Deploying Time-based Sampling Techniques in Software-Defined Networking

David R. Teixeira
Centro Algoritmi
University of Minho
4710-057 Braga, Portugal

João Marco C. Silva
INESC TEC
University of Minho
4710-057 Braga, Portugal

Solange Rito Lima
Centro Algoritmi
University of Minho
4710-057 Braga, Portugal

***Abstract*—Network data volumes have seen a substantial increase in recent years, in part due to the massive use of mobile devices, the dissemination of streaming services and the rise of concepts such as IoT. This growing trend highlights the need to improve network monitoring systems to cope with challenges related with performance, flexibility and security. Software-Defined Networking (SDN) and traffic sampling techniques can be combined to provide a toolset that can be used for enhancing network management activities and performance evaluation. In this context, this paper presents a proposal for supporting time-based sampling techniques in SDN, providing network statistics at the controller level and allowing the self-configuration of traffic sampling in network devices. The proposed solution, designed to improve the efficiency and flexibility of network measurement systems, takes into account the underlying need to establish a balance between the reliability of the collected data and the computational effort involved in the sampling process. The proof-of-concept results emphasize the potential of applying and configuring different time-based sampling techniques through a SDN framework and a small set of standard OpenFlow messages. Comparative results on the accuracy and overhead of each technique when sampling real traffic traces are also provided.**

***Index Terms*—SDN, traffic sampling, time-based techniques, OpenFlow, Ryu.**

I. INTRODUCTION

Faced with the exponential increase in services supported by computer networks, monitoring activities are playing a vital role in the maintenance of these communication channels. Alongside this, it is important to ensure that network management activities are able to keep up with this accelerated pace, supporting next generation networks and seamless integration with trending concepts such as the Internet of Things (IoT), Cloud Computing and Smart Cities.

Software-Defined Networking (SDN) is viewed as a promising solution in this context, due to its versatility and the amount of diversified solutions that can result from this concept for purposes of forwarding, monitoring, security, auditing, among others. The main principle behind the SDN architecture is the decoupling of the control and data planes. A centralized controller has a global view of the network infrastructure and directly communicates with the applications, establishing a bridge between them and the forwarding devices. To support this interaction, a communication layer exists between the control plane and those components in the data plane, providing programmability of the network behaviour [1].

On the other hand, traditional networks have the data and control planes combined in the same network node. In the control plane, forwarding policies are static, most times unaware of dynamic changes and specific demands coming from users and applications. This approach has proven to be restrictive, affecting network scalability and performance as traffic patterns change [1], requiring considerable efforts to monitor the amount of packets/flows at a given point in time.

Thus, considering the huge traffic volumes of today's networks, traffic sampling has become mandatory in monitoring systems for effective network measurements, especially in the network core, reducing the packets collected to a manageable amount [2], [3]. Based on the flexibility of programmable measurements, SDN also brings a different approach for deploying sampling techniques strategies, enabling a proper balance between sampling efficiency and estimation accuracy.

In this context, having in mind design goals such as versatility and compatibility, this paper presents a new proposal for supporting sampling techniques at SDN controller level, which allows a proper mapping between measuring systems requirements and the self-configuration of sampling tasks on network devices. Considering the gap of existing solutions on supporting time-based sampling, the focus of the developed prototype is on the lightweight deployment of distinct time-based sampling techniques, i.e., Systematic Time-based (SysT) [4], Linear Prediction (LP) [5] and Multiadaptive Sampling Technique (MuST) [6] in SDN environments. In these techniques, timers instead of packet counters rule the intervals of collected packets, being these timers defined systematically or adaptively according to the current network load. The obtained results in a Mininet prototype demonstrate the feasibility and potential of the present proposal, providing quantitative evidence of the low-overhead and accurate estimation of network parameters using real traffic traces from high-speed networks.

The next sections of this paper are organized as follows: related work in this area is analyzed in Section II; the SDN sampling prototype is presented in Section III; the tests methodology is described in Section IV; and the conclusions are included in Section V.

