Reactive Logic in Software-Defined Networking: Measuring Flow-Table Requirements

Maurizio Dusi[§], Roberto Bifulco[§], Francesco Gringoli[†], Fabian Schneider[§]
[§]NEC Laboratories Europe, Germany – E-mail: <firstname.lastname>@neclab.eu
[†]CNIT - Università degli Studi di Brescia, Italy – E-mail: <firstname.lastname>@ing.unibs.it

***Abstract*—The capability of a network is ultimately bounded by limitations of the devices that compose it. In this paper we argue that Software-Defined Networking (SDN) can increase the importance of certain limitations, such as the size and the flexibility of switches forwarding tables. In particular we focus on the implications of reactive installation of flow entries in the switch fabric: by analyzing traffic traces captured in different scenarios we show the existence of a trade-off between the size of the flow table and the rate of dynamic installation of a missing or expired rule. We leverage on this finding to further show that reactive flow (re-)configuration is a promising mechanism for improving the traffic engineering flexibility with no additional requirement in terms of flow table size. We examine links located in various parts of the network and we consider different flow definitions to evaluate the feasibility of using SDN controllers in both access and core network scenarios.**

***Keywords*-Network communications, Software-Defined-Networking, network measurements.**

I. Introduction

In software-defined networking (SDN), the network is introduced with a software control plane, usually implemented as a centralized `controller` and separated from the actual switching hardware. The control plane monitors the state of the network over time by interacting with the switches to determine their load and availability: by reacting on such measurements, the controller dynamically instruments the switches to optimize the packet forwarding process, balance the network load and reduce the overall end-to-end delay.

The ability to dynamically reconfigure the network behavior allows for faster response to emerging applications or requirements that the network has to support, and ultimately for fine-grained traffic engineering, e.g., by considering more packet fields in the routing decision other than the destination IP address alone.

A SDN control plane, e.g., as implemented in the OpenFlow architecture [1], is based on the concept of flow, where a flow is described by a combination of packet header fields. A SDN control application selects the network path followed by a flow by installing flow table entries (FTEs), i.e., couples of flow match rule and forward action, into the flow table (FT) of every network device. OpenFlow further assigns two timeouts to each entry. The *idle* timeout specifies the time after which the entry is removed if no new packets of the flow have come. The *hard* timeout indicates the absolute life-time of an entry, after which the entry is anyway removed. This flexibility, however, impacts on the granularity at which a given network device has to handle incoming flows and on the frequency new FTEs must be installed or updated. Both factors in turn affect the number of entries that must be accommodated within a device's flow table.

978-1-4799-0959-9/14/$31.00

Despite their flexibility, current SDN controller applications primarily rely on a *proactive control logic* [2], [3], which may require the switches to accommodate a number of flow table entries that exceed the capabilities of their Ternary Content-Addressable Memories (TCAMs): providing SDN switches with bigger TCAMs is possible, at the expense of raising operational and power consumption costs. The use of a *reactive control logic*, where resources are allocated and freed depending on the network load and on the effective behavior of the flows, their granularity and their inter-packet arrival time, has been so far overlooked.

The goal of this paper is to show how (i) the flow definition, and (ii) the FTE timeout affect the flow table of the switch: while the former impacts the granularity of the switch forwarding policies, the latter ultimately impacts the delay that the switch introduces to the flows.

We investigate those factors by analysing actual traces collected on links located both in access and core points of the network, which reproduce real scenarios where SDN is deployed. By varying the flow definition and the timeout of the FTEs, we show that current access networks may require small flow tables in the switch fabric (in the order of few thousands, which is an order of magnitude lower than what a proactive control logic requires), with a table update rate in line with the limit of today's technology (about 250 updates/s). Although we show that current generation switches can potentially implement a reactive flow installation logic to improve the routing decision in access-network scenarios, i.e., by considering several packet-header fields to assign the output interface, current flow update technology seems not yet ready to handle the traffic of a 10Gb/s core link by using a reactive logic. We point out the requirements that would satisfy this improved flexibility.

The rest of the paper is organised as follows. In Section II we discuss related work. In Section III we describe the methodology we used to evaluate the requirements needed on a given switch in terms of flow table size and flow installation rate, to handle the traffic that traverses the switch. In Section IV we report our dataset, which we used to represent actual SDN scenarios. In Section V we discuss the

experimental results when applying our methodology to the dataset. Finally, Section VI concludes the paper.

