Computational Weight of Network Traffic Sampling Techniques

João Marco C. Silva, Paulo Carvalho and Solange Rito Lima
Centro Algoritmi, Universidade do Minho, Braga, Portugal
Email: joaomarco@di.uminho.pt, pmc@di.uminho.pt, solange@di.uminho.pt

***Abstract*—Within network measurement context, traffic sampling has been targeted as a promising solution to cope with the huge amount of traffic traversing network devices as only a subset of packets is elected for analysis. Although this brings an evident advantage to measurement overhead, the computational burden of performing sampling tasks in network equipment may overshadow the potential benefits of sampling. Attending that sampling techniques evince distinct temporal and spatial characteristics in handling traffic, this paper is focused on studying the computational weight of current and emerging techniques in terms of memory consumption, CPU load and data volume. Furthermore, the accuracy of these techniques in estimating network parameters such as throughput is evaluated. A sampling framework has also been implemented in order to provide a versatile and fair platform for carrying out the testing and comparison process.**

I. Introduction

Performing network measurement tasks in today's networks is a continuous challenge attending to the massive traffic volumes involved, to the wide range of possible monitoring objectives to fulfill, sometimes in a near real-time basis and requiring minimal interference in the network operation. Aiming at efficient network measurements, traffic sampling techniques are broadly deployed in strategic network nodes, generically called measurement points. Their main objective is to select a subset of packets which will be used to estimate network parameters, avoiding processing all traffic [1].

Despite the efforts in developing sampling techniques able to capture network parameters accurately, studying the impact those techniques have in terms of computational weight is a topic still open. This study is however relevant as measurements should be carried out without compromising the performance of network elements when executing tasks involving end-user traffic. Thus, taking conventional sampling techniques commonly used in today's network devices [1] and recent adaptive sampling techniques [2] as reference, this paper is devoted to evaluate the computational burden of sampling regarding memory and CPU usage in presence of real network traffic. The volume of sampled traffic and the accuracy of these techniques in the estimation of parameters such as throughput and mean packet size are also analyzed.

The methodology of tests resorts to a sampling framework developed with the purpose of implementing different sampling techniques in a flexible way to allow the combination of their inner characteristics in forthcoming operational scenarios. Using a low-cost open computing device currently deployed in measurement architectures [3], this paper presents quantitative results comparing the computational overhead of multiple sampling techniques in presence of similar workloads. This study aims to contribute to a better understanding of the current weight of sampling procedures, allowing to identify the most suitable sampling techniques for using in low-cost and scalable passive measurement solutions. In addition, this allows to extend the applicability of measurement nodes proposed in [3], where a large scale and inexpensive active measurement infrastructure is deployed.

This work is organized as follows: the related work is discussed in Section II; the traffic sampling framework is introduced in Section III; the methodology of tests is presented in Section IV; the computational weight results are discussed in Section V; and the conclusions are included in Section VI.

II. Related work

Currently, traffic sampling sustains a wide range of network tasks. For instance, its usefulness has been explored in: *traffic engineering* to assist traffic classification and characterization [4]; *network security* for anomaly and intrusion detection, botnet and DDoS identification [5]; *SLA compliance* and *QoS control* for estimating parameters such as packet delay, jitter and loss [6], [7], [8]. Despite the importance of sampling to reduce the computational effort of handling huge amount of data, most works in this area are only focused on analyzing the accuracy of traffic parameter estimation. The resource requirements of sampling techniques are covered through proposals which aims to reduce the memory usage [9] and the data volume involved in measurement process [2], maintaining the accuracy in traffic classification. Classic techniques [1] are also analyzed in terms of CPU load and memory usage in dedicated equipments [10], however there are no embracing studies analyzing and comparing the computational weight of existing sampling approaches, namely *i.e., systematic, random and adaptive* (see Section III), motivating the present work.

III. Traffic sampling framework

Network sampling techniques may be structurally classified according to three well-defined components, *i.e.*, *granularity*, *selection scheme* and *selection trigger* (see Figure 1). Each component is further divided into a set of approaches able to encompass both classic and recent sampling techniques:

- *Granularity* identifies the atomicity of the element under analysis in the sampling process: a flow-level approach

consists in applying the traffic capture policy only to packets belonging to a flow or a set of flows of interest; in a packet-level approach, packets are eligible as single independent entities.