II. RELATED WORK

Although firstly oriented to packet switching, network monitoring have emerged as a promising field for SDN and, currently, several monitoring approaches are already using OpenFlow protocol to collect network metrics. Regarding sampling-based monitoring, few meet the challenges of implementing sampling techniques in SDN. In this context, sFlow, FleXam, PayLess and MonSamp are highlighted.

sFlow is an industry standard (RFC 3176 [7]), used by multiple vendors and early adopted as the sampling technology embedded in many existing switches and routers. This standard defines a set of sampling mechanisms which aim to decentralize monitoring operations typically implemented on the switches, by sending sampled packets to a remote external data collector in partnership with one or more sFlow Agents monitoring traffic and a sFlow Management Information Base (MIB). sFlow and OpenFlow can play complementary roles when implemented over the same network. On one side, the OpenFlow protocol provides a controller running its own software on a separate component, responsible for configuring both switches and hardware forwarding tables. On the other side, the sFlow standard specifies instrumentation to collect real-time information about the state of the network and sends it to a remote monitor. Despite its large deployment, sFlow includes only systematic and random count-based techniques for packet sampling, not offering other sampling strategies, such as the time-based techniques, which are relevant for several activities, such as anomaly detection [6]. Another relevant fact to highlight is the project update status, whose latest version (version 5) was launched several years ago [8], meaning new sampling techniques appeared in the meantime and have not been integrated in sFlow so far.

Another tool implementing sampling using OpenFlow is FleXam [9]. This per-flow sampling framework is described as a flexible sampling extension for OpenFlow that enables the controller to access packet-level information. It provides two sampling possibilities: stochastic, by selecting packets according to a predetermined probability p; or deterministic, which involves collecting m consecutive packets from each k consecutive packets, skipping the first δ packets of the flow. The controller has the power to decide which part of the packet should be sent (*e.g.*, headers only, payload, etc.), as well as their destination. To meet its design goals, modifications to the OpenFlow protocol specification are required aiming to include sampling abilities, thus proposing changes to the standard maintained by the Open Networking Foundation (ONF). Such extension is defined through a new action (OFPAT_SAMPLING) that can be assigned to each flow [9].

From the analysis of FleXam, it is noticeable that the customization of the sampling mechanism has obvious implementation advantages. It can be added to current OpenFlow implementation with no need for matching specific flow tables or perform multiple actions. Secondly, there's no overhead for flows that do not require sampling, given it is flow driven. Nonetheless, this approach promotes unwanted fragmentation in the implementation of a protocol (OpenFlow) that, in itself, is already implemented in several versions by the manufacturers. Furthermore, flow sampling has associated disadvantages, such as it involves classifying packets into flows before or during the sampling process, which consists in high computational burden [3]. Beyond that, FleXam does not offer a per packet sampling mechanism, making it unfeasible to implement time-based sampling techniques [9].

A different approach is supplied by PayLess. It is a query-based monitoring framework for SDN that provides a flexible RESTful API for flow statistics collection at different aggregation levels [10]. This monitoring framework gathers real-time information with high accuracy without causing significant network overhead. PayLess replaces the controller policies when polling the switches, implementing an adaptive scheduling algorithm proposed to achieve the same level of accuracy as continuous polling but with much less communication overhead. Thereby, different network applications can develop new monitoring applications and access the data collected by PayLess with different aggregation levels (such as flow, packet and port). This strategy offers the possibility of also getting information from the network from time to time without causing significant network overhead. It is comparable to the behaviour when a flow is being removed from the switch. In that scenario, if the flow entry's OFPFF_SEND_FLOW_REM flag is set, the switch is required to send a flow removed message to the controller when the timeout is reached, with a complete description of the flow that has been removed, including statistics [11]. The monitoring frequency in PayLess is adjusted according to the network load [12]. Despite the mentioned advantages of PayLess, it is relevant to note that this kind of solution is highly dependent on OpenFlow collectable statistics, potentially compromising the solution's flexibility for specific monitoring purposes and it does not provide a traffic sampling solution.

MonSamp [13] is a SDN application performing flow-based sampling designed with the requirements for a later quality of service assessment. For each monitored flow MonSamp installs a rule into the OpenFlow switches that triggers the action to send a copy of the matched packets to the monitor. These rules are installed on the switches by the controller and the Northbound API. MonSamp limits the number of installed OpenFlow rules and uses thresholds to adjust the amount of monitoring load that is forwarded by the SDN enabled devices to avoid drops. Despite the validity and usefulness of the inherent concepts, the authors found that their monitoring concept requires new developments on a scalable controller architecture that provides a Northbound API able to bring flexibility on the installation and adjustment of rules. In addition, MonSamp do not intent to introduce new techniques for data analysis or traffic sampling.