II. Related work

Limitations on FT size as well as FTE installation rate in SDN networks have been reported before in the research community in several fields, e.g., in network virtualization [2], network programming [4] and participatory networking [5].

Works such as [3], [6], [7] address the problem by solely considering the FT size, and suggesting ad-hoc solutions for a limited set of applications, mainly based on aggregation of flows. For instance, Mogul *et al.* [8] propose to keep legacy network control protocols in operation at the switches while exposing a full view of the forwarding configuration as well as the current traffic to the controller. Therefore, the controller can focus on controlling only certain "important" flows.

FT sizes obviously depend on the switch model. HP switches are reported to hold 1500 flow entries (5406zl [8] and J9451A [9]), NEC reports 64k+ flows [10].

Huang *et al.* [9] recognize the need to account for the specific resource constrains at the level of the single switch, and propose an approach to model such constraints into emulation systems specifically for SDN. They find that hardware switches support around 40 flow updates per second, while software switches (OVS) supports around 400/sec on their Xeon X3210.

Apart from providing solutions to specific applications, none of the above works actually quantifies the requirements for FT size and update rate of flow entries on actual traces, for a given flow definition. Our work explores these requirements and the trade-off between them, shedding some light on the viability of proactive and reactive control logic on a SDN network. A preliminary work supporting our thesis can be found in [11].

III. Methodology

We performed this study with the following model in mind: a SDN controller interacts with a given switch by instrumenting it with new FT entries, in response to network events. This means that the switch is not aware of any devices other than the controller within the network, and we ignore any other type of interaction which may take place between the control plane and the switch. To the switch's point of view, the SDN control application solely provides the FTEs to populate and update its flow table.

Starting from this model of interaction between the controller and the switch, we develop a simulator[1] that analyzes the traffic traversing the switch and keeps track of each flow. The traffic is provided either in form of an off-line trace or live packets, while the definition of the flow, including also the entry's timeout value, is configurable and is based on the header fields of, e.g., the IP addresses and the transport protocol. As output, the simulator returns the number of flows being created (i.e., an entry is added to the table), being active (i.e., the table has an entry related to the flow and its timeout has not expired), or being missed (i.e., the entry of a previously expired flow is added again in the table) over time. In the remainder of this paper we will use the term *flow table size* to indicate the number of flows active in a given point in time in a switch's FT.

Note that the statistical characterization of the traffic traversing the switch – e.g., first and second moments of the distribution of packets' inter-arrival time – would not enable us in getting any insight about, e.g., the amount of flow table space occupied in the switch at a given point in time. In fact, by considering only such statistics for every flow we would miss on one side the information related to the flows as defined by the control plane, and on the other side the information about the aggregated resources usage of such flows over time – due for instance to the relation between inactivity of the flow and its timeout expiration as configured in the switch's TCAM.

[1]The code is made available to the community on demand.

IV. SDN Scenarios and Datasets

In this section we present SDN scenarios that motivate our study and use actual traffic traces to reproduce them. In particular, we use datasets collected in different parts of the Internet and different flow definitions to investigate the interactions between the SDN controller and the switch and indeed the requirements in terms of flow table size and flow entry churn on the single switch.

A. Firewalling and QoS policy in access networks

We start by considering the scenario where SDN is used to implement firewalling and quality-of-service policies to flows traversing a switch located in an access network. To that purpose, we assume a fine-grained flow definition, that is, the flow entries in the forwarding table of the switch can be as specific as a /32 IP source and destination addresses, and a port number. We consider two traffic traces that were collected on two geographically different access network links: the first trace was collected at the DSLAM of an European customer ISP, and it lasts one day long of 2007. Due to non-disclosure agreement, we cannot release the trace; the second trace, UniBS2009, was instead collected on the edge router of the campus network of the University of Brescia, Italy on three consecutive working days of 2009. The trace together with its full details are available upon request [12].

B. Forwarding policy in core networks

We then consider the scenario where SDN is used to implement forwarding policies to flows traversing a core-network switch. To that purpose, we assume a coarse-grained flow definition, e.g., the flow entries in the forwarding table of the switch relate to a class of /24 and /16 IP source and destination addresses.

