# Improving Latency and Burst-tolerance for Software Rate Limiters*

Keqiang He[†] Weite Qin[‡] Qiwei Zhang[§] Wenfei Wu[◦] Junjie Yang[‡]
Tian Pan[‡] Chengchen Hu[§] Jiao Zhang[‡] Aditya Akella[†] Ying Zhang[$]
[†]UW-Madison [‡]BUPT [§]XJTU [◦]Tsinghua [$]HP Labs
{keqhe,akella}@cs.wisc.edu {pan,jiaozhang}@bupt.edu.cn
wenfeiwu@outlook.com chengchenhu@xjtu.edu.cn ying.zhang13@hpe.com

## 1. INTRODUCTION

Today's cloud provisions resources (e.g., CPU, memory, and network) to tenants in a flexible way, which is an important reason for its success. The network resource is usually provided by reserving bandwidth [2]. However, tenants' applications may have diverse service level requirements such as low latency and low loss rate, which two are usually offered by best effort without guarantee.

According to our measurement in a public cloud [1], the provider cannot simultaneously satisfy the three service-level-agreement (SLA) requirements from tenants, i.e., bandwidth guarantee, low latency, and low loss rate. In details, current bandwidth allocation mechanisms utilize *software rate-limiters* such as Linux Traffic Control (TC) [3]. However, software rate limiters either uses traffic policing (i.e., dropping packets when packet arrival rate is above the desired rate) or traffic shaping (i.e., queueing packet in a large queue to absorb burst and send packet to the network based on token and bucket algorithms). Thus, bandwidth allocation, low latency and low loss rate can be not achieved at the same time.

Previous works mainly focused on solving "in-network" queueing latency (i.e., latency in the switches) [5, 6, 7], little research effort has been done to solve the latency, packet loss and burstiness issues for the software rate limiters on the end-host. To this end, we propose Combined Congestion Control ($C^3$), a scheme that combines rate limiter queueing information and the ECN signals from the switches in the network to perform congestion control in the virtualization layer. Congestion control is enforced in the virtualization layer via modifying TCP receiver's advertised window (RWND) [6]. $C^3$ reduces end-to-end latency, loss rate and improves burst-tolerance for software rate limiters. There are a few challenges to achieve the benefits of $C^3$. First, each flow's incoming packets and outgoing packets have diverged datapaths in virtualization layer—software rate-limiters can only observe outgoing packets of a flow [9]. Thus, $C^3$ needs to combine the observed congestion information (outgoing) and apply congestion control to incoming packets in the reverse direction. Second, traffic in software rate-limiters directly comes from TCP/IP stack and it tends to be very bursty. Thus, $C^3$ needs to overcome the drastically oscillating rate limiter queue length and optimize the congestion control algorithm to avoid throughput oscillation. We did a proof-of-concept implementation of $C^3$ and our preliminary experiment results demonstrate that $C^3$ ensures the goals of bandwidth guarantee, low latency and low packet drop rate can be achieved simultaneously.

*Keqiang He et al. are student authors

## 2. RATE LIMITERS AND THEIR LIMITATIONS

Software Rate limiters, e.g., Linux Traffic Control (TC), are the commonly used methods to provide per-VM/Container bandwidth allocation because of their flexibility and scalability. In Linux traffic control, there are two kinds of rate limiting methods. First is traffic shaping. In traffic shaping, packets are first pushed into a queue and delayed, then packets are scheduled and sent to the network based on token and bucket-based mechanisms. Shaping traffic ensures the traffic speed meets a desired rate. Second method is traffic policing. Traffic policing monitors the packet arrival rate, it only pushes a packet into the queue when traffic rate is not above the desired rate, otherwise the packet is dropped. Policing is a less accurate and less effective rate limiting method [4].

We quantified the latency caused by software rate limiters on a real testbed. We use Linux HTB as our rate limiter [8]. Linux HTB is a traffic shaping scheme. We use two bare metal machines (one traffic sender and one receiver) which are connected to the same 10G switch in CloudLab. We install Open vSwitch (OVS) and configure rate limiters. We send both large background flows and mice flows through rate limiters. Figure 1 and Figure 2 show TCP RTT results when we configure one rate limiter and two rate limiters for the 10G NIC. In both cases, we observe that TCP RTT is increased significantly (10X-20X) due to rate limiters on the end-host.

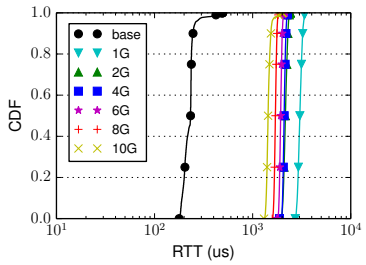


Figure 1: One rate limiter.