- *Selection scheme* identifies the function defining which traffic packets will be selected and collected; this follow a systematic, random, or adaptive function. In the systematic approach, the packet selection is ruled by a deterministic function. The random approach selects packets according to a random or probabilistic function [1]. In the adaptive approach, the sampling technique is endowed with the ability to change the selection of packets during the course of measurement [11].
- *Selection trigger* is used to decide the spatial and temporal sample boundaries. It may use a time-based, a count-based or an event-based approach.

Following this taxonomy, firstly introduced in [12], a framework implemented in Java using Jpcap, connects the sampling components in order to enable a versatile deployment of sampling techniques. This framework is designed in two planes comprising the relationship among the sampling components, as illustrated in Figure 1, and may be applied to both online and offline measurement scenarios.

The sampling plane has a modular design, allowing a flexible sampling technique selection and configuration. This plane is also responsible for identifying and selecting the network interface in which the sampling will be applied.

In the network plane, traffic is collected from network interfaces by applying the sample rules defined in the sampling plane. Then the collected packets are reported to be analyzed according to network task measurement needs.

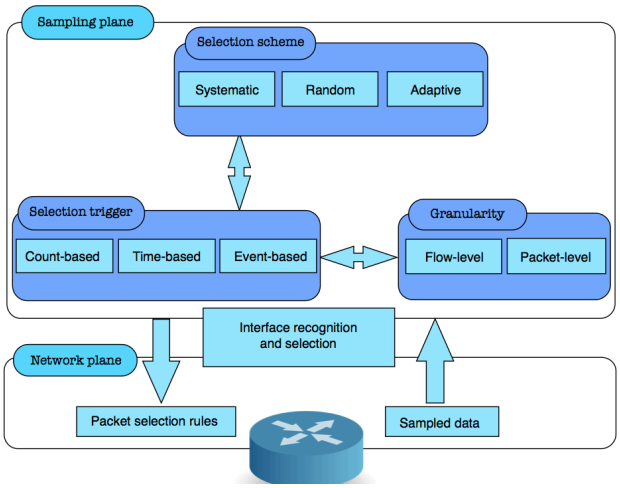


Fig. 1. Framework design

IV. Methodology of Tests

The computational weight of network sampling techniques is analyzed regarding the *memory consumption*, *CPU load* and *volume of data* involved in the sampling process. For this, a set of real traffic traces with different load features is applied to classic and emerging traffic sampling techniques. In addition to the comparison of the computational weight, the sampling techniques are also compared regarding their ability to identify traffic characteristics. These aspects are detailed below.

To perform tests, the sampling framework was deployed and installed in a general purpose low-cost single-board computer, *i.e.*, Raspberry Pi Model B, running a ARM processor at 700MHz and 512MB RAM. This type of equipment is taking ground in network monitoring context, namely in large scale measurement architectures [3].

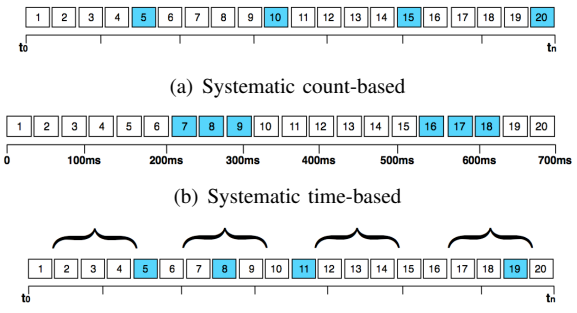
A. Sampling techniques

The sampling techniques evaluated correspond to the main techniques currently used in network measurement tools, *i.e.*, *systematic count-based*, *systematic time-based* and *random count-based* [1]. In addition, two adaptive techniques are also evaluated, *i.e.*, *adaptive linear prediction* [13] and *multiadaptive sampling* [2].

1) Systematic count-based (SystC): drives the packet selection through a deterministic and invariable function based on the packet position, using counters [1]. As exemplified in Figure 2 (a), every 5th packet is selected and captured by the sampling process.

2) Systematic time-based (SystT): the process of the packet selection follows a deterministic function based on the arrival time at the measurement point [1]. In this technique the sample size and the time between samples are set at the beginning and remain unchanged along the sampling process, as presented in Figure 2 (b). As shown, all packets arriving at the measurement point along a period of 100ms are selected for a sample, whereas all incoming packets along 200ms are ignored for measurement purposes.