In this case, we exploit traffic traces that were collected on geographically different backbone links and across several time windows in 2005 and 2013. In particular, we consider the TCP and UDP traffic on the following traces: PAIX 2005-01-21, collected on a OC48 trunk of an US Commercial Tier1

Access Network

Flow definition	Flow table size	
	UniBS2009	DSLAM
/24 dst	27664	38964
/24 dst + /24 src	56669	164501
/32 dst	31213	61613
/32 dst + dst port	84597	237718
/32 dst + /32 src	64430	182356
/32 dst + /32 src + dst port	107025	336599

Core Network

Flow definition	Flow table size	
	PAIX2005	CAIDA2013
/16 dst	2914	1979
/16 src	1720	5662
/16 src + /16 dst	127039	293384
/24 dst	74152	97152
/24 src + /24 dst	1267187	1094808
/32 dst	739376	389636
/32 src + /32 dst	1933482	1578977

TABLE I
PROACTIVE CONTROL LOGIC IN SDN NETWORKS: NUMBER OF FLOW-TABLE ENTRIES IN AN ACCESS-NETWORK (TOP) AND CORE-NETWORK (BOTTOM) SWITCH FOR DIFFERENT FLOW DEFINITIONS.

backbone link connecting San Jose and Seattle, and CAIDA 2013-08-15, collected on a OC192 link of a Tier1 ISP between Chicago and Seattle. The traces with their details are made publicly available by CAIDA [13].

V. EXPERIMENTAL ANALYSIS

In this section we investigate the feasibility of using SDN controllers in core and access networks with different flow definition: the goal is to provide an insight of the granularity at which rules can be installed within a SDN network at different locations.

We first consider the case in which the SDN controller exploits a proactive control logic, and analyze the requirements of the switch in terms of flow table size. We then consider SDN controllers which implement a reactive control logic, and evaluate the requirements in terms of flow table size and flow updates when varying the timeout value of the entries in the table of the switch.

A. Proactive Control Logic in SDN: access and core networks

Table I reports the number of flow-table entries required by a switch located on an access-network link and on a core-network link, with different flow definitions and considering ten minutes of traffic.

Numerical results on an access network (DSLAM) reveal that a proactive control logic would require at least 39K entries for forwarding rules that are based on /24 classes of IP addresses (Table I[top], row no.1), up to around 240K in case we are to implement fine-grained rules that take into account the destination IP address and the transport port, such as when implementing firewalling policies (Table I[top], row no.4). Whereas some switches would be able to fulfill the forwarding based on /24 subnets [9], having finer-grained entries would be too costly in a proactive control logic (over 300K entries according to the DSLAM trace and over 100K entries in the UniBS trace, in case we consider both IP addresses and the destination transport port, (the last row in Table I[top]).

On the CAIDA2013 core network link, instead, the number of entries is below 6K in case the forwarding is performed based on /16 subnets (first two rows of Table I[bottom]). In case the forwarding is based on /24 subnets, the number of entries increases by a factor >15 (Table I[bottom], row no.4). Even assuming that future SDN switches, designed for core networks, will be able to support the same number of entries available in today's BGP routers (e.g., the flow table size may exceed 450K entries [14]), SDN controllers cannot manage a flow definition of /24 source and /24 destination IP addresses (Table I[bottom], row no.5): even with a ten-minute traffic trace, it doubles the capacity of today's routers. Similar considerations hold also on the other core network trace, PAIX2005. On the other side, this analysis reveals that today's routers can handle a flow definition which includes /16 source and /16 destination IP address (Table I[bottom], row no.3).

