH
Leipzig, Germany
{ralf.hoffmann, klaus.degner}@ipoque.com

***Abstract*—The exponential increase of bandwidth on the Internet has made the online traffic classification a highly exigent task. All the operations in the classification process must be efficiently implemented in order to deal with an enormous amount of data. A key point in this process is the selection of a flow termination, a decision that has important consequences for several traffic classification techniques (e.g., DPI-based, Machine Learning-based). For instance, properly expiring the flows reduces the amount of memory necessary and avoids erroneous computation of flow features. In addition, the heterogeneous behaviour of the applications on the Internet have dismissed the traditional techniques to determine the flow termination (i.e., TCP 3/4-way handshake, TCP timeout). In this paper, we first perform a comprehensive study of the flow termination by application groups. Results confirm that traditional techniques are no longer sufficient to determine the flow termination (i.e., <50% finish with TCP handshake for some groups). In order to address this new scenario we propose a profiled (i.e., by application group) flow termination timeout. This solution has been evaluated in a well-known commercial DPI tool (the Ipoque's PACE engine) achieving a drastic reduction of memory, while keeping the same computation cost and classification accuracy. In order to obtain representative results, two completely different traces have been analysed, one from the core network of a large ISP and another from the edge link of a mobile operator.**

***Index Terms*—Traffic classification, profiled termination timeout, classifier optimization.**

I. Introduction

During the last years network traffic classification has become a prolific research topic that it is far from being totally addressed [1]. The increase of bandwidth and diversity of protocols on the Internet has made the online traffic classification a difficult and exigent task. In current networks (e.g. 10/40/100 Gbps) packets are received every few ns, and that is the time available for parsing them without requiring data buffering. However, many proposed traffic classifiers require to keep the state of each flow or to store statistics or even the actual content of the packets to properly classify it. Thus, the operations involved in this process have to be carried out very efficiently in terms of memory and CPU (Central Processing Unit).

We focus our attention in the optimization of the memory consumption. A crucial issue in this process is the expiration of the structure that keeps the state of each flow. On the one hand, expiring the flows too late increases the amount of memory necessary. On the other hand, expiring the flows quickly would create segmented flows with less information, making the accuracy of classification more difficult or even impossible. So far, this process has been usually carried out using timeouts or specific protocol mechanisms like the TCP (Transmission Control Protocol) termination handshake.

The first contribution of this paper is a comprehensive study of the flow termination by group of applications with current Internet traffic. The study shows that non-desired flow terminations, like RST (Reset) or undefined terminations, are considerably common. Furthermore, we have detected different behaviours related to the flow termination among the different groups of applications. From that characterization we have gone further and proposed an expiration technique to achieve optimized timeouts in a profiled way. Our method has been evaluated using the PACE engine [2], a well-known commercial DPI (Deep Packet Inspection) tool. We achieve a substantial reduction of memory while maintaining the accuracy and the CPU consumption.

The rest of the paper is organized as follows. Section II reviews the related work. Section III describes the traces used. Section IV briefly presents the methodology used to extract the results. Section V presents the statistical results, and based on them, the estimation and evaluation of the timeouts is done. Finally, section VI concludes the paper.

II. Related work

To the best of our knowledge, this is the first paper that presents a comprehensive study about the flow termination by group of applications. Some previous studies concern this topic but in a simplified way or considering just a few parameters. In [3] it studied what is the influence of different types of traffic on the Internet in terms of volume of data, the behaviour of establishment and termination. The classification only covers P2P (Peer To Peer) and HTTP (Hypertext Trasfer Protocol) traffic. Concerning the statistical study of different protocol groups, the related works are either old or sharing just a few similarities. Just a few mentions. In [4] it is proposed a methond to establish a single timeout to prevent DoS (Denial of Service) attacks by studying the initial RTT (Round Trip Time). Another exception is [5]. Although this work is quite old (1995) it establishes a single timeout which can be

978-1-4673-2480-9/13/$31.00 ©2013 IEEE

compared to this paper findings. The use of a single timeout for the expiration of the flows is a mechanism commonly used by the traffic classification techniques proposed in the literature [6], [7].