3) Random count-based (RandC): selects the starting points of the sampling intervals in accordance with a random process. As presented in Figure 2 (c), in the *n-out-of-N* random approach, n elements are randomly selected out of the parent population that consists of N elements [1].



(a) Systematic count-based

(b) Systematic time-based

(c) Random count-based (1-out-of-5)

Fig. 2. Examples of sampling techniques

4) Adaptive linear prediction (LP): this time-based technique uses linear prediction to identify changes in the network activity, adjusting the sampling frequency accordingly while the sample size remains invariant [13]. Its basic operation consists of increasing the sampling frequency, *i.e.*, reducing

the interval between samples, when more network activity than predicted is observed in order to identify the new traffic pattern. Conversely, when less network activity than predicted is observed, the interval between samples is increased, reducing the sample frequency and, consequently, the amount of data involved in the sample process.

5) Multiadaptive sampling (MuST): this time-based technique resorts to a mechanism for identifying the level of network activity similarly to the adaptive linear prediction technique. However, the multiadaptive technique considers both the interval between samples and the sample size as adjustable parameters [2].

Apart from increasing the sampling frequency in periods of more activity than predicted, the multiadaptive technique also reduces the sample size, avoiding the overload of the measurement point in a critical scenario of its operation. Conversely, in periods of less activity than predicted, in addition to sampling frequency reduction, the multiadaptive sampling also increases the sample size in order to acquire more information about the network without the risk of overloading the measurement point. Previous work has shown the ability of this technique in capturing correctly the network throughput with very low overhead as regards to the volume of data involved [2].

B. Comparative parameters

The computational weight of each sampling technique is analyzed in terms of hardware resource usage and volume of sampling data stored. While resource usage may impact the performance of the measurement point, data volume affects the bandwidth required by measurement data as well as the storage and processing overhead [10]. The main computational resources observed along the sampling process are *CPU load* and *memory usage*, analyzed through *vmstat*, a computer monitoring tool able to collect summary information about the operating system activity on a near real-time basis. The *volume of data* corresponds to the sum of all packets collected by each sampling technique, using the total length field within IP header.

Considering that in traffic sampling only a subset of total network packets is captured and considered for measurement purposes, the computational weight should be balanced with the correctness in estimating traffic behavior. A common way to achieve this goal is estimating throughput, measuring the amount of sampled data in a time interval. Thereby, the accuracy in estimating traffic behavior is analyzed through *instantaneous throughput*, *i.e.*, the throughput estimated in each sample along the measurement process, and *mean throughput*, *i.e.*, the total estimated load, and its mean relative error (MRE). In addition, the accuracy in instantaneous throughput estimation is measured using the variance, where a smaller variance means a more accurate estimation. This comparison is performed resorting to the mean square error (MSE), a common metric to compare estimators [10]. Furthermore, the mean packet size and complementary descriptive statistics to measure the variability of packet time series, *i.e.*, the ratio between peak and average packet size, are also analyzed.

C. Traffic scenarios

The traffic scenarios used in the analysis correspond to three workload periods (low, moderate and high) in the network backbone of the University of Minho campus along a typical workday. Initially, only *https* traffic was collected, then submitted to each sampling technique. This traffic type leads to a significant trace for analysis and allows to keep the privacy of users data. The measured values for each traffic load scenario are presented in Table I.

TABLE I
TRAFFIC SCENARIOS

Workload scenario / Feature	Low	Moderate	High
Number of packets	311159	1273068	1718804
Volume of data (MBytes)	112.04	712.99	1063.16
Mean throughput (Mbps)	3.90	26.65	68.79
Mean packet size (Bytes)	377.58	587.26	648.59

V. EVALUATION RESULTS

This section includes the main test results evaluating the computational weight of the sampling techniques described in Section IV. After performing initial tuning of count and time-based techniques to assess the impact of sampling frequency, the discussion is focused on evaluating CPU load, memory consumption, volume of sampled data and accuracy of the sampling techniques under study.