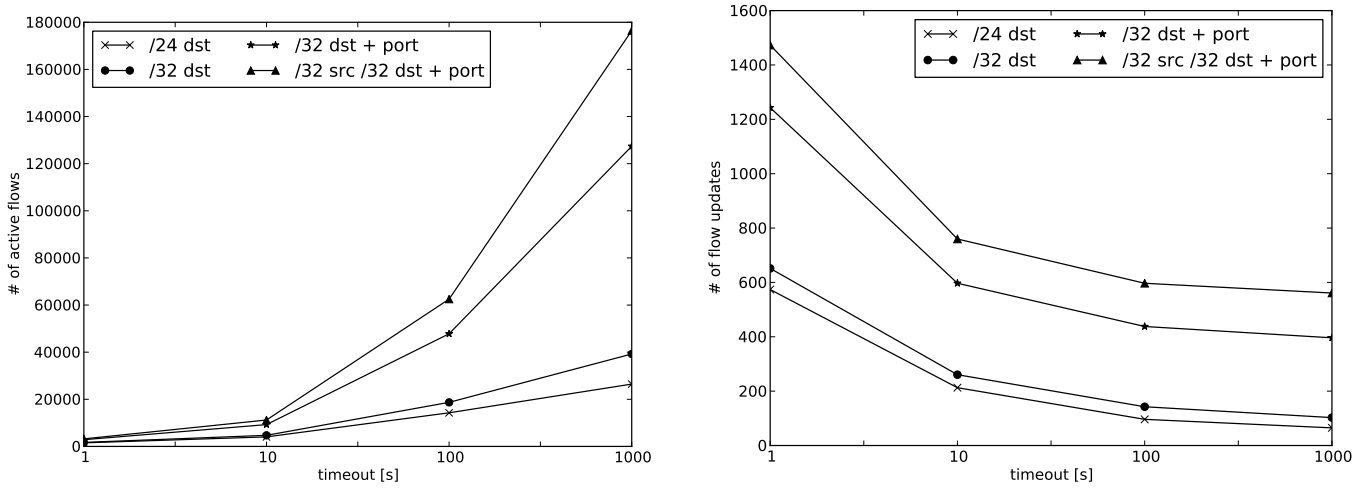
B. Reactive Control Logic in SDN: access and core networks

In a reactive control logic, the entries in the flow table of the switch are installed only when a packet for the corresponding flow is first seen; on the other side, the entry is removed if a time interval has elapsed since last reception of a packet belonging to that flow. The value of this timeout is crucial as it allows to trade the number of entries to be kept in the flow table with the number of interactions between the switch and the controller for re-installing expired rules, i.e., the flow-table update.

To understand the nature of this trade off, in Figure 1 and in Figure 2 we show the requirements in terms of flow table size and flow table update as we increase the timeout values from 1 to 1000 seconds, with a logarithmic step. Figure 1 refers to the DSLAM access network trace, while Figure 2 refers to the CAIDA2013 core network trace. We observe similar trends for the other two traces, i.e., UNIBS2009 and PAIX2005, respectively (not shown here). In the following we show graphs related to the same core and access network links, i.e., CAIDA2013 and DSLAM: unless otherwise reported, same considerations hold for the other traces collected on the same logical links, respectively.

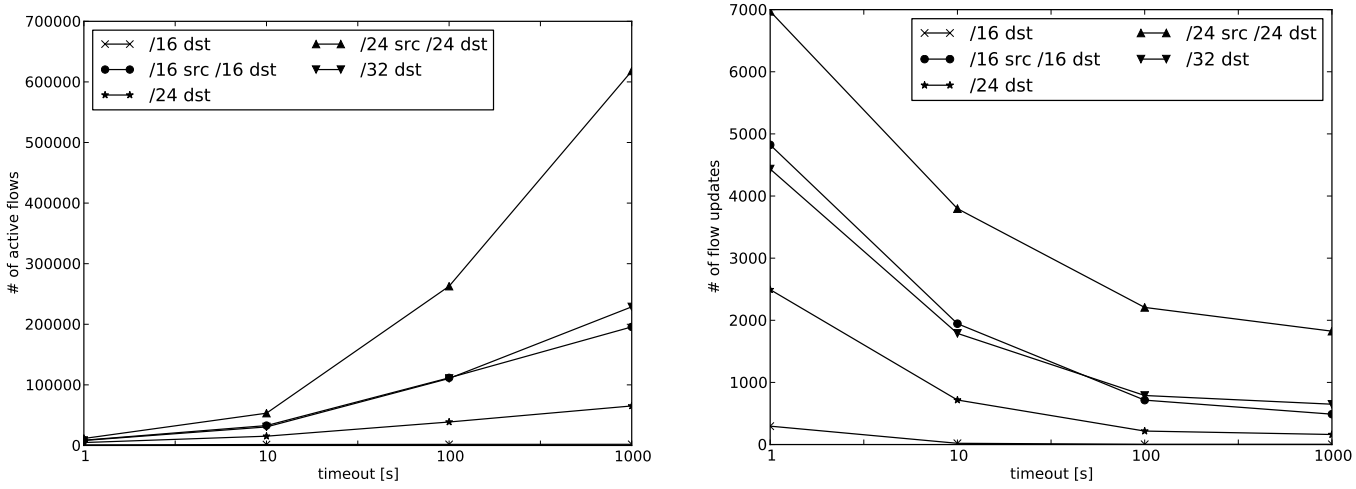
For a given flow definition, the graphs show the average value of the flow table size (flow table update), computed over the entire traffic trace. As expected, the timeout value has a positive correlation with the flow table size, and a negative correlation with the number of updates, that is, the longer we allow flows to be kept into the table, the more space we need, and fewer interactions are needed with the controller. This is particularly true with fine-grained flow definitions, e.g. with rules that deal with /32 addresses, as the probability of hitting a timeout expiration is higher due to the less flow aggregation.