Facing the above discussion, it becomes evident that supporting sampling techniques in SDN environments has still many open issues, in particular, regarding the flexible implementation of time-based sampling techniques, motivating the present proposal.

III. Proposed SDN Sampling Architecture

This section identifies the main goals driving the design of the SDN sampling proposal, followed by a detailed explanation of the OpenFlow messages in use and the sequence diagram sustaining the communication between switch and controller entities.

A. Design Goals

Proposing a new sampling approach in SDN raises several challenges. Firstly, the solution must provide sampling measurements that are efficient and lightweight while respecting both the SDN model and the OpenFlow standard, ensuring compatibility with any OF-enabled Layer 2 and 3 devices. However, OpenFlow existing specifications were not designed to accommodate sampled data collection. Therefore, the main objective is to enable traffic sampling by defining methodologies in a SDN controller using only currently available messages from OpenFlow protocol. The devised strategy explores the interaction promoted by the separation of data and control planes aiming to collect traffic packets and make sampled measurements available in real-time environments.

The implementation here described focuses on mechanisms that allow the application of time-based sampling techniques, taking advantage of OpenFlow protocol specification and having performance in mind. On that subject, a set of relevant messages within the OpenFlow standard are worth mentioning, as well as their combination to achieve the proposed objective. The messages tailored for the task are *Packet-In* for packet transmission, *Flow Mod* for flow table modification and *Group Mod* for group table modification.

B. OpenFlow Messages

Packet-In: the switch can send a packet to the controller through a Packet-In message, technically called OFPT_PACKET_IN, when packets are received by the datapath. There are three reasons for this message to be sent: i) resulting from an explicit action defined by a match rule asking for this behaviour; ii) due to a miss in the match tables; iii) caused by a Time-To-Live (TTL) error [14].

Flow Mod: one of the main messages existing in OpenFlow that allows the controller to modify the flow characteristics of an OpenFlow switch.

Group Mod: historically, group table modification messages were defined in OF 1.1 [14] and are initiated by the controller to add, delete or modify groups in datapaths with the purpose of defining groups of action for certain flows.

C. Sampling Process

With the defined objective of collecting network details at packet level, all existing and active flows in the network are valid for being queried, aiming at obtaining a diverse and reliable measurement of the network state. The main interactions are detailed in Figure 1, through a sequence chart clarifying the dynamics of the communication process between switch and controller entities in the proposed architecture.

Once the controller and switch acknowledge each other, the controller waits to detect activity in the network. This signal is triggered by the switch when it receives a packet for which it has no actions defined in its flow table (table-miss), and therefore sends it encapsulated in a Packet-In message to the controller. In turn, the controller parses the packet header in order to identify a possible destination in the network for the packet, based on pre-established rules. If a valid destination is found, a Flow Mod message to add the new flow is sent to the switch, which will be used to forward packets with the same destination, without incurring the penalty of awaiting new controller's decision. This is the expected procedure for a Media Access Control (MAC) learning switch.

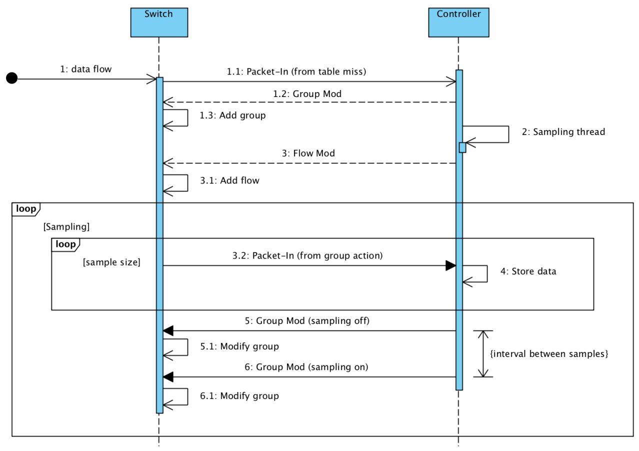


Fig. 1: Sequence diagram representing messages exchanged between switch and controller

Regarding the sampling process, when the first Packet-In message arrives at the controller, it sends back a Group Mod message to insert into the switch called *the sampling group*, essentially containing a bucket with two actions: one to continue the standard packet forwarding action for new packets with the same characteristics; and another to create a copy of each packet that should be sent to the controller, which will take care of saving it without triggering further actions . The packets retrieved include payload, previously specified with an OFP_NO_BUFFER instruction.