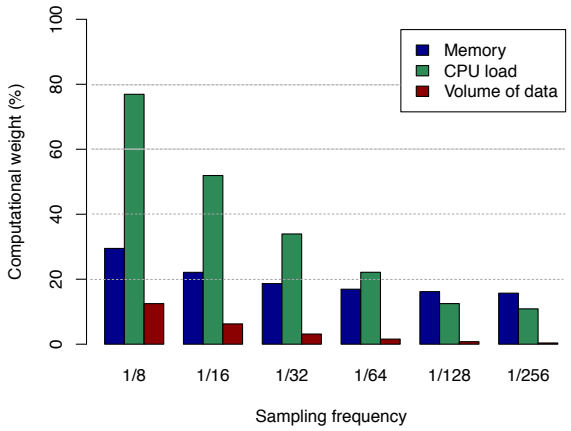
A. Tuning of systematic techniques

Considering that the computational resources consumption of systematic techniques is empirically proportional to the sampling frequency, Figure 3 shows the difference of the mean computational consumption for distinct sampling frequencies of the systematic techniques when applied to the high workload scenario. For SystC (Figure 3(a)), a frequency of 1/8 means that every eighth packet is collected at the measurement point, while for SystT (Figure 3 (b)), a frequency of 100/500 means that all incoming packets are collected along 100ms periods, and ignored in the following 400ms periods.

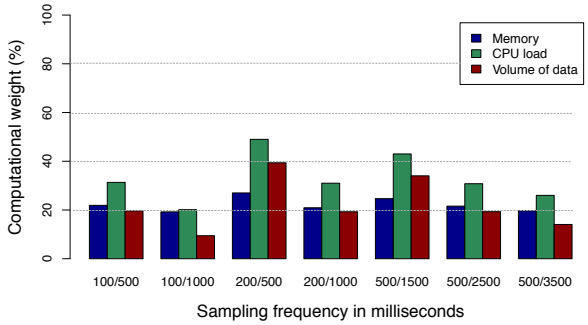
Figure 3 confirms the relation between sampling frequency and computational weight, although the CPU load presents a greater variation than memory usage for both systematic techniques. In SystC (Figure 3(a)), the computational weight exhibits a smoother variation as the sampling frequency decreases, indicating a minimal demand in terms of resources. In SystT (Figure 3(b)), besides the relation with the sampling frequency, the computational weight varies for different frequencies with the same sampling ratio, *i.e.*, 100/500, 200/1000 and 500/2500.

In addition, Figure 3 also demonstrates the relation between sampling frequency and volume of data processed and stored by the systematic techniques. Although SystC is more widely used than SystT, SystT requires less computational resources when considering an equivalent volume of sampled data (*e.g.*, for sampling frequencies of SystC 1/8 and SystT 500/3500).

Despite of the values presented in Figure 3, the following comparative evaluation uses the frequency 1/100 for SystC and



(a) SystC



(b) SystT

Fig. 3. Systematic techniques comparison - High workload

RandC techniques, as suggested in [14]. For SystT technique, the sampling frequency in use is 100/1000 as it led to the best results for the analysis performed.

B. CPU load

Regarding the comparative analysis of all sampling techniques deployed in the framework, Figure 4 presents the CPU load along the sampling process of high workload traffic. As shown, the LP technique clearly requires more CPU resources than the other techniques. This occurs because this technique requires processing all packets to analyze the evolution of traffic activity based on accumulated data, even of packets not collected. Conversely, the MuST technique requires the lowest CPU usage, confirming its main goal and ability to reduce the resource consumption during high activity periods [2]. This aspect is particularly relevant for high-load, high-speed networks.

The complete comparison of the average CPU load is presented in Table II. As shown, SystC and RandC techniques outperform MuST for low workload scenarios. This is due to the impact of the adaptive process in self-adjusting facing workload variations. In addition, the difference in CPU load of count-based techniques (SystC and RandC), for the same sampling frequency, is due to the additional cost of the random function in RandC.

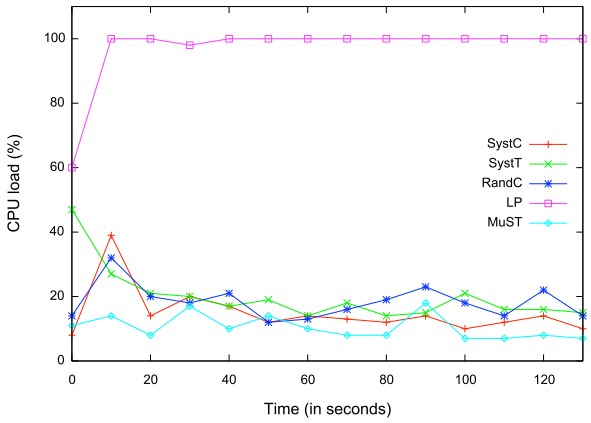


Fig. 4. CPU load - High workload

C. Memory usage

Figure 5 illustrates the memory usage by each sampling technique along the sampling process for the high workload scenario. In this case, the SystT technique requires the highest amount of memory, for all traffic scenarios considered, as presented in Table II. The SystC technique demands the lowest memory amount, although none technique has incurred in a significant resource consumption, as observed in the CPU load analysis. Regarding MuST, Table II also ratifies its ability to reduce the memory consumption during high activity periods, as observed in the CPU load analysis.