We repeated the analysis by varying the timeout value between 1 and 10 seconds, with a linear step. This zoom-in allows us to investigate the presence of any threshold effect



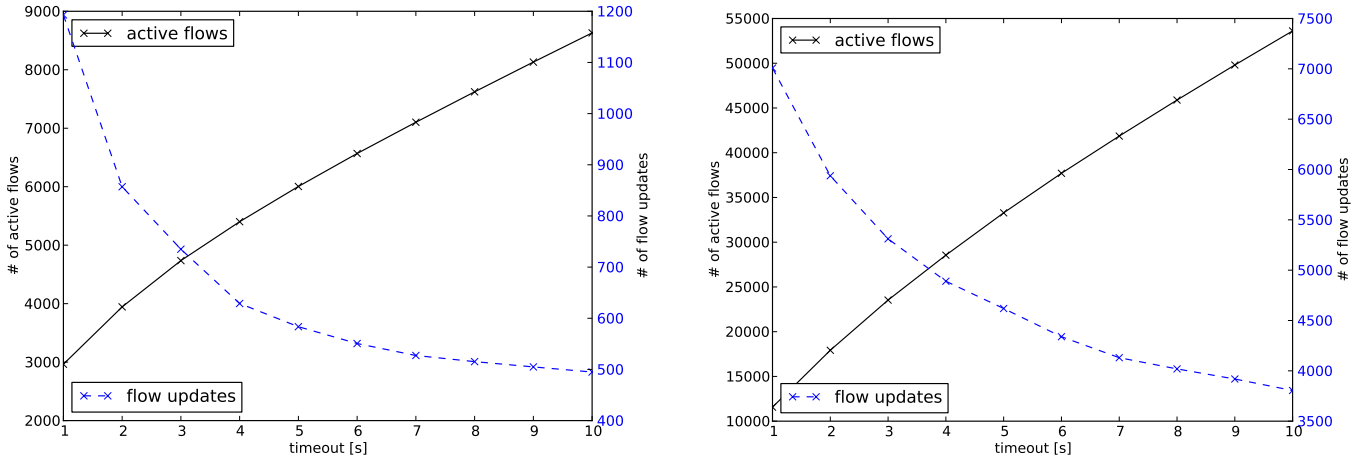
(a) Active flows (b) Flow updates

Fig. 1. Flow table size and update rate for different timeout values and different flow definitions: DSLAM traffic trace.