III. Used traces

The traces used for the evaluation were recorded in two different network scenarios and they provide a good comparative basis. They were taken under confidential agreements, so specific details can not be revealed. In Table I are synthesised some of their properties.

a) ISP_Core: This trace was recorded on 2009 in an internal link of a Tier-1 ISP, after the access and aggregation sections and before the output router to the backbone. This type of networks are often designed with multiple links due to balancing and failure issues, so many of the traffic recorded is asymmetric.

b) ISP_Mob: This trace is a good complement for the `ISP_Core` one since it was recorded in a mobile operator network. It was taken somewhere in a middle point of standard GPRS (General Packet Radio Service) network. As it is near to the edge, this trace contains a high proportion of symmetric traffic. It was recorded on 2010.

TABLE I
Traces properties

	Duration	Size (MB)	Packets	Flows
ISP_Core	38,160 s	2,591,636	7,074,618,384	295,729,886
ISP_Mob	2,700 s	133,046	233,359,695	6,093,604

IV. Methodology

The main tool employed in this study is based on the Ipoque's PACE engine [2]. It is a commercial DPI library with nearly 100% detection rate with no false positive. We use it to group the protocols according to a general classification which simplifies the evaluation. However, it does not mean that the behaviour of the protocols inside any group is homogeneous. The groups considered are the following: `generic`, `p2p`, `gaming`, `tunnel`, `voip`, `im`, `streaming`, `mail` `network_management`, `filetransfer` and `web`. The `generic` group is mainly composed by unknown traffic. A detailed classification can be found in [8].

We have developed a module over PACE that calculates different statistics by group of applications. The most important for our purpose are the times between packets, named PCK-PCK, and the times between packets with data, named DAT-DAT. The PCK-PCK times include the TCP's acknowledgement packets (ACK) while the DAT-DAT times include only the packets with application data. All the evaluations are done both bidirectionally and unidirectionally (considering each of the Client-Server and Server-Client direction). The establishment of Client-Server or Server-Client direction is handled by PACE. If it is not able to detect it, or the direction is either Client-Client or Server-Server, we assume that the first packet seen comes from the client. In addition, we also study the termination processes of the different groups for the TCP traffic. The terminations detected are the standard FIN (Finalize) process, the RST process, the RST process in a blocking scenario and the undefined. These termination modes are further described in section V-A2.

To avoid the inclusion of worthless data we take the following considerations. For TCP traffic are only parsed those flows showing the first step of the so called 3 way-handshake. That means the SYN (Synchronize)-SYNACK process. Doing so, we avoid considering ongoing flows already initiated and we only consider bidirectional flows. For UDP (User Datagram Protocol) traffic the only condition is to see at least two packets (otherwise any PCK-PCK time could not be calculated). In Table II is shown the impact of applying these considerations on each trace.

TABLE II
Flow usage

	TCP	UDP	TCP used	UDP used
ISP_Core	159,444,165	127,930,249	42,521,933	55,182,658
ISP_Mob	3,904,103	2,063,391	3,454,210	1,850,579

V. Results

The results are presented in two parts. First, the results observed for the PCK-PCK and DAT-DAT times and the proportion of the different types of flow termination are discussed. Second, using these previous results, a set of timeouts by application group are derived for an efficient flow expiration. Additionally, we show in Table III the traffic type proportions of the traces. The heterogeneous composition of these two traces from completely different scenarios supports the representativeness of the results. The excess traffic outside the already cited protocol groups (see section IV) is grouped as `other`.

TABLE III
Flow proportions

protocol	TCP Core	TCP Mob	UDP Core	UDP Mob
generic	12.57%	10.02%	14.16%	13.22%
p2p	5.98%	2.57%	13.81%	13.21%
gaming	0.02%	0.00%	0.03%	0.08%
tunnel	10.66%	9.29%	0.02%	0.30%
voip	0.84%	0.07%	1.94%	1.05%
im	10.60%	0.46%	0.35%	0.05%
streaming	1.07%	0.71%	58.33%	0.42%
mail	14.17%	1.50%	0.00%	0.00%
management	0.01%	0.01%	11.35%	69.93%
filetransfer	0.69%	0.30%	0.00%	0.00%
web	42.98%	74.69%	0.00%	0.00%
other	0.41%	0.38%	0.01%	1.74%

A. Time profiled study

1) Inter-packet times evaluation: The results related to the DAT-DAT and greater extent the PCK-PCK times (see section IV) are fundamental to establish the flow termination timeouts

according to our proposed method, as further explained in section V-B. In this section, some interesting outcomes are extracted from the statistics obtained.