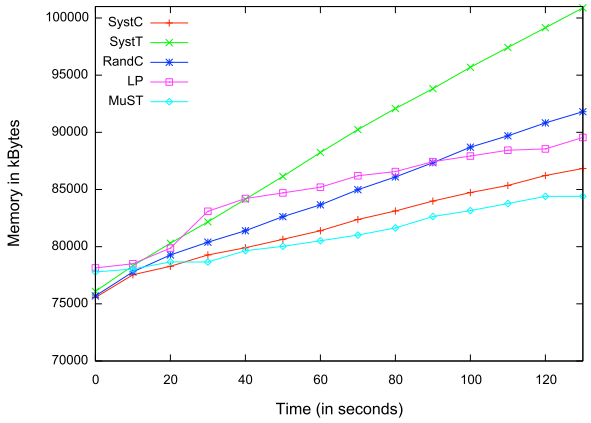


Fig. 5. Memory usage - High workload

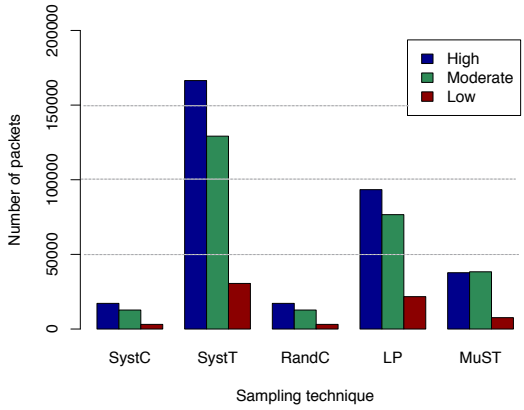
D. Volume of Data

Regarding the volume of data collected and stored along the sampling process, the count-based techniques, *i.e.*, SystC and RandC, demonstrate less use of resources, as illustrated in Figure 6. Figure 6(a) represents the total number of packets collected by each technique and Figure 6(b) the sum of the

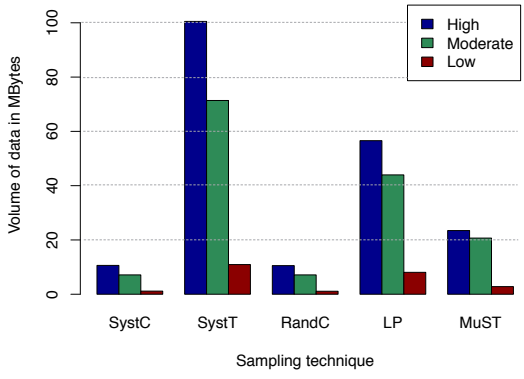
TABLE II
AVERAGE USE OF COMPUTATIONAL RESOURCES

Parameter	SystC	SystT	RandC	LP	MuST
Low workload					
CPU load (%)	5.03	14.55	5.50	27.35	8.82
Memory (kBytes)	76566	95900	81222	82440	85295
Moderate workload					
CPU load (%)	10.80	17.95	16.86	96.68	10.72
Memory (kBytes)	80773	96410	84042	87698	84371
High workload					
CPU load (%)	14.92	20.12	18.26	97.27	10.76
Memory (kBytes)	81801	90754	86163	85551	80765

corresponding packet lengths, in Mbytes. As shown, MuST achieves the best results among the time-based techniques.



(a) Number of packets



(b) Volume of data

Fig. 6. Comparative data volume

Although with higher consumption of storage resources, SysT and MuST achieve a better relationship between the volume of data collected and the computational cost involved. This is shown in Table III, in which the ratio of CPU load and memory usage per MByte collected and stored extends the discussion of SectionV-A. This may be an advantage for network activities in which more information about the traffic represents improved results in measurement accuracy (*e.g.*, traffic classification and intrusion detection).