Not least important is the use of Flow Mod messages in this prototype, responsible for setting the rules to be followed by the switch for packets with certain characteristics. For the Group Mod message to be correctly parsed, the actions field of the Flow Mod message includes a group with the action bucket to be fulfilled, either simultaneously for all rules, if the sampling mechanism is enabled. In this way, periods of packet sampling (sample size) and interval between sampling can be properly interleaved, according to the desired time-based parameterization.

In a second moment, the controller signals to stop receiving all the packets from the switch. For this purpose, it communicates the decision by sending a Group Mod message with the requirement to modify the previously created group so that the switch only forwards packets to their original destination.

IV. Tests Methodology and Results

The proposed solution is validated through a set of experimental tests executed in a Mininet virtual environment.

To do this, the controller is tested with different trace files and traffic sampling techniques widely discussed in the literature and used in production networks, in particular, one systematic technique - Systematic Time-based (SysT) [4]; and two adaptive techniques - Linear Prediction (LP) [5] and Multiadaptive Sampling Technique (MuST) [6]. Thus, the tests aim at evaluating the feasibility of using time-based sampling in SDN devices, highlighting the comparative accuracy and overhead of different sampling schemes.

A. Experiment setup

To carry out the experiments, different anonymized datasets from real networks and publicly available through The Center for Applied Internet Data Analysis (CAIDA) were selected. The characteristics of the traces are summarized in Table I.

TABLE I: TRACES USED FOR TESTING

Traffic Label	Characteristics	Source
OC-48	Anonymized passive trace taken at an US west coast OC-48 peering link for a large ISP in 2003.	CAIDA [15]
OC-192	Trace contains anonymized passive traffic collected at CAIDA's equinix-chicago high-speed monitor on a commercial backbone in 2015.	CAIDA [16]

Even though there is some general knowledge about the mean throughput and total number of packets the traces contain, there is no in-depth knowledge of the network traffic pattern. The result of each sampling technique depends entirely on the parameters initially stipulated and the systematic or adaptive properties of the technique being used. In the case of the SysT technique, due to the technique's static characteristics, only the initial parameters are relevant to the final result. Regarding time-based adaptive techniques, although the initial parameters are also important, the sampling intervals are modified over time by the techniques themselves in order to self-adapt the sampling frequency towards the characteristics of the traffic traversing the network.

In this test environment, the SysT sample size and time between samples parameters are set to 100 ms and 500 ms, respectively. These values are also proposed in [6] and [5]. For the Linear Prediction (LP) technique, the sample size is set to 100ms and the interval between samples is set to 200ms. The order of prediction (N), *i.e.,* the quantity of previous samples whose values are considered in the next sampling forecast, is set to 2, according to the original specification [5]. In Multiadaptive Sampling Technique (MuST), the sample size and interval between samples are set to 200ms and 300ms, respectively. The order of prediction is set to 3 for this technique. These values were set according to their default values set in [6].

A parameterized topology is defined in Mininet, composed of one Open vSwitch (OVS) switch (version 2.8.90), two hosts and a remote controller. The Ryu controller (version 4.10) is selected to work as a remote entity connected to the switch via OpenFlow (version 1.3). Figure 2 illustrates each component in the architecture.

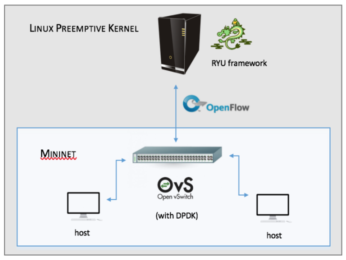


Fig. 2: Architectural scheme

Regarding data security, two configuration parameters can be enabled to strengthen the security of the solution. One setting that can be changed is the size of each packet received by the controller through the global parameter MISS_SEND_-LENGTH. For instance, this parameter can be set to 14 bytes to include only the standard Ethernet header. Furthermore, a TLS connection can be established to secure the channel used by the controller when connecting to the switches. Both Ryu and and Open vSwitch provide this feature.