- **Similar PCK-PCK times over TCP regardless of the scenario:** Excluding the `network_management` group, there is a strong correlation comparing the PCK-PCK values between the `ISP_Core` (see Table IV) and `ISP_Mob` (see Table V). This is also accomplished for the DAT-DAT values, but not for the UDP traffic (which is always DAT-DAT). This result suggests that the average packet cadence over TCP is independent on the network.
- **PCK-PCK similarities in Client-Server and Server-Client:** In both scenarios and for the TCP (see Tables IV and V) and the UDP (see Table VI) traffic there are similar PCK-PCK values comparing the Client-Server and Server-Client directions. In the UDP case, it could be expected to see an asymmetric behaviour in some protocol groups like the `streaming` as it happens in the DAT-DAT values over TCP (see Table VII). This highlights the different role that some protocols take over UDP.
- **TCP and UDP usage differences:** UDP flows for some protocol groups in the `ISP_Core` show (see Table VI) a lower or higher packet transmission rate than the DAT-DAT seen over TCP (see Table VII), or as just explained for the `streaming` group they have an unexpected behaviour. That is justified by the usage of control flows over TCP or UDP by some protocols. This behaviour is seen in `p2p`, where the DAT-DAT times are around 4 times lower in the TCP case (probably real data transfer) than in the UDP (probably control flows) one. It also occurs but in the opposite way for the `gaming`, `tunnel`, `voip` and `im` groups, which would imply that now the real stream of data is sent over UDP while the TCP flows are the control ones. This behaviour is similar in the `ISP_Mob` trace, with the main exception seen for the `streaming` group since the UDP traffic seems to carry as well real traffic (low value for the PCK-PCK times).

TABLE IV
PCK-PCK TIMES TCP `ISP_CORE` (ms)

group	avg	std	avg c-s	std c-s	avg s-c	std s-c
generic	1,510	12,860	2,760	17,840	2,720	16,740
p2p	1,196	7,084	2,216	9,201	2,217	9,593
gaming	209	1,319	448	1,777	384	1,732
tunnel	1,029	11,080	2,004	15,412	1,737	14,706
voip	1,688	9,981	3,161	13,677	3,450	14,137
im	2,715	13,356	5,240	17,857	5,302	18,390
stream	120	3,200	286	4,842	181	3,871
mail	518	8,259	802	10,435	1,040	11,959
manage	1,494	12,774	2,430	17,382	3,313	19,552
filetx	390	36,037	855	55,946	647	46,807
web	1,127	13,383	2,294	18,757	1,827	17,586

2) Flow termination evaluation: The study of the different flow termination types is carried out for the TCP traffic given its different termination mechanisms. Four possible terminations are observed: TCP handshake, RST, undefined termination and RST in a blocking scenario. Fig. 1 shows the behaviour of a RST termination in a blocking scenario. That is when one or both extremes of the communication do not receive the corresponding packets from the other side. Unlike we expected, as we can see in Fig. 2, the number of flows not finished by the standard TCP handshake is very high. Additionally, there is a big amount of flows terminating with a RST in a blocking scenario. In these situations the retransmission times grow exponentially [9], [10]. Thus, we have particularly studied the times for the flows finished with a RST in a blocking scenario (i.e., time between a RST and the last packet of the flow, and PCK-PCK time). They are shown in Table VIII. Notice that these PCK-PCK times are generally bigger than the standard ones (see Table IV). The results for the `ISP_Mob` are very similar and are omitted for the sake of space. These results can find a small comparison basis in [3].

TABLE V
PCK-PCK TIMES TCP `ISP_MOB` (ms)

group	avg	std	avg c-s	std c-s	avg s-c	std s-c
generic	1,796	14,293	3,043	18,868	3,526	19,840
p2p	1,907	8,660	3,475	11,648	3,579	13,058
gaming	109	824	139	946	473	2,520
tunnel	1,396	20,782	2,778	29,860	2,414	27,576
voip	1,976	11,315	3,744	15,264	3,892	15,843
im	1,897	15,667	3,587	21,663	3,894	22,373
stream	87	2,269	216	3,564	117	2,870
mail	1,057	13,678	2,135	20,005	1,957	18,696
manage	316	4,374	604	6,055	682	6,527
filetx	427	9,819	700	13,992	808	14,469
web	704	7,349	1,364	9,793	1,117	8,861

