Stem: A Table-Based Congestion Control Framework for Virtualized Data Center Networks

Jie Wu[1](✉), Binzhang Fu[1], and Mingyu Chen[2]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
jillwu1991@gmail.com
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Congestion control is one of the biggest challenges faced by networks, and is enlarged in current data centers due to its large scale and variety of applications. Generally, different kinds of applications prefer different congestion control solutions. However, current mechanisms often exploit customized framework and require dedicated modules to realize certain functions, then deploying multiple solutions at the same time or reloading another solution when a new application is served is almost impossible. To address this problem, this paper proposes a solely table-based congestion control framework which is compatible with most of the current congestion control solutions. We implement the prototype with Open vSwitch, and the experiments results show that the proposed Stem could achieve the above claimed benefits with negligible overhead.

1 Introduction

Cloud computing has been viewed as an efficient way to provide on-demand IT capability to emerging applications, such as AI, IoT and 5G. However, the Cloud computing itself is still facing a lot of challenges. For example, traditionally VMs work in the same address space as the physical network and leads to the so-called address space problem [4]. To address these problems, the most efficient way is to exploit network virtualization.

Unfortunately, virtualized network leads to new challenges. For example, since different tenants may adopt different network stacks and configurations themselves, different reactions will be taken when congestions happen, which leads to "unfairness" in network resource utilization, especially when there are malicious tenants with specially modified network stacks. Besides that, when new network technology is found, it is hard for legacy applications to take advantage of it [2]. The last but not the least, Cloud data centers featuring multi-tenancy have to satisfy the customized requirements for network, like QoS,

This work was supported by the National Key Research and Development Program of China (13th Five-Year Plan) under Grant No. 2016YFB1000200, by National Natural Science Foundation of China (NSFC) under Grant No. 61331008, and Grant No. 61521092.

X. Shi et al. (Eds.): NPC 2017, LNCS 10578, pp. 122–126, 2017.
DOI: 10.1007/978-3-319-68210-5_12

network services. Different tenants may want to do congestion control for distinct kind of traffic or distinguish congestion control laws for traffic with different preferences. These all show that there is an urgent need of the capability to specify congestion control mechanism for network operators.

The state-of-the-art solutions, such as [2,3], could solve part of the above challenges, but all failed to provide a highly extendable and unified framework to provide the Cloud vendors ability to dynamically provide tenant-specific congestion control mechanisms. To this end, this paper proposes the Stem, a solely table-based framework to implement congestion control mechanisms in virtualized data centers.

Generally, Stem takes a rate-based solution for a flexible rate-adjusting. To make the framework unified, we define a new OpenFlow action, namely EXEC_FUNC, to enable users to register their own congestion control logic. To validate the proposed solution, we build a prototype, which implements a DCTCP [1]-like congestion control solution, by exploiting the Open vSwitch. To this end, a new OpenFlow group, namely the "specific-match group", is defined to collect stateful statistics.

We summarize our contributions as follows:

1. A table-based solution is proposed for implementing flexible congestion control mechanisms for virtualized Cloud data centers;
2. A new OpenFlow action, namely the EXEC_FUNC, is defined to enable user-defined control logics with a uniform interface;
3. A new OpenFlow group, namely the specific-match group, is defined to collect stateful statistics.

2 Architecture of Stem

2.1 Overview of Stem

Stem consists of a sender-side module and a receiver-side module. Sender-side module is in charge of rate-adapting following user-specified congestion control law. The receiver-side module is responsible for recording congestion information, calculating the congestion status and setting the calculated status on feedbacks to sender. In the following, we will show how Stem could be realized in a solely table-based way. Figure 1 shows the architecture of Stem. Packets are enforced congestion indication support, ECN we use in Stem, at sender side. They will be marked Congestion Experienced(CE) when congestion is encountered in network. As packets arrive at receiver-side, congestion information is recorded in our proposed new OpenFlow group, namely *specific-match group*. Then, the congestion indications are masked to be invisible as upper VMs may not support ECN or as there shouldn't be further reaction to CE mark for those VMs who support ECN in order to achieve fairness among VMs, after which packets are sent to upper VMs. Stem gets statistics from *specific-match group*, calculates ECN mark fraction and sets that in the encapsulation header of ACK packets. When sender-side server receives ACK packets, it extracts the ECN mark fraction and send the packets to upper VMs after congestion indications get masked. Sender-side module calculates a new sending rate following tenant-specific congestion control algorithm, and enforces the new rate for Sender VM.