TABLE III
COMPUTATIONAL RESOURCE PER MBYTE - HIGH WORKLOAD

Ratio	SystC	SystT	RandC	LP	MuST
%CPU/MByte	1.40	0.20	1.73	1.71	0.45
%Memory/MByte	1.63	0.16	1.73	0.32	0.72

E. Accuracy

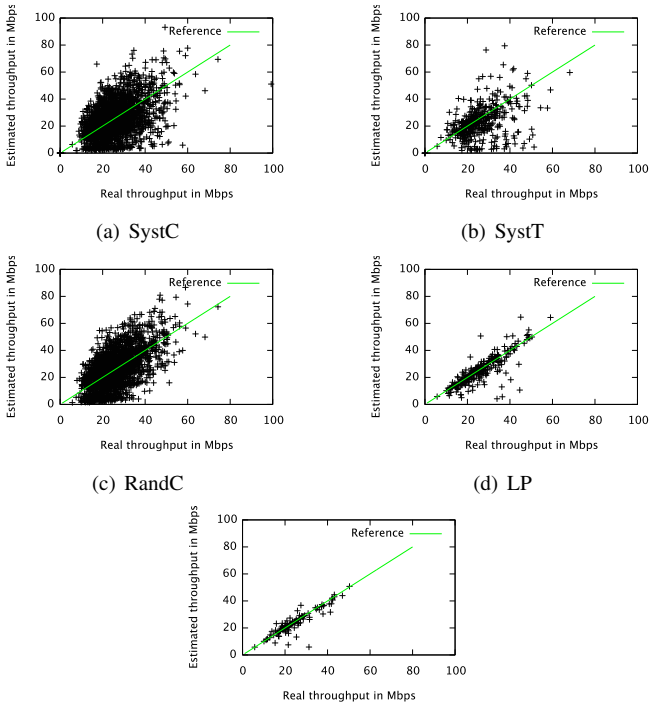
Despite the importance of reducing the consumption of computational resources associated with traffic sampling, the sampling techniques must still be able to represent the network behavior accurately. Table IV presents the mean throughput estimated after each sampling process. For all scenarios, the mean relative error is low (less than 10%). Exceptions are: (i) the MuST technique when applied to the high workload scenario, achieving a significant lower relative error (less than 1%); and (ii) the RandC technique when applied to the moderate workload scenario, with a relative error above 10%. This analysis is particularly useful for activities such as traffic engineering, accounting and SLA compliance, as measurements over large time intervals are considered.

Regarding the MSE of the instantaneous throughput estimation (see Table IV), the LP and MuST techniques are more stable for all traffic load scenarios, indicating high accuracy for short-time measurement intervals. These results are detailed in Figure 7, where each point corresponds to one sample and the values closer to the reference line indicate a lower estimation error and an overall stability of its algorithms.

Analyzing the packet size distribution, Table V shows that all techniques achieve accurate estimations of mean packet size, where the low workload scenario presents the less accurate results. The ratio between peak and average packet size, a descriptive statistics to measure the variability of packet time series, for identifying burstiness, also ratifies the accuracy of all techniques estimation. In this case, the low workload

TABLE IV
THROUGHPUT ESTIMATION

Parameter	Total	SystC	SystT	RandC	LP	MuST
Low workload						
Mean throughput (Mbps)	3.90	3.72	3.70	3.66	3.85	3.81
MRE		0.044	0.049	0.059	0.011	0.021
MSE		1.42	0.84	0.84	0.17	0.38
Moderate workload						
Mean throughput (Mbps)	26.65	25.40	25.08	23.73	25.51	25.44
MRE		0.046	0.059	0.11	0.042	0.045
MSE		1.66	1.59	1.50	0.42	0.21
High workload						
Mean throughput (Mbps)	68.79	65.54	64.06	65.05	64.28	68.47
MRE		0.047	0.068	0.054	0.065	0.004
MSE		0.57	1.95	0.56	0.39	0.44



(a) SystC (b) SystT

(c) RandC (d) LP

(e) MuST

Fig. 7. Dispersion of estimated throughput - Moderate workload

TABLE V
PACKET SIZE DISTRIBUTION

Parameter	Total	SystC	SystT	RandC	LP	MuST
Low workload						
Mean pkt size (B)	377.58	387.87	375.65	371.39	390.32	386.70
Peak-to-average	4.00	3.90	4.03	4.07	3.87	3.91
Moderate workload						
Mean pkt size (B)	587.26	586.90	579.42	586.53	589.38	587.82
Peak-to-average	2.57	2.57	2.61	2.58	2.56	2.57
High workload						
Mean pkt size (B)	648.59	647.95	633.33	643.15	635.49	652.36
Peak-to-average	2.33	2.33	2.39	2.35	2.38	2.32

scenario exhibits the highest variability, feature correctly identified by the sampling techniques. For both metrics, the count-based techniques (SystC and RandC) have presented more accurate overall results, followed by the MuST technique, improving the results obtained by time-based techniques.