TABLE VI
PCK-PCK TIMES UDP `ISP_CORE` (ms)

group	avg	std	avg c-s	std c-s	avg s-c	std s-c
generic	2,878	23,551	3,427	25,744	4,719	27,850
p2p	17,310	64,100	26,099	77,177	18,017	70,828
gaming	199	2,997	295	3,816	374	3,948
tunnel	454	7,215	572	8,284	1,061	12,091
voip	306	7,888	509	10,211	404	10,430
im	151	1,471	209	1,829	198	3,171
stream	4,375	37,134	7,068	47,118	5,846	43,562
manage	11,161	55,806	20,342	73,220	21,947	79,760

B. Estimation and evaluation of profiled timeouts

A flow timeout has to be big enough to not consider new flows with a late packet of an existing one, but as low as possible to release the memory the sooner. An old previous study already defined the use of a global timeout of 64 seconds [5]. In order to address this problem we have found a way to establish it according to different protocol groups by using the statistics obtained, basically the PCK-PCK times. The usage of the inter-packet times for the termination timeout lies on what we want to know is the expected time for a packet

TABLE VII
DAT-DAT TIMES TCP ISP_CORE (ms)

group	avg	std	avg c-s	std c-s	avg s-c	std s-c
generic	2,137	14,585	8,673	41,852	2,566	16,870
p2p	1,772	8,077	3,763	14,747	2,741	11,162
gaming	372	1,521	1,020	4,274	574	2,271
tunnel	1,334	13,057	3,465	24,198	1,584	23,618
voip	2,656	12,609	5,571	19,021	4,825	18,193
im	4,321	16,135	12,360	37,903	6,167	38,115
stream	136	2,960	3,230	18,987	137	3,055
mail	746	10,423	826	11,717	1,945	19,036
manage	1,278	10,442	7,825	33,821	2,634	13,823
filetx	533	46,120	2,944	130,181	589	49,281
web	880	9,269	4,714	24,456	911	9,843

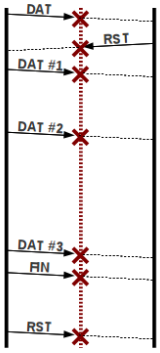


Fig. 1. Blocking scenario behaviour. Both sides send data, but since there is no acknowledgement from the other side they try to retransmit (after a time which grows in every attempt) until finally break the connection with a RST

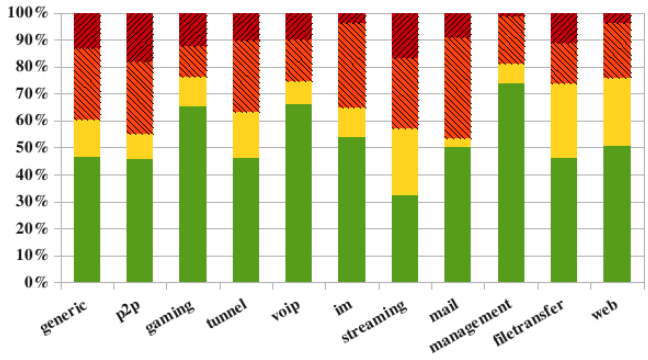


Fig. 2. Termination proportions ISP_Core, green stands for the standard FIN process, yellow for the unclosed flows, red for the RST and intense red for the RST in a blocking scenario

TABLE VIII
RST TIMES ISP_CORE (ms)

group	avg rst-end	std rst-end	avg pck	std pck
generic	34,276	138,999	6,722	33,147
p2p	15,095	53,451	6,354	18,664
gaming	57,834	128,352	765	5,673
tunnel	13,618	96,933	5,618	33,423
voip	20,311	91,428	4,753	29,569
im	8,819	68,851	1,964	17,671
stream	7,449	50,738	344	4,060
mail	29,384	126,616	9,357	38,504
manage	182	259	157	239
filetx	11,967	86,210	3,768	45,231
web	11,534	59,228	1,996	13,051

to arrive. For TCP traffic, we also consider the times regarding the PCK-PCK after a RST in a blocking scenario (see Table VIII) and the proportion of the different terminations. They are important since in these situations the time of inactivity grows exponentially and forces a high timeout.

