Delay-based virtual congestion control in multi-tenant datacenters

To cite this article: Yuxin Liu et al 2018 IOP Conf. Ser.: Mater. Sci. Eng. 322 052040

View the article online for updates and enhancements.

Related content

- Implementation of the Algorithm for Congestion control in the Dynamic Circuit Network (DCN)
  H S Nalamwar, M A Ivanov and G U Buddhawar

- A high-linearity and high-resolution delay line structure with a calibration algorithm in delay-based LINC transmitters
  Han Yue, Qiao Shushan and Hei Yong

- Feedback control scheme of traffic jams based on the coupled map car-following model
  Zhou Tong, Sun Di-Hua, Zhao Min et al.
Delay-based virtual congestion control in multi-tenant datacenters

Yuxin Liu¹, Danhong Zhu¹ and Dong Zhang¹,*

¹ School of Mathematics and Computer Science and Technology, Fuzhou University, Fuzhou, China

*Corresponding author e-mail: zhangdong@fzu.edu.cn

Abstract. With the evolution of cloud computing and virtualization, the congestion control of virtual datacenters has become the basic issue for multi-tenant datacenters transmission. Regarding to the friendly conflict of heterogeneous congestion control among multi-tenant, this paper proposes a delay-based virtual congestion control, which translates the multi-tenant heterogeneous congestion control into delay-based feedback uniformly by setting the hypervisor translation layer, modifying three-way handshake of explicit feedback and packet loss feedback and throttling receive window. The simulation results show that the delay-based virtual congestion control can effectively solve the unfairness of heterogeneous feedback congestion control algorithms.

1. Introduction

With the evolution of cloud computing and big data, the datacenter has become the most efficient and promising infrastructure to support data storage and deploy diversified network services. The traditional data center architecture is difficult to meet the network applications' high requirement for the quality of various resource service (storage, computing power, bandwidth and delay) in pace with the diversification of network applications and the explosion of data volume [1]. Moreover, the low utilization rate relatively higher total operating costs of dedicated datacenter resources. Then the emergence of virtualized data center provides good technology for the implementation of multi-tenant data centers, sharing the same physical infrastructure between different tenants to run virtual data center workloads, and seamlessly porting applications and services to the cloud. Multi-tenant data center has become the major direction of data center development because of its flexible management, strong scalability, high resource utilization and energy efficiency. Congestion control is the foundation and key for the multi-tenant datacenter. Multi-tenant datacenters implement independent congestion control algorithms for different tenants so that they exist friendly conflicts [2]. This means multi-tenant datacenter managers must ensure that a variety of congestion control algorithm run friendly in parallel in the shared data center networks.

Three ways to avoid mutual interference among different multi-tenant datacenters congestion control algorithms are as follows, 1) Strictly dividing the bandwidth between the datacenter tenants and giving each tenant a fixed allocation [3]. 2) Adjusting the fair rules of the tenants on the data center switch, such as using different tag threshold in the same queue or a separate queue [4,5]. 3) According to the feature that traffic must go through the hypervisor, using the hypervisor conversion layer to ensure that the entire data center uses one single optimal congestion control algorithm, while the different tenants still use their own congestion control algorithm configuration in the same time.
Namely, the hypervisor transforms the guest congestion control algorithm into the underlying algorithm [6] which is used in the datacenter network. The solution is Virtual Congestion Control (vCC). [7] proposed a similar solution --AC/DCTCP. [6] and [7] together convert heterogeneous congestion control into DCTCP [8]. However, explicit feedback congestion control has limitations on traditional datacenters [9]. To sum up, this paper proposes the Delay-based virtual congestion control (Delay-based vCC) for multi-tenant datacenters. By modifying three-way handshake and throttling the receive window through altering the TCP header, then the multi-tenant heterogeneous feedback congestion control protocols are unified into delay feedback to solve the unfairness of RTT (round-trip time).

2. The introduction of virtual congestion control

Each host in the multi-tenant data center has multiple virtual machines managed by the hypervisor. The hypervisor receives packets from the local virtual machine through internal software virtual switches and forwards it to another local virtual machine, or other hypervisor.

Multi-tenant technology allows a large number of tenants to share the same software and hardware resources, each tenant can use resources customize software services on demand, without affecting other tenants. The multi-tenant data center hosts a large number of different tenants on a shared infrastructure, and allocate the computing, storage, and network resources for tenants equitably.