B. Results

A set of results herewith described was obtained after ten tryouts executed for each sampling technique, applying the average to the results collected. The decision to collect the data this way is due to the fact the test values are not constant because of OVS's variable latency and throughput limitation when using an emulated virtual network environment such as Mininet [17]. In total, 60 tests (30 per trace file, 10 per sampling technique) were performed and the mean values are available below.

To estimate the overhead created by each sampling technique and OpenFlow, the parameters collected in the simulation must be scrutinized. The variables under study are: (i) number of packets collected; (ii) number of Packet-In messages handled by the SDN controller; (iii) data volume; and (iv) number of samples taken during execution. The results, shown in Table II represent objective values to perform a comparative analysis of the obtained results. Figure 3 provides a visual representation of this data.

TABLE II: METRICS COLLECTED FROM ALL SAMPLING TECHNIQUES

Traffic / Parameter	Total	SysT	LP	MuST
OC-48				
Packets	6550395	539228	209322	289925
Packet-In Messages		1254978	493600	604550
Data Volume (MBytes)	372.52	30.67	11.91	16.49
Samples		493	193	42
OC-192				
Packets	14990493	80965	38377	22318
Packet-In Messages		683719	401047	246583
Data Volume (MBytes)	945.87	5.13	2.44	1.43
Number of Samples		82	43	16

As visible, SysT technique captures significantly more traffic than adaptive techniques, through the use of Packet-In messages. The collection of the total number of Packet-In messages aims to assess the overhead introduced by sampling

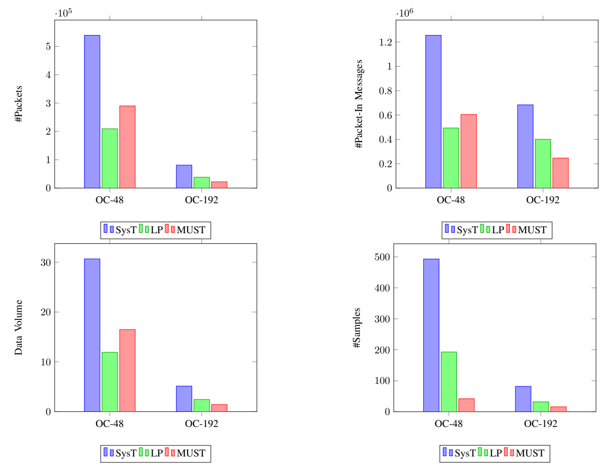
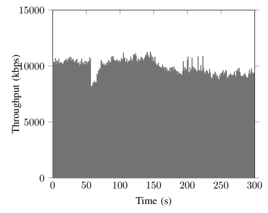
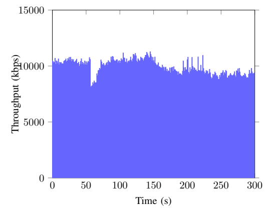


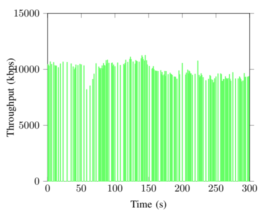
Fig. 3: Comparison of SysT, LP and MuST techniques



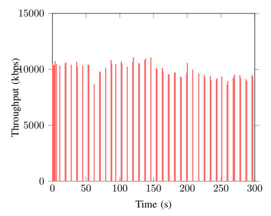
a: Total traffic



b: SysT



c: LP



d: MuST

Fig. 4: Throughput for OC-48 traffic

techniques applied over the proposed architecture in the communication from the switch to the controller, and compare it to a SDN architecture without sampling. To effectively do so, the overall number of Packet-In messages is collected at the controller. The ratio between the number of packets and this value indicates the overhead introduced along with sampling techniques. Other chosen parameters, as suggested in [6] can be defined as:

- *Number of Packets* - amount of packets collected during the period of activity, set by the technique's sample size;
- *Data Volume* - size of the result file (in megabytes), containing the amount of packets collected. It is calculated from the sum of every packet's total length value, within the IO frame;
- *Number of Samples* - number of times the packet capture process was triggered on the measurement point (switch).