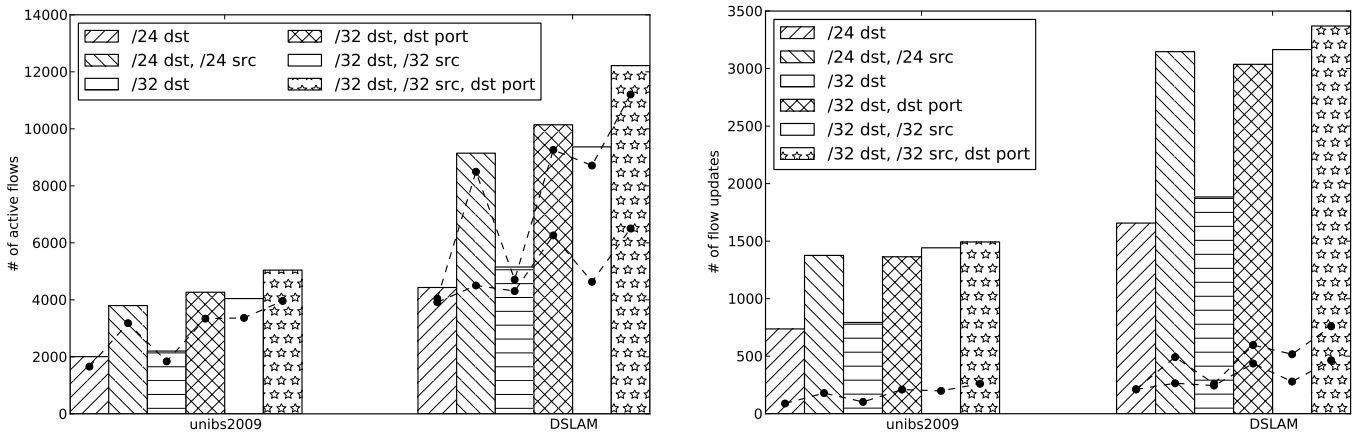
(a) Active flows (b) Flow updates

Fig. 2. Flow table size and update rate for different timeout values and different flow definitions: CAIDA2013 traffic trace.



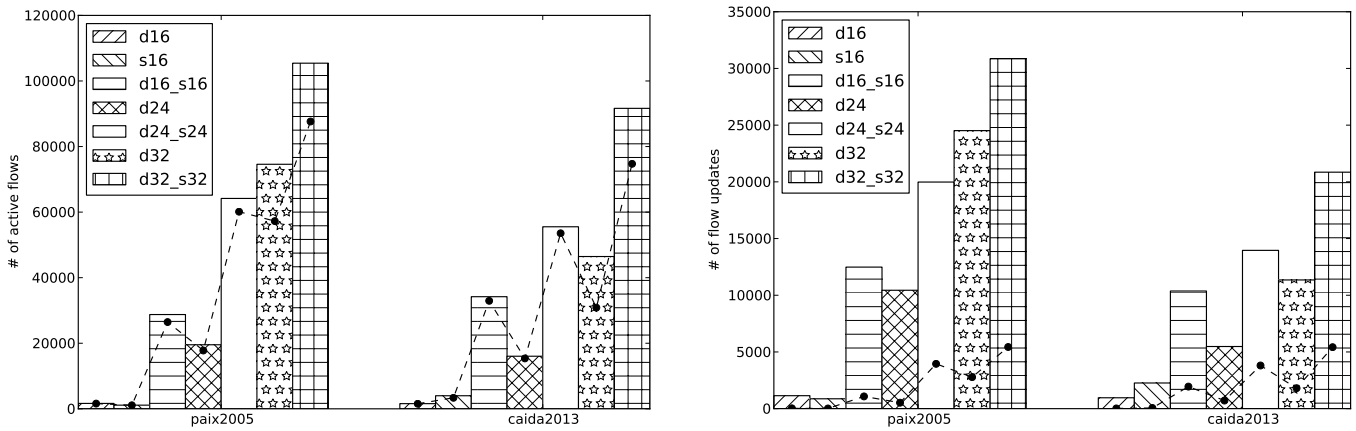
(a) DSLAM (b) CAIDA2013

Fig. 3. Active flows (left scale) and flow updates (right scale) when varying the flow timeout between 1 and 10 seconds, when using a flow definition based on /32 destination and /32 source IP addresses (DSLAM case) and /24 destination and /24 source IP addresses (CAIDA2013 case)



(a) Active flows (b) Flow updates

Fig. 4. Flow table size and update rate for different flow definitions and a timeout value of 10s. The dotted lines connect the average number of active flows and flow updates for each trace and for each flow definition, respectively. In the DSLAM trace, there is a dotted line for the outgoing direction of traffic (users to Internet, represented as the bottom line) and for both directions (the top line).



(a) Active flows (b) Flow updates

Fig. 5. Flow table size and update rate for different flow definitions and a timeout value of 10s. The dotted lines connect the average number of active flows and flow updates for each trace and for each flow definition, respectively.

above which the flow table size tends to explode. In other words, by doing this analysis we try to find out timeouts that allow for values of flow table size and flow updates which can be handled by a SDN scenario represented by a switch interacting with a controller.

Figure 3 shows the results of this analysis for the DSLAM trace, with flow definition based on /32 source and /32 destination IP addresses, and for the CAIDA2013 trace, with flow definition based on /24 source and /24 destination IP addresses. Apart for a normalization factor, in both cases the number of flow table updates sensibly decreases when increasing the timeout from 1 to 5 seconds, and it slows down from 5 to 10 seconds, following a hyperbolic trend. Instead, the size of the flow table follows a linearly increasing trend.

