Coarse-grained Scheduling with Software-Defined Networking Switches

Myriana Rifai
University Nice Sophia
Antipolis, France
mrifai@i3s.unice.fr

Dino Lopez-Pacheco
University Nice Sophia
Antipolis, France
dino.lopez@unice.fr

Guillaume Urvoy-Keller
University Nice Sophia
Antipolis, France
urvoy@i3s.unice.fr

ABSTRACT

Software-Defined Networking (SDN) enables consolidation of the control plane of a set of network equipments with a fine-grained control of traffic flows inside the network. In this work, we demonstrate that some coarse-grained scheduling mechanisms can be easily offered by SDN switches without requiring any unsupported operation in OpenFlow. We leverage the feedback loop - flow statistics - exposed by SDN switches to the controller, combined with priority queuing mechanisms, usually available in typical switches on their output ports. We illustrate our approach through experimentations with an OpenvSwitch SDN switch controlled by a Beacon controller.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations

Keywords

Software-Defined Networking, Size-Based Scheduling

1. INTRODUCTION

Software-Defined Networking (SDN) offers a centralized control of the traffic flowing inside a network. The flexibility of the flow definition enables implementation of advanced functions like virtual networks or firewalls. In contrast, the data plane remains opaque and cannot be directly influenced by the controller. Few works have addressed the extension of SDN to the data-plane with scheduling in mind. A noticeable exception is [6] where the authors first advocate the need to use different scheduling or buffer management solutions in different scenarios as there is no one-fit-all solution.

Our objective in this work is to demonstrate that despite the limited toolbox offered by SDN to directly manipulate the data plane, one can however implement some form of coarse grained scheduling with legacy SDN equipments. Our focus is on size-based scheduling [1, 5], where priority is given to flows in their early stage. This approach is valuable in the current Internet and data centers networks where (i) the bulk of traffic is carried by TCP and (ii) consists of a majority of short flows. Our approach takes advantage of the feedback loop offered in SDN where a switch exposes to the controller per rule statistics. The latter enables us to identify long flows and separate them from short flows using multiple queues per port that serve to implement 802.1p QoS mechanisms. Our objective is to minimize flow completion of the majority of flows.

Related work includes [7] where the authors divert some flows on longer rarely used paths to decrease completion time. In [2], the authors rely on the same capacity of (non SDN) commodity switches to offer several queues with a tagging process performed at the end hosts, complemented with ECN in the network. The authors in [4] propose to improve the performance of TCP in SDN networks, albeit at the cost of modifying the end-hosts protocol stack.

In contrast, we investigate a purely network centric approach, with no modification of the end-hosts for SDN networks.

We demonstrate the feasibility of building an SDN based size-based scheduler using OpenvSwitch with Beacon controller. We further propose a scalable version of our scheduler to avoid continuous monitoring of each active flow by the switch and the controller.

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.

DOI: http://dx.doi.org/10.1145/2785956.2790004

2. DESIGN

We have devised two size-based schedulers, a state-full and a scalable scheduler, and implemented them as applications on top of a Beacon controller. Both solutions require two queues per port (802.1p mandates 8 queues per port) and assume that those queues are managed with a strict priority scheduler where the high priority queue is served as long as it has packets - the other queue is served when the first one is empty. We next detail the two schedulers:

State-full scheduler: The switch monitors all ongoing flows and the controller queries every $T_{\text{monitor}} = 10ms$ flow statistics. Upon arrival of a new flow, the controller installs a new rule for this flow and assigns it the high priority queue of the corresponding port. If a flow has exceeded $T_{\text{threshold\_pkts}} = 100$ packets, then the controller modifies the queue used by this flow and assigns it the lower priority queue.

Scalable scheduler: Continuous monitoring of each active flow is resource consuming, and installing per-flow rules could quickly overload the forwarding table of SDN devices. Also, when the load on the port is low, say lower than 50%, all schedulers typically offer the same performance as queues do not build up. To address these concerns, we propose a second scheduler where the controller initially sets up one default rule for a set of flows. Then, if the throughput for that rule exceeds a threshold $T_{\text{threshold\_utilization}} = 0.1 \times \text{LinkCapacity}$, then the scheduler zooms in the traffic to separate large flows from short flows. To zoom in traffic, the scheduler uninstalls the forwarding rule triggering the mechanism and installs

per flow rules on traffic demand. Every $T_{\text{monitor}} = 10ms$ the scheduler will check the flow size, if it is more than $T_{\text{threshold\_pkts}} = 100$ packets it will modify the queue used to the lower priority one. After a few T_{monitor} cycles, large flows are isolated and the grouped forwarding rule is reinstalled with a high priority.