Figure 4 shows the original network throughput for the OC-48 traffic and its counterparts (estimated throughput) when applying each time-based sampling technique. The throughput and sampled packets were collected in 1 second time intervals. Since traffic sampling implies selecting only part of the packet stream in the switch, for reliable estimation of the throughput it is necessary to also consider the packets that were not selected. This calculation can be done in several ways, the most common being a statistical extrapolation of the number of unsampled packets, from the sample value gathered in the defined time interval [3].

For statistical purposes, some extra parameters of the original traffic and each sample have to be analysed. These parameters aim to quantify the accuracy of the acquired sample [6]. The parameters in which this work focuses are the following:

- *Throughput* - indicates the estimated data rate in kbps. With the use of statistical extrapolation, allows a correlation between the expected average value and the value obtained with sampling measurements;
- *RME* - allows to infer the mean error of the estimated average throughput compared to the average throughput of the total traffic;
- *Peak-to-average ratio* - ratio between the peak and average throughputs for the same sample. It helps to measure the traffic burstiness;
- *Latency* - average delay response time between the controller and the switch obtained during the tests;
- *Packet loss* - percentage of dropped packets.

The parameters calculated from the tests performed are listed in Table III.

TABLE III: ACCURACY ESTIMATION

Traffic / Parameter	Total	SysT	LP	MuST
OC-48				
Throughput (kbps)	9934.22	9934.90	9903.53	9936.72
RME		0.00007	0.00309	0.00025
Peak-to-average ratio	1.13	1.13	1.14	1.11
Latency (ms)		0.021	0.019	0.019
Packet loss (%)		0	0	0
OC-192				
Throughput (kbps)	154429.34	156406.07	162038.02	156623.15
RME		0.01280	0.04927	0.01415
Peak-to-average ratio	1.06	1.11	1.03	1.04
Latency (ms)		0.023	0.022	0.022
Packet loss (%)		10.93	5.07	4.25

C. Discussion

From the presented results in Figure 3 it is possible to verify that the number of Packet-In messages has the peculiarity of resembling the graphical representation of the number of packets and data volume. This is an interesting fact, having in mind many Packet-In messages were not originated by the sampling action. Effectively, Packet-In messages sent by the switch while OC-48 traffic was being forwarded, represent 42.97%, 42.41% and 47.96% of all Packet-In messages received by the controller, in SysT, LP and MuST sampling techniques, respectively. With regard to OC-192 traffic, Packet-In messages resulting directly from a sampling action represent 11.84%, 9.57% and 9.05% of all Packet-In messages in SysT, LP and MuST sampling techniques, respectively.

The visual differences between charts related to the two traces in Figure 3, are justified by their duration, as the OC-

48 trace is longer than the OC-192 (five to one minute of traffic captured in the network). Hence, the OC-192 sampling results are expected to be quantitatively below the OC-48 data sampled values. However, the difference is smaller for the number of Packet-In messages, which is explained by the large amount of information in the OC-192 trace sent in a short time that has to be properly communicated by the switch to the controller. To corroborate the sampling accuracy in all time-based sampling techniques, the throughput, RME, peak-to-average ratio, latency and packet loss parameters were collected. The values in Table III demonstrate that the sampling results present accurate measures for both test scenarios, for all techniques, having an average accuracy and error level within the values observed in [6], for OC-48 traffic. Despite having a larger margin of error, the OC-192 sampled traffic corresponds, statistically, to the total traffic (unsampled). Table III corroborates this assertion, showing values of RME (Relative Mean Error) tendentially close to zero and values of throughput and peak-to-average ratio near to the values calculated on the total traffic. Notwithstanding the good results obtained, the performance analysis of Open vSwitch showed that in times of heavy load on the virtual switch, *e.g.*, for OC-192 traffic, packet routing is not performed for all packets, resulting in packet loss. The pattern of loss is caused by empty routing tables on the switch in the first milliseconds of execution, forcing it to send the whole traffic to the controller (default action of OpenFlow), ultimately causing packets to be dropped before reaching the receiving host, consequence of a buffer overflow scenario [18]. These occurrences of packet loss are observed up until version 2.8.90 of Open vSwitch.

