Fully Functional Rate Limiter Design on Programmable Hardware Switches

Yongchao He
Tsinghua University
heyc18@mails.tsinghua.edu.cn

Wenfei Wu
Tsinghua University
wenfeiwu@tsinghua.edu.cn

CCS CONCEPTS

• **Networks → Middle boxes / network appliances**.

KEYWORDS

P4, Rate Limiter, Network Function

ACM Reference Format:
Yongchao He and Wenfei Wu. 2019. Fully Functional Rate Limiter Design on Programmable Hardware Switches. In *SIGCOMM '19: ACM SIGCOMM 2019 Conference (SIGCOMM Posters and Demos '19), August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3342280.3342344

1 INTRODUCTION

Rate Limiters play a key role in network QoS management such as bandwidth allocation and performance isolation. Rate limiters can be implemented in various locations in the network (e.g., Linux HTB, NIC, switches [4–6]); however, in certain scenarios where network operators have no access to end hosts, a rate limiter can only be implemented on network devices (e.g., mobile core networks, IaaS cloud providing bare mental machines). A recent trend of programmable switches [1] gives an opportunity to implement such *in-network rate limiters*.

In this project, we would explore the *approach to implement a rate limiter under the constraints of hardware programmable switches*. While the current programmable hardware switch provides some degree of packet processing flexibility, but there still exist constraints[1]: (1) the data flow in a switch is uni-direction, and it can only go from the switch buffer to the switching circuit; (2) the programmability is only limited to work on the switching circuit, not available on buffers; (3) the computation on switches is limited, without native support to operations like multiplication and division, and temporal logic; (4) switch memory is scarce to scale to many flows' processing.

[1]These constraints do not mean they are not implementable on a switch, but they are from the tradeoff of functions/features and the performance (i.e., backbone processing speed).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCOMM Posters and Demos '19, August 19–23, 2019, Beijing, China
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342344

The design space of a rate limiter includes algorithmic choice (leaky bucket v.s. token bucket), excessive traffic policy (traffic shaping v.s. traffic policing [3]), and implementation approach (timer-based v.s. event-based). Due to the limitation on programmable hardware switches, we could only implement the token bucket algorithm with traffic policing[2].

In the rate limiting algorithm, two parameters committed $rate$ and burst size BS are configured, and a variable $token$ is maintained. The $token$ accumulates with time and is constrained by BS as threshold. For each packet, if there are sufficient tokens (i.e., $token >= pkt.size$), the packet is sent and the $token$ is reduced; otherwise, the packet is dropped. Note that "the token accumulates with time" can be implemented in two ways: in a timer-based approach, a timer periodically triggers the token update to accumulate the token within the period; in an event-based approach, each packet triggers the accumulation calculation within the duration between the current packet and the previous successfully sent packet (multiplied by the $rate$).

We first profile the performance of a timer-based and an event-based rate limiter, and show their *insufficiency caused by hardware limitations*: the timer-based rate limiter is not *TCP friendly* with throughput oscillation; and the event-based rate limiter is not flexible in rate control. To build a *fully functional* rate limiter, we propose several improvements and optimizations to achieve a rate limiter on the current program hardware switches with (1) committed rate saturation, (2) low oscillation, (3) rate flexibility, and (4) memory efficiency.

2 PROFILING RATE LIMITERS

We implement the two rate limiters and profile them. The experiment has a switch in the middle and two servers on the edge (a sender and a receiver) with 10Gbps NIC. The RTT on the testbed is less than 100us. We tune the committed rate and burst size as the parameter and measure the throughput and the oscillation (i.e., the standard deviation of the throuhgput). Figure 1 shows the results, and we observe that both rate limiters can achieve rate control. But each of them has its own insufficient aspects.

The timer-based rate limiter makes TCP flows experience obvious oscillation and the oscillation *cannot* be improved by tuning burst size. We notice that the refreshment interval is larger than RTT (>4ms v.s. 100us). Within each refreshment interval, there are several rounds of packet round trips, which leads to the final window size (before drop) to be larger and the rate limiter to be easier to drop packets. We also argue that, in most data center networks,

[2]In details, we cannot program the switch buffer to *selectively* drop excessive packets, so the leaky bucket algorithm is excluded; and the switch hardware circuit cannot withhold or put packet back to buffer, so traffic shaping is excluded.