We establish a threshold selecting the value $\{average + 2.5 * STD\}$[1]. Doing so, unusual big measures are excluded, while still representing around 99% of the complete time values. However, this does not mean that 99% of the flows are not segmented. Such proportion is difficult to evaluate, although is always lower since each flow is composed by many packets. This adds some issues to our evaluation and is further discussed later on (see explanation refering to Fig. 4). For each trace and separately for the TCP and UDP flows we choose the biggest threshold from either the Client-Server or the Server-Client direction. Thereby, we consider always the worst case. We define these thresholds as $\widehat{t_{pck\_pck}}$ for the PCK-PCK standard times and $\widehat{t_{pck\_blk}}$ and the PCK-PCK times after a RST in a blocking scenario.

We have seen that $\widehat{t_{pck\_blk}}$ times are forcing a higher value when considering the optimal timeout. For that reason we use those values to round off the final results. The criteria used is based in two aspects. On the one hand, we consider which proportion of the traffic is behaving as in a blocking scenario. On the other, we also evaluate the CDF (Cumulative Distribution Function) of the PCK-PCK times in relation to the timeout for each group to see what is its variability. That is, what is the effect of reducing the timeout in the proportions of values inside the $\widehat{t_{pck\_blk}}$ interval. Fig. 3 shows the three groups we have detected. A strong variation is seen for the p2p, voip, streaming and network_management groups. A medium variation for the generic, tunnel, im, mail and web groups. Finally, a slow variation is seen for the gaming and filetransfer groups.

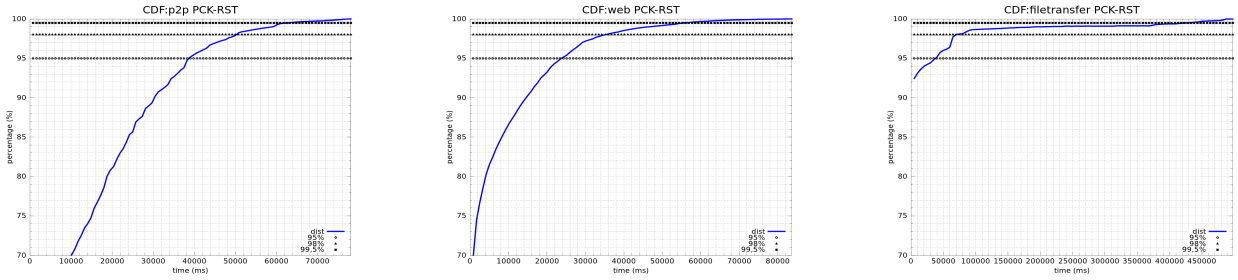
Although it is difficult to include these behaviours in the calculation of the final timeout, we have done it applying a factor named F. It modifies the final timeout taking into account the behaviour above described. This factor is set up as $\{0.33, 0.66, 1\}$ for the slow, medium and strong variation respectively. Thus, we propose that for TCP traffic a good theoretical timeout could be approximated as follows:

$$t_{timeout} = (1 - fin_{blk}) * \widehat{t_{pck\_pck}} + fin_{blk} * \widehat{t_{pck\_blk}} * F \quad (1)$$

$$F = \{0.33, 0.66, 1\} \quad (2)$$

Remember the $\widehat{t_{pck\_blk}}$ regards the RST times in a blocking

[1]STD = Standard deviation



(a) Strong variation (P2P) (b) Medium variation (Web) (c) Slow variation (Filetransfer)

Fig. 3. CDF of inter-packet times with RST termination in a blocking scenario

scenario. Therefore, the proportion fin_{blk} is as well related to that field, but concerning the proportion of termination modes. Both parameters are considered getting the worst case value among the `ISP_Core` and the `ISP_Mob` (i.e., the case where the timeout is bigger), so we intend to find out a global timeout indepedent of the traces evaluated. By applying this formula we consider the $\widehat{t_{pck\_blk}}$ according to its proportion of appearance in terms of flows termination and also the behaviour seen from the CDF's. The parameters used for this calculation and the results are shown in Table IX. For UDP traffic no rounding off has been done since the only parameter to consider is the $\widehat{t_{pck\_pck}}$.