Multi-tenant datacenters represent an extremely challenging networking environment. In addition to ensuring the bandwidth and fairness of different tenants, the workload of the tenant should be migrated from the enterprise network to the datacenter [11]. The congestion control algorithm proposed by [10] failed to solve the complex interaction between the congestion control algorithms of different tenants in multi-tenant datacenter. Therefore, the virtual congestion control came into being.

In the multi-tenant data center different tenants use different congestion control algorithms because of their own virtual machines. The hypervisor can transform the different congestion control algorithms into the unified underlying algorithm which data center network used. The process calls Virtual congestion control. It can achieve seamless support for any congestion control algorithm.

The virtual congestion control transforms the heterogeneous congestion control modes into the same congestion feedback mode and achieves the purpose of virtual congestion control. By throttling the receive window [12,13] and modifying the three-way handshake, the underlying congestion control algorithm is implemented in the hypervisor, and it is decoupled with the congestion control algorithm in the tenant management system.

3. Delay-based vCC

The delay-based virtual congestion control can convert the feedback of multi tenants heterogeneous congestion controls into delayed feedback universally by modifying the three handshake and throttling the receiving window and modifying the TCP header at the translate layer.

3.1. Translating delay feedback to explicit feedback

When the congestion control algorithm used by tenants implements explicit feedback, the translation layer uses a translating delay feedback to explicit feedback scheme, as shown in Fig. 1.

In the three-way handshake, 1) in the first handshake, the translation layer shields the ECE and CWR (Congestion Window Reduced) flag in TCP header which was sent to the network side from the guest, and adds the ECE flag in TCP header was sent back the guest. 2) In the second handshake, the translation layer receives the normal data packet, and shields the ECT flag TCP header which was sent to the network side from the guest, and the unchanged header which was sent back to the guest. 3) In the third handshake, the translation layer modifies the RWIN in the TCP header according to its window state, and the translation layer hides CWR on the next outgoing packet.

The receiving window calculation formula refers to the calculation of the TCP Vegas [14] congestion window, and the RWIN adjusts the receive window size according to the diff value.
3.2. Translating Delay Feedback to Packet Loss Feedback
When the congestion control algorithm used by tenants implements packet loss feedback, the translation layer uses a translating delay feedback to packet loss feedback scheme, as shown in Fig. 2.

In the three handshake, 1) the first handshake and the second remains unchanged. 2) In the third handshake, in order to reduce the receiving window, the translation layer modifies the RWIN in the TCP header according to its window state and Formula (1).

4. Delay-based vCC
In order to verify the network performance and fairness of Delay-based vCC, we modify the Linux kernel code to achieve Delay-based vCC. Delay-based vCC related experiments rely on Linux virtual machine which system kernel version is 3.19, and use mininet as network simulation tool, iperf as traffic generation tool and tshark as packet capture analysis tool. The experimental topology is that 10 hosts communicate with each other through switches.

4.1. RTT unfairness

4.1.1. TCP Vegas vs TCP Reno unfairness experiments. The link bandwidth of the experiments is 100 Mbps and the delay is 0.25 ms. The exchange queue of experiments use RED, and the specific parameters set as shown in Table 1. TCP Vegas is chosen as the congestion control algorithm of RTT flow, and TCP Reno is chosen as the congestion control algorithm of non-RTT flows.

\[
\begin{align*}
\text{RWIN} &= \begin{cases} 
\text{cwnd} - 1, & \text{diff} > \beta \\
\text{cwnd}, & \text{cwnd} < \text{diff} < \beta \\
\text{cwnd} + 1, & \text{diff} < \alpha 
\end{cases} 
\end{align*}
\]

(1)

RWIN represents the congestion window size of the current TCP, \(\alpha\) and \(\beta\) are the threshold values, consistent with the Linux 3.19 kernel TCP Vegas parameter, the value of \(\alpha\) is 2, and that of \(\beta\) is 4, and diff representing the difference between the expected rate and the actual rate, as shown in (2).

\[
\text{diff} = \frac{\text{Expected}}{\text{Actual}}
\]

(2)

The calculation of the expected rate and the actual rate is shown in (3) and (4).