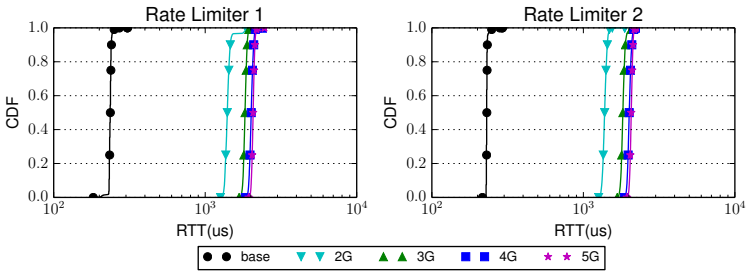


Figure 2: Two rate limiters with the same minimum rate.

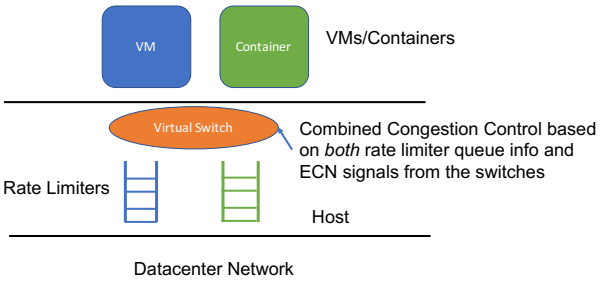


Figure 3: $C^3$ high-level architecture.

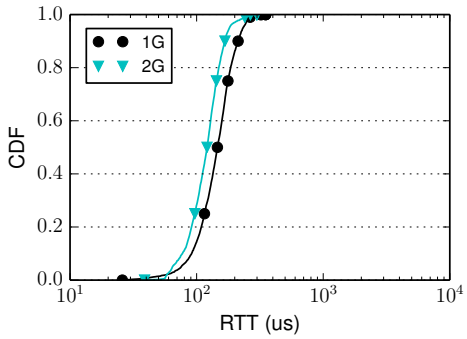


Figure 4: RTT of packets going through software rate limiter after applying $C^3$.

## 3. $C^3$ DESIGN

We propose Combined Congestion Control ($C^3$, shown in Figure 3). $C^3$ is based on traffic shaping-based rate limiting schemes. The idea is that we *utilize traffic shaper's large queue to absorb traffic burst while maintaining the queue occupancy at low level via congestion control in the virtualization layer in the virtual switch*. $C^3$ combines both queueing information in software rate limiters and ECN signals from the switches to perform congestion control. There are two challenges to implement $C^3$. The first is that to perform congestion control in the virtualization layer, we need to monitor both incoming traffic and outgoing traffic. However, packets in the rate limiter are outgoing while incoming packets do not go through rate limiters. The second challenge is software rate limiter queue oscillation is more severe compared with switch queue because it is at the traffic source. So our congestion control algorithm should be optimized to achieve low latency while avoiding cutting congestion window too aggressively.

We implemented $C^3$ in OVS and Linux HTB. HTB exposes its instantaneous queue length to OVS. In OVS, we perform congestion tracking and congestion control [6]. The congestion control algorithm takes multiple kinds of congestion information (software rate limiter queueing and ECN signals from switches). Finally, $C^3$ enforces congestion control in the virtualization layer via modify RWND field in the packet header.

## 4. EXPERIMENT RESULTS

We set up two bare metal servers in CloudLab. The two servers are connected to the same 10Gbps switch. We set up Linux HTB rate limiters to different rates on the sender side. After running $C^3$, we are able to achieve very low latency, low loss rate for software rate limiters while maintaining the desired bandwidth guarantee. Figure 4 shown TCP RTT results caused by Linux HTB when rate limiter rate is 1Gbps and 2Gbps. For example, when the rate limiter speed is 1Gbps, TCP's throughput is around 956Mbps while TCP RTT is less than 200 microseconds.

## References

[1] Cloudlab. https://www.cloudlab.us/.
[2] Google compute engine. https://cloud.google.com/compute/.
[3] Linux traffic control. http://tldp.org/HOWTO/Traffic-Control-HOWTO/elements.html.
[4] Open vswitch quality of service (qos). http://docs.openvswitch.org/en/latest/faq/qos/.
[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center tcp (dctcp). In *SIGCOMM*, 2010.
[6] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella. Ac/dc tcp: Virtual congestion control enforcement for datacenter networks. In *SIGCOMM*, 2016.
[7] R. Mittal, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats, et al. Timely: Rtt-based congestion control for the datacenter. In *SIGCOMM*, 2015.
[8] S. Radhakrishnan, Y. Geng, V. Jeyakumar, A. Kabbani, G. Porter, and A. Vahdat. Senic: Scalable nic for end-host rate limiting. In *NSDI*, 2014.
[9] W. Wu, K. He, and A. Akella. Perfsight: Performance diagnosis for software dataplanes. In *IMC*, 2015.