TABLE IX
TIMEOUT TCP (ms)

group	$\widehat{t_{pck\_pck}}$	$\widehat{t_{pck\_blk}}$	fin_{blk}	F	$t_{timeout}$
generic	53,126	117,269	0.1326	0.66	56,344
p2p	36,255	78,086	0.181	1	43,826
gaming	6,772	26,440	0.1246	0.33	7,015
tunnel	77,426	119,298	0.1244	0.66	77,589
voip	43,500	90,671	0.1008	1	48,255
im	59,826	163,225	0.0787	0.66	63,596
stream	12,392	35,098	0.3653	1	20,687
mail	52,147	127,211	0.1011	0.66	55,363
manage	52,192	76,480	0.1018	1	54,665
filetx	140,720	497,736	0.291	0.33	147,568
web	47,804	83,657	0.0428	0.66	48,121

TABLE X
TIMEOUT UDP (ms)

group	$t_{timeout}$
generic	74,343
p2p	219,043
gaming	45,584
tunnel	40,814
voip	71,636
im	14,030
streaming	124,862
management	221,346

In addition, we have gone further and we have evaluated the impact of these outcomes by means of the Ipoque's PACE engine [2]. Thanks to that, we are able to measure not only the memory saving but also the detection rate impact. However, due to technical limitations in the PACE engine we have not been able to set up each timeout in a profiled way for each protocol group.

To solve that issue, we have elaborated a method to see what is the effect of varying the timeout on each group. We measure for a set of intervals from 20 to 300 seconds of timeout, in steps of 20 seconds, what are the number of flows detected by PACE according to each protocol group. We also add the values for 600 seconds to obtain a wide timeout to compare with. Then we have plot (see Fig. 4) the proportion of flows per protocol group for each timeout compared with the maximum number of flows seen (when the timeout is 20 seconds). Thus, we get a curve to approximately see when the number of flows per protocol group tends to be stable (see Fig. 4). Only the `ISP_Core` trace is evaluated since it is more representative than the `ISP_Mob` one, which results show similarities.

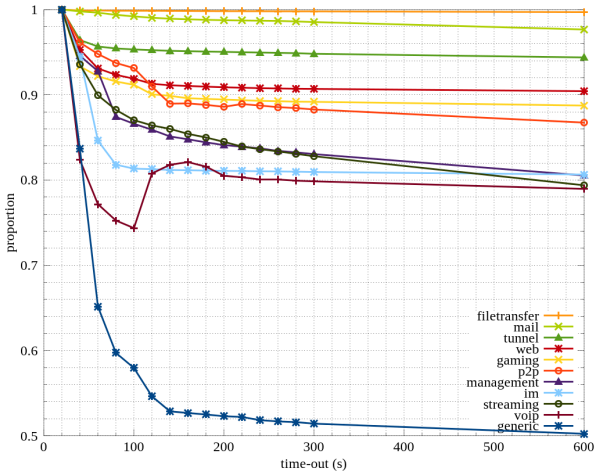


Fig. 4. Timeout PACE evaluation `ISP_Core`

Consider that for the FTP (File Transfer Protocol) inside the `filetransfer` group the timeout is directly established

by PACE and its operation is independent from the one we vary. That is why in Fig. 4 it is not seen a wide variation for this group. Also, the effect of the `voip` and lesser extent in the `p2p` group, since at some point the variation increases, hindering the evaluation of our theoretical timeouts. This happens due to internal detection methods of PACE. With the exception of the `generic` group (remember that it contains the unknown traffic), the already commented situations and considering that the evaluation is done over all the traffic together (i.e., TCP and UDP), the theoretical results are roughly accomplished. However, for some protocol groups the curves seems to not stabilize. For `p2p` and `network_management` this behaviour is mainly due to a high timeout seen for UDP traffic, which is majority against TCP (see Table III). In lasting flows with many packets, like `streaming`, the effect of having around 1% of PCK-PCK times outside the threshold results in a bigger number of flows seen. It is not the same having 100 flows with 10 packets than 10 flows with 100 packets. In both situations there would be around 1000 PCK-PCK times, which means that around 10 of them would be outside the threshold. In the first situation it would result in having around 110 (10% increment) flows while in the second around 20 (100% increment).

More precisely we have also checked a global timeout by measuring what is the direct impact in the memory requirements, the detection rate and the performance of PACE. We can see that parameters in Fig. 5 where the undetection rate is the global one, the memory elements are the number of flows which are stored for each timeout and the total elements stand for the number of flows seen for each timeout. For a certain timeout both the detection rate and the total number of flows (which is directly related to the performance) tend to be stable. Comparing a timeout in which the detection rate and number of flows stabilizes (approximately 200 seconds) with 600 seconds (this was the reference timeout implemented by PACE), we achieve a reduction of around 60% of memory necessities.