3. PRELIMINARY RESULTS

To illustrate our approach, we consider the experimental set-up described in Figure 1. We have ten clients and one server that acts as a sink for traffic and an OpenvSwitch (OVS) switch connected to a Beacon controller. The traffic workload is generated using impt [1] and consists of bulk TCP transfers. The distribution of flow size follows a bounded Zipf distribution with a flow size between 15 KB (10 packets) and 10 MB. The average flow size is around 100 packets, in line with typical average flow size in the Internet. The Zipf distribution is the discrete equivalent of a Pareto distribution, which is known as a reasonable model of Internet and also data center flow size [3]. The load is controlled by tuning the flow inter-arrival time, which follows a Poisson process.

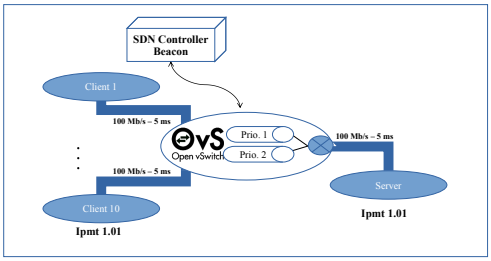

Figure 1: Experimental set-up

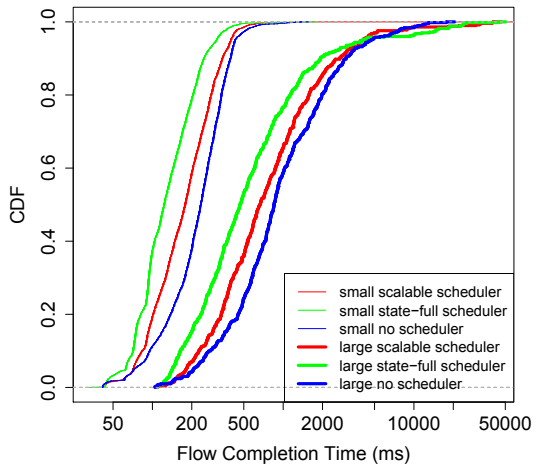

Figure 2: Flow completion time CDF for long and short flows

We present in Figure 2 results obtained out of 10 experiments for a load of about 90%. We distinguish between small flows and large flows, where small flows are defined as flows smaller than the 90-th quantile of the flow size distribution. The 90-th quantile corresponds to about 95 packets (close to $T_{\text{threshold\_pkts}}$). It is important to note that small flows represent about 15% of the load.

We can observe from Figure 2 that the state-full scheduler offers the best response time for small flows and for the majority of large flows as their 100 first packets will take advantage of the scheduling mechanism. Only a minority of the largest flows suffer in the state-full scheduler. As for the scalable scheduler, it offers intermediate results between the state-full scheduler and the absence of scheduler as it needs 10 additional ms ($= T_{\text{monitor}}$) to detect large flows. As we cannot simultaneously obtain better response times for all flows, one can observe the longer tails of response times for the two schedulers as compared to the case with no scheduler.

Though preliminary, those results are encouraging and clearly highlight the validity of our approach.

4. FUTURE WORK

In terms of future direction, we intend to develop an autonomic version of the scheduler that would adjust its parameters dynamically. For instance, $T_{\text{threshold\_pkts}}$ could be set as a quantile of the flow size distribution. Estimating the tail of a highly varying distribution is complex; however, we are interested in separating large flows from short flows only, which does not require a high precision on the computation of quantiles of the distribution. More interestingly, variations in the input distribution or load are easily exposed through the statistics offered by SDN and we expect to be able to take advantage of this feature to build a fully adaptable scheduler.

We also would like to explore the use of size-based scheduling in the data center, where it could be deployed at any switch. Using size-based scheduling at several bottleneck links has never been explored so far, to the best or our knowledge.

Last but not least, SDN enables to enlarge our vision of scheduling as the controller enables us to mix routing and scheduling. This could be useful for specific applications such as VM migration, a crucial operation in modern data centers.

5. REFERENCES

[1] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation between short and long TCP flows: Predictability of the response time. In *Proceedings of IEEE INFOCOM 2004*, 2004.

[2] W. Bai, K. Chen, H. Wang, L. Chen, D. Han, and C. Tian. Information-agnostic flow scheduling for commodity data centers. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 455–468, Oakland, CA, May 2015. USENIX Association.

[3] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding data center traffic characteristics. *Computer Communication Review*, 40(1), 2010.

[4] M. Ghobadi, S. H. Yeganeh, and Y. Ganjali. Rethinking end-to-end congestion control in software-defined networks. In *Proceedings of HotNets-XI*, 2012.

[5] I. A. Rai, E. W. Biersack, and G. Urvoy-Keller. Size-based scheduling to improve the performance of short TCP flows. *IEEE Network*, 19(1), 2005.

[6] A. Sivaraman, K. Winstein, S. Subramanian, and H. Balakrishnan. No silver bullet: Extending sdn to the data plane. In *Proceedings of HotNets-XII*, 2013.

[7] F. P. Tso, G. Hamilton, R. Weber, C. S. Perkins, and D. P. Pezaros. Longer is better: Exploiting path diversity in data center networks. In *Proceedings of IEEE ICDCS '13*, 2013.

[1] http://ipmt.forge.imag.fr/