VI. CONCLUSIONS

Facing the undeniable role of sampling in supporting network measurement tasks, this paper was devoted to a lesser studied yet relevant topic within traffic sampling, which was inspecting the computational weight of classical and recent sampling proposals. The computational weight was measured in terms of CPU load, memory usage and data volume, under distinct workload scenarios. For this purpose, this study resorted to a flexible traffic sampling framework, developed to provide a comprehensive and modular implementation of current sampling techniques, able to be applicable to online and offline traffic environments.

Despite the extensive deployment of count-based techniques, the results have demonstrated that, time-based techniques SystT and MuST have a better relationship between volume of sampled data and computational resource usage. In the overall traffic sampling process, CPU is a more demanded resource than memory, also exhibiting a higher load variation. When considering smaller observation intervals (*e.g.*, one sample), the adaptive techniques present a better accuracy in throughput estimation.

Although the present study considers a specific test environment, the obtained results bring a valuable comparative insight among existing sampling techniques. The present contribution is therefore a step forward in providing a better understanding of traffic sampling techniques overhead regarding their deployment in real scenarios.

Acknowledgements - This work has been supported by FCT - *Fundação para a Ciência e Tecnologia* in the scope of the project: PEst-OE/EEI/UI0319/2014.

REFERENCES

[1] T. Zseby, M. Molina, and N. Duffield, "Sampling and Filtering Techniques for IP Packet Selection RFC 5475," Internet Engineering Task Force, Tech. Rep., 2009.

[2] J. M. C. Silva, P. Carvalho, and S. R. Lima, "A multiadaptive sampling technique for cost-effective network measurements," *Computer Networks*, vol. 57, no. 17, pp. 3357 – 3369, 2013.

[3] H. Young, "Archipelago measurement infrastructure." [Online]. Available: http://www.caida.org/projects/ark/

[4] D. Tammaro, S. Valenti, D. Rossi, and A. Pescapè, "Exploiting packet-sampling measurements for traffic characterization and classification," *International Journal of Network Management*, pp. n/a—-n/a, 2012.

[5] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *Network, IEEE*, vol. 23, no. 1, pp. 6–12, 2009.

[6] Y. Gu, L. Breslau, N. Duffield, and S. Sen, "On Passive One-Way Loss Measurements Using Sampled Flow Statistics," in *INFOCOM 2009, IEEE*, 2009, pp. 2946–2950.

[7] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Improving accuracy in end-to-end packet loss measurement," in *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '05. New York, NY, USA: ACM, 2005, pp. 157–168.

[8] C. Hu, S. Wang, J. Tian, B. Liu, Y. Cheng, and Y. Chen, "Accurate and Efficient Traffic Monitoring Using Adaptive Non-Linear Sampling Method," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008, pp. 26–30.

[9] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 323–336, 2002.

[10] B.-Y. Choi and S. Bhattacharyya, "Observations on Cisco sampled NetFlow," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 3, pp. 18–23, 2005.

[11] H. Wang, Y. Lin, Y. Jin, and S. Cheng, "Easily-Implemented Adaptive Packet Sampling for High Speed Networks Flow Measurement," in *Computational Science ICCS 2006*, ser. LNCS, V. Alexandrov, G. van Albada, P. Sloot, and J. Dongarra, Eds. Springer Berlin / Heidelberg, 2006, vol. 3994, pp. 128–135.

[12] J. M. C. Silva, P. Carvalho, and S. R. Lima, "Enhancing Traffic Sampling scope and efficiency," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, Apr. 2013, pp. 71–72.

[13] E. A. Hernandez, M. C. Chidester, and A. D. George, "Adaptive Sampling for Network Management," *Journal of Network and Systems Management*, vol. 9, no. 4, pp. 409–434, 2001.

[14] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on NetFlow traffic classification," *Computer Networks*, vol. 55, no. 5, pp. 1083–1099, Apr. 2011.