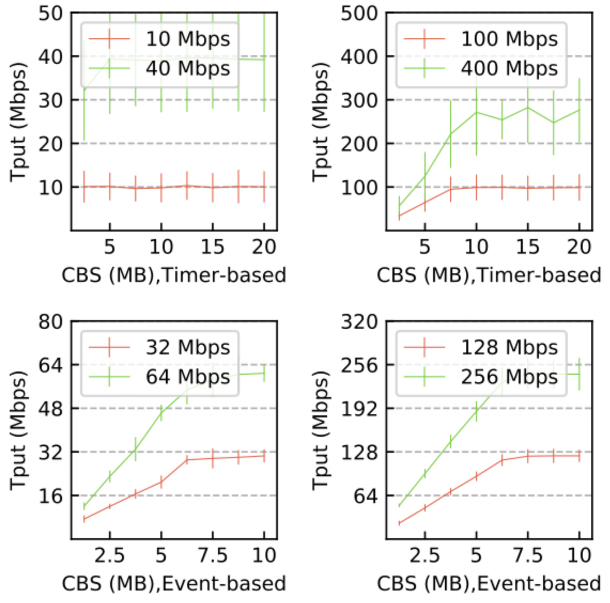

Figure 1: Mean values and variances of TCP throughput varying with Committed Burst Size (CBS)

the RTT is sub-millisecond [2], and they would experience such oscillation problem once the P4 meters are deployed.

The event-based rate limiters reduce the oscillation significantly. For example, in the timer-based rate limiter the oscillation is $> 30\%$ when the rate is 40Mbps; but in an event-based rate limiter, that is $\leq 7\%$ when the rate is 64Mbps. The reason of such improvement is that the packet arrival interval in the event-based rate limiter is usually smaller than the refreshment time interval in the timer-based one, and even smaller than RTT. Thus, the token value can be accumulated, updated, and consumed in a more timely manner, which contributes to a more smooth and precise rate control.

However, the event-based rate limiters are *inflexible in rate control*. The essential reason is that in the token update, the product of $timeInterval \times rate$ requires multiplication $\times$, which is not supproted by the current programmable hardware. The best candidate to support such an operation is "shift $<<$", and thus, only limited rate can be configured (i.e., 2^nMbps).

3 FULLY FUNCTIONAL RATE LIMITER DESIGN

We propose the following improvements and optimization for the event-based rate limiter to get a fully functional rate limiter.

Achieving Multiplication. We present Approximate Multiplication Table (AMT) to overcome limited computation. The key idea of AMT is to pre-compute intermediate results and store in a table, then all computations in the runtime are transformed as table look-ups. For example, if we need to compute $a \times b = c$, we can pre-compute a table with $< a, b >$ as the key to look up and c as the value in the entry.Since it is not possible to pre-compute products of multiplicand and multiplier in arbitrary granularity and range (e.g., real number domain), it is actually an approximate algorithm. And an analysis of the tradeoff between storage space (i.e., granularity) and accuracy (error in result) is needed. Similarly, we can achieve division with Approximate Division Table(ADT).

Memory Efficient Optimization. We also notice that the scarce switch memory is a potential scalability limitation (in terms of the number of rate limiters configured), and thus improve the memory usage of the token bucket algorithm. The original algorithm maintains two variables, previous timestamp *prevTime* and *token* to compute whether the current packet should be sent. The variable *token* records the "credit" as the budget to send packets. We can eliminate this variable and compute the "credit" in time domain. That is, we use ONE variable "$timeToken$" to record the credit and the credit consumption of previous successfully sent packets. Each successfully sent packet would consume credits by adding $pkt.size/rate$ to the $timeToken$, and $timeToken$ update is also constrained in the range of $currentTime - BurstSize/rate$ and $currentTime$.

We prototype our design and show the effect in Figure2. We achieve a fully functional rate limiter with the following features: (1) committed rate saturation, (2) low oscillation, (3) rate control flexibility, and (4) memory efficiency.

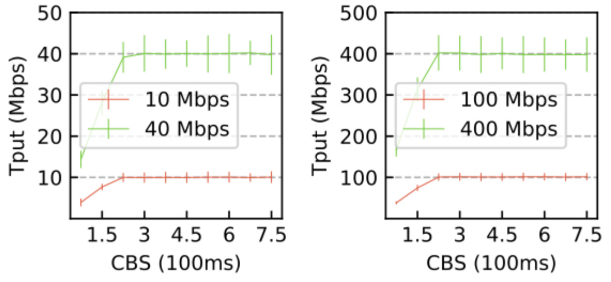

Figure 2: Mean values and variances of TCP throughput varying with Committed Burst Size (CBS)

REFERENCES

[1] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.

[2] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 537–550. https://doi.org/10.1145/2785956.2787510

[3] Cisco Tech Notes. [n. d.]. Comparing traffic policing and traffic shaping for bandwidth limiting. *Document ID* 19645 ([n. d.]), 22–42.

[4] Sivasankar Radhakrishnan, Yilong Geng, Vimalkumar Jeyakumar, Abdul Kabbani, George Porter, and Amin Vahdat. 2014. SENIC: Scalable NIC for End-Host Rate Limiting. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. USENIX Association, Seattle, WA, 475–488. https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/radhakrishnan

[5] Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C Snoeren. 2007. Cloud control with distributed rate limiting. In *ACM SIGCOMM Computer Communication Review*, Vol. 37. ACM, 337–348.

[6] Ahmed Saeed, Nandita Dukkipati, Vytautas Valancius, Carlo Contavalli, Amin Vahdat, et al. 2017. Carousel: Scalable traffic shaping at end hosts. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 404–417.