VI. Conclusion

Nowadays Internet bandwidth has forced online traffic classifiers to be implemented very efficiently. Among the different constraints that online traffic classification present, this paper focuses on the optimization of the memory consumption. A key issue in this aspect is the expiration of the structures that keep the state of each flow. This paper proposes a new methodology for an efficient expiration of the flows based on profiled timeouts. In order to obtain these timeouts it has been studied the behaviour of the traffic by group of applications. Unexpected results have been observed regarding the proportion and types of flow termination in the current traffic. The profiled timeouts obtained have been evaluated in a well-known commercial DPI tool (the Ipoque's PACE engine), achieving a drastic reduction of memory while keeping the computation cost and classification accuracy. The results suggest that the implementation of the profiled timeouts in the traffic classification techniques proposed in the literature [11] could considerably improve their memory consumption, alleviating by this, their feasibility for online classification.

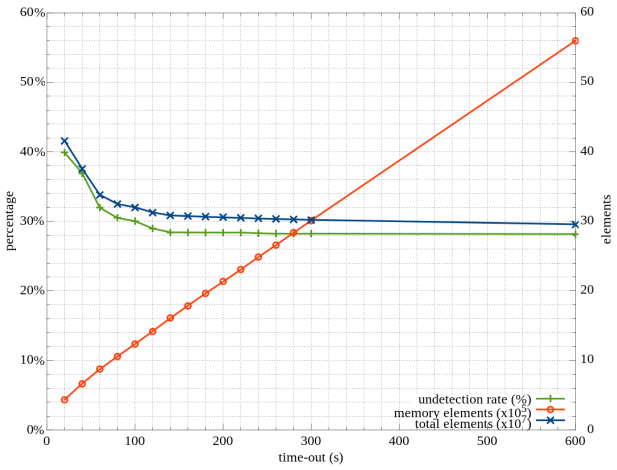


Fig. 5. Timeout PACE performance `ISP_Core`. Elements stand for flows

Acknowledgment

This research was partially funded by the Spanish Ministry of Science and Innovation 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).

References

[1] A. Dainotti, A. Pescapè, and K. Claffy, "Issues and future directions in traffic classification," *Network, IEEE*, vol. 26, no. 1, pp. 35–40, 2012.

[2] Ipoque, "Datasheet PACE," Ipoque GmbH, 2012, website: http://www.ipoque.com/sites/default/files/mediafiles/documents/data-sheet-pace.pdf.

[3] Wolfgang John, Sven Tafvelin, Tomas Olovsson, "Trends and Differences in Connection-behavior within Classes of Internet Backbone Traffic," Chalmers University of Technology, 2008, website: http://www.sjalander.com/wolfgang/publications/PAM08.pdf.

[4] H. Kim, J.-H. Kim, I. Kang, and S. Bahk, "Preventing session table explosion in packet inspection computers," *Computers, IEEE Transactions on*, vol. 54, no. 2, pp. 238–240, 2005.

[5] Kimberly C. Claffy, Hans-Werner Braun, George C. Polyzos, "A Parameterizable Methodology for Internet Traffic Flow Profiling," pp. 1481–1494, October 1995.

[6] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," in *Proc. of ACM CoNEXT, December*, 2008.

[7] V. Carela-Espanol, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Sole-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification," *Computer Networks 55*, pp. 1083–1099, 2011.

[8] Ipoque, "SUPPORTED PROTOCOLS AND APPLICATIONS," Ipoque GmbH, 2012, website: http://www.ipoque.com/sites/default/files/mediafiles/documents/data-sheet-protocol-support.pdf.

[9] R. Braden, "RFC 1122 - Requirements for Internet Hosts – Communication Layers," Internet Engineering Task Force, October 1989, website: http://tools.ietf.org/html/rfc1122.

[10] V. Paxson, M. Allman, J. Chu, M. Sargent, "RFC 6298 - Computing TCP's Retransmission Timer," Internet Engineering Task Force, June 2011, website: http://tools.ietf.org/html/rfc6298.

[11] T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 4, 2008.