The selection of a timeout value can be driven by the results shown in this paper, jointly with the hardware constraints of the switches. Here we show the effects on flow table size and update requirements when varying the flow definition, for a fixed timeout value set to 10 seconds. We report the results in Figure 4 and Figure 5. The graphs report as bars the maximum values while the dots displayed on the bars represent the average values.

The first effect of introducing a reactive logic with this timeout value is that the requirements in the flow table size are lowered, regardless of the flow definition. In particular, a flow definition based on the destination IP address would lower its requirement on the FT size from 60 thousand entries (see Table I) to less than 5 thousand (Figure 4a), for the DSLAM trace, while still maintaining a reasonable average flow update rate of 250 updates per second (the dotted line in Figure 4b shows the average flow update rate). Please note that the DSLAM usually observes asymmetric traffic, given that it is the connection point between home users and the rest of the Internet. Hence, we decided to show both the average values when considering only the network traffic in the home user-to-Internet direction (the bottom connected dots on the bars) and when considering traffic in both directions (the top connected dots on the bars).

In the CAIDA2013 case, the same flow definition would lead to reduce the flow table size of 350K entries, going down to 40K, with a number of flow updates around 2K, that today's technology struggle to support. Particularly for the update rates, the maximum is much bigger than the average. The analysis of the time series of the values reported by our simulator explains these results: during the initialization of the simulation, i.e., when the switch flow table is still empty, each new flow requires to create an entry, leading to a spike in the required number of flow table updates, which is related to the maximum value observed.

VI. Conclusions

In this paper we studied the feasibility of dynamically configuring forwarding policies through a SDN controller. By analyzing traffic traces collected on different network links, from the access to the core of the network, we investigated the requirements in terms of flow table size and flow table updates, for different flow definitions and varying the timeout value. This study is also meant to provide insights on the granularity at which the controller can install rules on today's switches. Our data reveals that current technology can already cope with the traffic flowing on access network links with granularity up to /32 IP destination addresses: for instance, on the DSLAM trace, the number of table updates are around 200 on average, with a flow table size of around 4500 entries. On core network links, we observe values of flow table size supported by today's switch with timeout of 10 seconds, even though the required updates represent a limiting factor and current SDN controllers cannot handle it. As future work, we plan to extend our analysis and find out values of timeout able to cope with today's update capabilities: we expect that the resulting flow table size will be anyway smaller than the size required by a proactive control logic, even when considering fine-grained flow definition. Finally, we plan to quantify the effects of the flow table updates such as the delay introduced by the interaction between the switch and the controller.

Acknowledgment

The research leading to these results has received funding from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane").

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[2] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924969

[3] N. Kang, Z. Liu, J. Rexford, and D. Walker, "An efficient distributed implementation of one big switch," open Networking Summit, April 2013. [Online]. Available: Open Networking Summit, April 2013

[4] N. Foster, A. Guha, M. Reitblatt, A. Story, M. Freedman, N. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger, D. Walker, and R. Harrison, "Languages for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 128–134, 2013.

[5] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: an api for application control of sdns," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 327–338. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486003

[6] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, and P. Hudak, "Maple: simplifying sdn programming using algorithmic policies," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 87–98. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486030

[7] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 27–38. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486022

[8] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "Devoflow: cost-effective flow management for high performance enterprise networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/1868447.1868448

[9] D. Y. Huang, K. Yocum, and A. C. Snoeren, "High-fidelity switch models for software-defined network emulation," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 43–48. [Online]. Available: http://doi.acm.org/10.1145/2491185.2491188

[10] NEC, "NEC ProgrammableFlow UNIVERGE PF5240 Datasheet," http://www.necam.com/docs/?id=5ce9b8d9-e3f3-41de-a5c2-6bd7c9b37246.

[11] A. Zarek, "OpenFlow Timeouts Demystified," Master's thesis, University of Toronto, 2012.

[12] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. Claffy, "GT: picking up the truth from the ground for Internet traffic," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 13–18, Oct. 2009.

[13] CAIDA, "Passive OC48 and OC192 Traces," http://www.caida.org/data/passive/trace_stats.

[14] M. Zec, L. Rizzo, and M. Mikuc, "Dxr: towards a billion routing lookups per second in software," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 29–36, Sept. 2012. [Online]. Available: http://doi.acm.org/10.1145/2378956.2378961