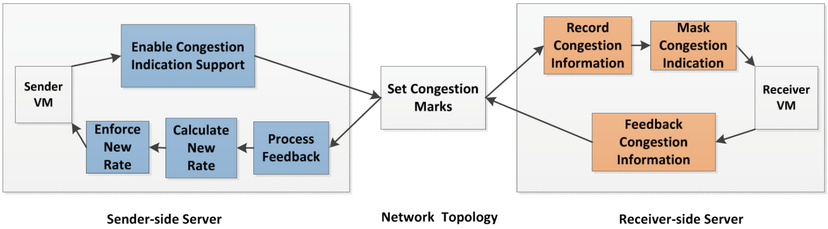

Fig. 1. Components of typical congestion control mechanism

2.2 A New Action: EXEC_FUNC

As the goal of Stem is to provide a framework of table-based congestion control, we provide a flexible and extendable way for users to add their own congestion control logic. To this end, we propose a userspace action: EXEC_FUNC(*function*), with the connotation of executing a user specified processing *function* . EXEC_FUNC action has only a parameter *function* which is simply a string. EXEC_FUNC extracts the string from OpenFlow message and developers can define their own function resolving the actual parameters of *function* to realize certain logic. We add in two functions in this use case. One is used for setting ECN mark fraction, and the other is used to do rate-limiting.

2.3 A New Group: Specific-Match Group

Stem takes a structure to group two flow entires, one for packets marked congested and the other for the rest, to collect congestion statistics and calculate the fraction of ECN-marked packets. When a packet executes the "group" action, it goes to the related "group" structure to execute actions specified in the group structure there. Basic components in a group structure are buckets. One bucket corresponds to one action(or action set). Originally, "select" group of OpenFlow has implemented a selection-method called "hash" to select a bucket. However, hash function is not flexible enough for us to implement network functions at most times. To address this problem, we add in "match" logic to implement a new selection method "specific-match". When a packet that has done a general match is directed to a "specific-match" group, it will match on the "match" part of the bucket to find the proper bucket to execute.

Now let's see how we use "specific-match" group to record congestion status. See Fig. 2, we create groups at receiver side for each sender VM to record the number of packets marked CE. There are two buckets in this group. One matches *tun_ecn*, ECN field on the outer header of tunnel. The other has no match field that it matches the rest of packets in this group. The "note" action is a kind of action that does nothing at all. We use "note" because we only want the hit statistics of these two buckets rather than executes a specific action. Obviously, if we use the hit times of the group to divide that of the bucket matches *tun_ecn*, we can get accurate ECN mark fraction at receiver side.

```
group_id=1,type=select,selection_method=specific-match,
bucket=bucket_id=1,match(tun_tos=3), actions=note:00,
bucket=bucket_id=2,actions=note:00
```

Fig. 2. Specific-match group to get ECN mark fraction

2.4 Congestion Control Algorithm

In the use case of solving "unfairness" in network resource utilization, Stem takes a DCTCP-like algorithm as an example. Particularly, the rate-decreasing part of DCTCP is kept while the increasing part is optimized by taking network congestion degree into account in the slow start phase, $rate \leftarrow 2 * rate/(1+\alpha)$. The calculated sending $rate$ here is inversely proportional to α, congestion degree estimated for network [1]. That is, when network is more congested, Stem lowers the increase of sending rate more.