Facing the above, SDN architecture promotes multilayer programming flexibility being a convenient platform for supporting the configuration of traffic sampling-techniques in network devices. However, some problems (such as latency) arise from the separation of layers that SDN advocates. Regarding the deployment of time-based techniques, the low-overhead and accuracy obtained on the estimation of network load were attested under traffic scenarios representing real network environments. Aiming at enabling time-based sampling in currently deployed SDN environments and supporting the implementation and test of forthcoming sampling schemes, the proposed prototype is currently available as a public project (http://github.com/drteixeira03/sdn_sampling).

V. Conclusions

The decoupling of the control and data planes with high programmability in SDN is changing the way network monitoring tools are being designed. Following this trend, a set of functionalities to be included in the control plane has been proposed with the objective of performing sampling data measurements remotely in the data plane and verify if it could be done in a sustainable way, considering real world traffic traces. Moreover, it was also interesting to verify whether this implementation brings performance benefit over sampling models applied in legacy network environments. From the proof-of-concept it was possible to conclude that the implementation of sampling techniques at the controller level is a viable solution. The calculated evaluation parameters attest it is possible to determine the state of the network with a high degree of accuracy from the sampled measurements that the controller provides, with a very small RME.

Acknowledgment

This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

[1] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, pp. 36–43, Jul. 2013.

[2] D. Tammaro, S. Valenti, D. Rossi, and A. Pescap, "Exploiting packet-sampling measurements for traffic characterization and classification," *International Journal of Network Management*, vol. 22, pp. 451–476, Nov. 2012.

[3] J. M. C. Silva, P. Carvalho, and S. R. Lima, "A Modular Traffic Sampling Architecture: Bringing Versatility and Efficiency to Massive Traffic Analysis," *Journal of Network and Systems Management*, vol. 25, no. 3, pp. 643–668, 2017.

[4] T. Zseby, M. Molina, and N. Duffield, "Sampling and Filtering Techniques for IP Packet Selection RFC 5475," tech. rep., IETF, 2009.

[5] E. Hernandez, M. Chidester, and A. George, "Adaptive sampling for network management," *Journal of Network and Systems Management*, vol. 9, no. 4, pp. 409–434, 2001.

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

[7] P. Phaal, S. Panchen, and N. McKee, "InMon Corporations sFlow: A Method for Monitoring Traffic in Switched and Routed Networks rfc 3176 (informational)." http://www.ietf.org/rfc/rfc3176.txt, 2001.

[8] P. Phaal and M. Lavine, "sFlow Version 5." http://www.sflow.org/sflow_version_5.txt, 2004.

[9] S. Shirali-shahreza and Y. Ganjali, "FleXam : Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow," *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, pp. 167–168, 2013.

[10] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, Oct. 2014.

[11] B. Pfaff *et al.*, "OpenFlow Switch Specification v1.1.0." https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.1.0.pdf, 2011.

[12] S. Chowdhury, M. Bari, and R. Ahmed, "PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks," in *14th IEEE/IFIP Network Operations and Management Symposium*, 05 2014.

[13] D. Raumer, L. Schwaighofer, and G. Carle, "MonSamp: A distributed SDN application for QoS monitoring," in *2014 Federated Conference on Computer Science and Information Systems*, (Warsaw, Poland), IEEE, Sept. 2014.

[14] J. Casey *et al.*, "Flowgrammable - OpenFlow Message Layer." http://flowgrammable.org/sdn/openflow/message-layer.

[15] C. Shannon, E. Aben, K. C. Claffy, D. Andersen, and N. Brownlee, "The CAIDA UCSD Anonymized Passive OC48 Internet Traces Dataset - 20030424-005000-UTC-anon." http://www.caida.org/data/passive/passive_oc48_dataset.xml.

[16] C. Shannon, E. Aben, K. C. Claffy, D. Andersen, and N. Brownlee, "The CAIDA UCSD Anonymized Internet Traces 2015 - 20150917-125911-UTC-anon." http://www.caida.org/data/passive/passive_2015_dataset.xml.

[17] P. Emmerich, D. Raumer, S. Gallenmüller, F. Wohlfart, and G. Carle, "Throughput and Latency of Virtual Switching with Open vSwitch: A Quantitative Analysis," *Journal of Network and Systems Management*, pp. 1–25, 2017.

[18] K. Phemius and M. Bouet, "OpenFlow: Why latency does matter," *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM2013)*, pp. 680–683, 2013.