\[
\begin{align*}
\text{Expected} &= \frac{\text{cwnd}}{\text{BaseRTT}} \\
\text{Actual} &= \frac{\text{cwnd}}{\text{RTT}}
\end{align*}
\]

(3)

(4)
Table 1. RED parameters.

| Parameter | REDmin | REDmax | REDlimit | REDburst | REDprob |
|-----------|--------|--------|----------|----------|---------|
| value     | 90000  | 90001  | 1M       | 61       | 1.0     |

Fig. 3 illustrates the time-series of goodput by three experiments and analyzes the unfairness between RTT and non-RTT flows. As shown in Fig. 3 (a), 1 TCP Reno flows can share bandwidth with 9 RTT flows, but the Reno flow throughput is significantly better than the RTT flow with the ratio about 3. In Fig. 3 (b), the experiment uses 5 Reno flows and 5 RTT flows, the throughput of the Reno flows is better than that of the RTT flows. The ratio of Reno flows to RTT flow is 1.75. In Fig. 3 (c), there are only 1 RTT flows in the 10 flows, and the RTT flows throughput are lower than the Reno flows, and the ratio is about 1.5. Because the bandwidth of Reno is not competitive, Reno has little impact on RTT flows.

4.1.2. TCP Vegas vs DCTCP unfairness experiments. DCTCP is used as the congestion control algorithm of non-RTT flow and other parameters are unchanged. The experiments of link bandwidth 100Mbps are carried out respectively. The experimental result of the link bandwidth of 100MB is shown in Fig. 4.

Because of DCTCP which has the strong bandwidth competition, the throughput of DCTCP is obviously better than that of RTT in three different experiments. In Fig. 4 (a), only 1 DCTCP flow in this experiment, the throughput ratio of DCTCP flow to RTT flows is 3; In Fig. 4 (b), there are 5 DCTCP 5 flows and RTT flows, the throughput ratio is 2.75; In Fig. 4 (c), the experiment only has 1 RTT flow, and the throughput ratio is 13.

4.2. Restoring Fairness with Delay-based vCC

Delay-based vCC converts packet loss feedback and explicit feedback into delayed feedback congestion protocols in order to solve the unfairness of RTT. We set up two experiments to verify the validity.

Figure 3. Unfairness experiments between RTT and Reno flows (100Mbps)

(a) 9 RTT VS. 1 Reno (b) 5 RTT VS. 5 Reno (c) 1 RTT VS. 9 Reno

Figure 4. Unfairness experiments between RTT and DCTCP flows (100Mbps)

(a) 9 RTT VS. 1 DCTCP (b) 5 RTT VS. 5 DCTCP (c) 1 RTT VS. 9 DCTCP
of Delay-based vCC. The first experiment is similar to the experiments in section A. It uses 10 TCP flows include 1 RTT flow and 9 non-RTT flows. In the second experiment the non-RTT flows is converted into the virtual-RTT flows by Delay-based vCC. So 1 RTT flow VS. 9 virtual-RTT flows are used. Besides, TCP Reno and DCTCP are used as the congestion control protocols for non-RTT flows.

4.2.1. TCP Vegas VS. TCP Reno Virtual congestion control experiments. There are 1 RTT flow and 9 non-RTT flows in this experiment. The link bandwidth is 100Mbps. In addition, TCP Reno is chosen as the congestion control protocol for non-RTT flows. Other experimental parameters are consistent with TCP Vegas VS. DCTCP unfairness experiments. Fig. 5 (a) shows RTT unfairness, the throughput ratio of Reno flows to RTT flows is 1.5. But Fig. 5 (b) indicates the throughput ratio of virtual-RTT flow to RTT is improved to 1.14. Obviously the throughput has been very close. So the problem of RTT Fairness can be effectively resolved by Delay-based vCC.

4.2.2. TCP Vegas VS. TCP Reno Virtual congestion control experiments. We use DCTCP as the non-RTT flow congestion control algorithm. The parameters are unchanged. The experimental results are shown in Fig. 6. Compared with the TCP Reno flow, Delay-based vCC is more effective because of the more competitive bandwidth of DCTCP. Fig. 6 (a) shows the RTT unfairness, the throughput ratio of DCTCP flows to RTT flows is 11. But the throughput ratio becomes closer shown in Fig. 6 (b). Therefore, the unfairness between DCTCP flow and RTT flow can be resolved by Delay-based vCC.