3 Evaluation

3.1 Environment Setup

We implemented a prototype in the OVS Version 2.4.0. To make the tradeoff between responsiveness and performance, we set the sampling proportion to 3000:1. Servers we use are HuaWei RH2258, with CPU of Intel(R) Xeon(R) E5645, 12 CPU cores, 2 hardware threads per core, 32G memory and all NICs are 1 Gbps. We take one server installed OVS as the switch(we call it OVS switch later) to connect servers and use TC RED for ECN marking. VMs in servers are emulated by qemu(with KVM enabled) and connected into network by Stem, or unmodified OVS as the baseline. The encapsulation protocol we take is VXLAN for the Overlay network virtualization environment.

3.2 Evaluation Results

Overhead. We run a single long iperf3 flow between a sender-side VM and a receiver-side VM, in that there isn't congestion in network. Though only rate-increasing logic is executed in this experiment, we don't think there is obvious difference in the overhead between rate-increasing and rate-decreasing logic. As there's a 50 bytes outer header for VXLAN, we set the "-M" (MSS) option of iperf3 to 1300 to not exceed MTU. The experiment result shows that Stem not only keeps the line rate of network but also gets a more stable rate. Due to the space limit, the figure is omitted.

Fairness. We apply all VMs with the same congestion control law in Sect. 2.4 to achieve fairness. We run VM1 in Server1 with network stack CUBIC and VM2 with DCTCP in Server2. We set the threshold of ECN marking for RED to 20 packets. VM1 runs an iperf flow to VM3 in Server3 and after 5 s VM2 runs an iperf flow to VM3 too. From Fig. 3, the experiment taken in an unmodified

OVS environment, we can see, DCTCP flow almost makes CUBIC bandwidth decrease to 0. Figure 4 is the result of the same experiment we take with Stem, we can see that fairness can be achieved in use of Stem.

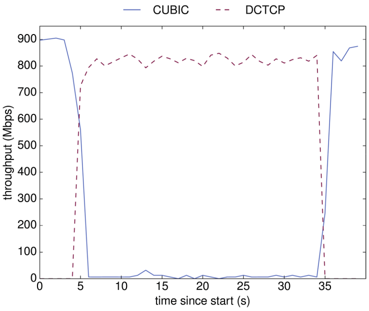

Fig. 3. DCTCP and CUBIC fairness in unmodified OVS environment

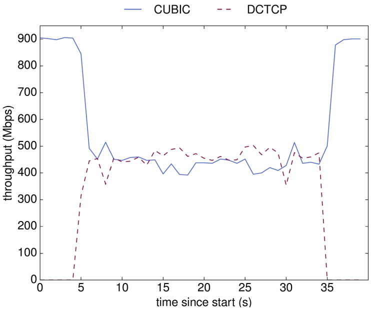

Fig. 4. DCTCP and CUBIC fairness in Stem

4 Conclusion

In this paper, we propose Stem, a table-based framework of user-defined congestion control for multi-tenant cloud computing. To achieve the goal, Stem first defines an OpenFlow action: EXEC_FUNC(*function*), which works as an interface to add new actions in more easily. After that, Stem proposes "specific-match group", which makes it possible to record the congestion status of any traffic defined by an OpenFlow entry. Stem implements a rate-based DCTCP-like algorithm in vSwitch to enforce the same congestion control law for all the VMs attached to that switch to get fairness.

References

1. Alizadeh, M., Greenberg, A., Maltz, D.A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., Sridharan, M.: Data center tcp (dctcp). In: ACM SIGCOMM Computer Communication Review, vol. 40, pp. 63–74. ACM (2010)
2. Cronkite-Ratcliff, B., Bergman, A., Vargaftik, S., Ravi, M., McKeown, N., Abraham, I., Keslassy, I.: Virtualized Congestion Control. In: Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference, pp. 230–243. ACM (2016)
3. He, K., Rozner, E., Agarwal, K., Gu, Y.J., Felter, W., Carter, J., Akella, A.: AC/DC TCP: virtual congestion control enforcement for datacenter networks. In: Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference, pp. 244–257. ACM (2016)
4. Koponen, T., Amidon, K., Balland, P., Casado, M., Chanda, A., Fulton, B., Ganichev, I., Gross, J., Ingram, P., Jackson, E., et al.: Network virtualization in multi-tenant datacenters. In: 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pp. 203–216 (2014)