4.3. Restoring Fairness with Delay-based vCC
This section describes the contrast experiments of Delay-based vCC and vCC. In vCC experiments the congestion control of ECN flow is DCTCP and the congestion control of non-ECN is TCP Reno. TCP Vegas is used as the congestion control of non-ECN flow in Delay-based vCC experiments. Other parameters are consistent with the above experiments. The experimental results are shown in Fig. 7.
Fig. 7 shows the experimental results of vCC. TCP Vegas was converted into the virtual-ECN flows by using the vCC solution. And virtual-ECN flow is significantly increased. But there is still a gap compared with DCTCP flow. The throughput ratio of DCTCP flow to virtual-ECN is close to 1.8. But the throughput ratio of virtual-RTT flows to RTT flow in Delay-based vCC is close to 1. From the two experiments we can see Delay-based vCC is better than vCC on the problem of coexistence of Vegas and DCTCP. So Delay-based vCC is more suitable for non-ECN datacenters.

5. Conclusion
The TCP friendliness of congestion control among different tenants is the fundamental problem of virtual data center transmission. The delay-based virtual congestion control converts the feedback of heterogeneous congestion control into delayed feedback control by modifying the three handshake and throttling the receiving window and modifying the TCP header at the translate layer. Thus, the newly deployed congestion control algorithm needn’t consider the TCP friendliness. But Delay-based vCC doesn’t consider the deadline-aware flow enough. How to introduce deadline-aware into virtual congestion control for multitenant datacenter is a further research direction.

Acknowledgments
This work was financially supported by the Guide key project of Fujian Province (Grant No.2016Y0060).

References
[1] Bari M F, Boutaba R, Esteves R, et al. Data Center Network Virtualization: A Survey[J]. IEEE Communications Surveys & Tutorials, 2013, 15(2):909-928.
[2] Glenn Judd. Attaining the promise and avoiding the pitfalls of TCP in the datacenter. USENIX NSDI, 2015.
[3] Zahavi E, Shpiner A, Rottenstreich O, et al. Links as a Service (LaaS): Guaranteed Tenant Isolation in the Shared Cloud[C]/Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems. ACM, 2016: 87-98.
[4] Küehlewind M, Wagner D P, Espinosa J M R, et al. Using data center TCP (DCTCP) in the internet[C]/2014 IEEE Globecom Workshops (GC Wkshps). IEEE, 2014: 583-588.
[5] Judd G. Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter[C]/12th US-ENIX Symposium on Networked Systems Design and Implementation (NSDI 15). 2015: 145-157.
[6] Cronkite-Ratcliff B, Bergman A, Vargaftik S, et al. Virtualized congestion control[C]/Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference. ACM, 2016: 230-243.
[7] He K, Rozner E, Agarwal K, et al. AC/DC TCP: Virtual congestion control enforcement for datacenter networks[C]/Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference. ACM, 2016: 244-257.
[8] Alizadeh M, Greenberg A, Maltz D A, et al. Data center tcp (dctcp)[C]/ACM SIGCOMM computer communication review. ACM, 2010, 40(4): 63-74.
[9] Devkota P, Reddy A L N. Performance of Quantized Congestion Notification in TCP Incast Scenarios of Data Centers[C]/IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE Computer Society, 2010:235-243.
[10] Carvalho T, Kim H S, Neves N. PACE your network: Fair and controllable multi-tenant data center networks[C]/IEEE International Conference on Communications. IEEE, 2013:3726-3731.
[11] Koponen T, Amidon K, Balland P, et al. Network virtualization in multi-tenant datacenters[C]/USENIX Conference on Networked Systems Design and Implementation. USENIX Association, 2014:203-216.
[12] Shrikrishna Karandikar, Shivkumar Kalyanaraman, Prasad Bagal, and Bob Packer. TCP rate control. ACM SIGCOMM, 2000.
[13] Huan-Yun Wei, Shih-Chiang Tsao, and Ying-Dar Lin. Assessing and improving TCP rate shaping over edge gateways. IEEE Trans. Comput., 53(3):259–275, 2004.
[14] Brakmo L S, Peterson L L. TCP Vegas: End to end congestion avoidance on a global Internet[J]. IEEE Journal on selected Areas in communications, 1995, 13(8): 1465-1480