AC/DC TCP: Virtual Congestion Control Enforcement for the Datacenter Networks[1]

Reviewed by
Xintong Wang and Jack Kosaian

Summary

The authors propose and develop Administrator Control over Datacenter TCP (AC/DC) to allow network administrators to control tenant TCP stacks without making modifications to virtual machines (VMs) or datacenter hardware. AC/DC places per-flow congestion control at the virtual switch (vSwitch) on top of which multiple tenants run. Individual tenants can use their own congestion control algorithms, but control is ultimately determined by the algorithm running on the vSwitch. Flows are altered at the vSwitch to support different parameters required to use AC/DC's chosen congestion control algorithm for that flow. For example, if a flow is to be controlled using DCTCP, an ECN field will be added to packets in the flow and used by the receiving vSwitch. By adding and removing fields from flows, AC/DC uses its own congestion control algorithms to determine network congestion. In order to enforce congestion-based rate limiting, AC/DC vSwitch's modify the RWND advertised to sending tenants. The authors implemented AC/DC on a physical testbed of 17 IBM servers configured in various topologies to determine its ability to reduce packet delays, achieve high throughput, fairly allocate bandwidth, and quickly respond to flow churn. Through their evaluations, the authors find AC/DC to achieve packet delays and network throughput comparable to a cluster with each server individually running DCTCP. Further, the authors show that AC/DC adds less than one percent computational overhead to vSwitches.

[1]He, K. et al., "AC/DC TCP: Virtual Congestion Control Enforcement for the Datacenter Networks," Proc. of ACM SIGCOMM '16, 45(4): 244-257, Aug. 2016.

Contributions, Highlights and Novelties

Below are the contributions and highlights we appreciated while reading the paper.

1. The Enforcement of Per-flow Congestion Control in vSwitch and its Benefits

Though mentioned in the paper, we want to stress the novelty, convenience and applicability of enforcing per-flow congestion control in vSwitch, especially given the constraints of leaving the VM and network hardware unchanged. We appreciate several benefits brought by this design. First, by supporting uniform congestion control across the datacenter, AC/DC TCP alleviates the problem of varying TCP stacks on the same fabric and thus *enhances fairness*. Second, different congestion control algorithms can be applied on the per-flow basis, which *enhances flexibility* to prioritize flows to meet certain requirements. Third, by taking advantages of the mature vSwitch technology and various existing congestion control algorithms, the proposed AC/DC TCP is more like a highly modular add-on that is easy to understand and *practical to implement* with relatively low labor and cost. Moreover, AC/DC TCP can co-exist with a bandwidth allocation scheme.

2. Clear Narrative of Motivation and Research Background

Before delving into the implementations of AC/DC TCP, the authors managed to give a clear and brief summary of the research motivation and background, and how their work naturally fits into the big picture. The authors first use examples (Section 2.1) to demonstrate that congestion in datacenter networks has been shown to arise at relatively low levels of network utilization and to cause significant packet drops. Then, the authors describe the current proposed solutions: transport protocols like TCP have evolved to include advanced congestion control algorithms to reduce the number of packet drops that a connection experiences due to congestion and to alleviate the network of its congested state. Among these congestion control algorithms, the authors adopted Data Center TCP (DCTCP) in AC/DC, as it has proved particularly effective in avoiding congestion in datacenter networks [2]. Based on the research background, the authors motivate their work and formulate their research problem well. They emphasize the necessity of running such congestion control algorithms atop operating systems with correct and up-to-date TCP implementations. However, it is unlikely that all among the myriad of VMs available to tenants in a public-cloud setting will have up-to-date TCP stacks. Thus, datacenter network administrators wishing to fine-tune the congestion control among tenants must either make modifications to VMs to support desired congestion control algorithms, or forbid the use of out-of-date VMs. We think the authors did a good job on building the background of their research problem and describing how their work can fill the gaps in current solutions.

3. Examples of Implementations of Per-flow Differentiation

We like the fact that instead of mentioning per-flow differentiation as a concept, the authors propose a priority-based congestion control algorithm to illustrate the usefulness of implementing different congestion control algorithms on a per-flow basis. The algorithm, together with its priority parameter, is straightforward and can demonstrate AC/DC TCP's power and flexibility to adjust to different requirements. Similarly, the authors also show that administrators can enable per-flow bandwidth allocation schemes by bounding RWND.

Possible Improvements and Extensions

There are a number of ways in which the authors' work could be improved upon or extended to strengthen the contributions listed above. This section outlines areas that could have improved the authors' work and offers avenues for extension of the work.

1. Evaluation with Flows that Cannot be Monitored by AC/DC

In Section 3.3 of their paper, the authors note that AC/DC's applicability is limited to unencrypted flows using TCP as their transport protocol. While TCP is a widely used protocol by datacenter applications, there exist scenarios where TCP is not the best transport protocol for a particular application. For example, applications seeking lower connection startup costs may prefer using a lighter transport protocol such as UDP, or may even implement their own transport protocols [3]. AC/DC is unable to control or monitor such applications.

Though it may be outside the scope of the authors' work to attempt to control congestion for arbitrary transport protocols, the authors' evaluation of AC/DC would have benefitted from an experiment in which some flows used a transport protocol that could not be monitored and controlled by AC/DC. This would help make clear how much of AC/DC's benefits stem from the opportunity to influence the congestion control algorithms of all flows running in a cluster. If the average flow completion time in such an experiment is significantly higher than the average flow completion time in experiments in which all flows use TCP, there would be substantial motivation for extension of AC/DC to control arbitrary applications or for integration of AC/DC with existing network-level congestion control systems.

2. Evaluation of Systems with Diverse Flow Types

In Section 3.4, the authors discuss AC/DC's ability to differentiate between flow types and enforce different congestion control algorithms depending on the flow type, but offer no evaluation of AC/DC when flows with different types are monitored concurrently. An interesting experiment to run might involve both intra-datacenter flows and WAN-destined flows simultaneously competing for bandwidth. In such a case, AC/DC should use a different congestion control algorithm for the different flows (e.g., DCTCP for intra-datacenter flows and CUBIC for WAN flows). It would be interesting to see how well AC/DC is capable of enforcing congestion control for these different algorithms at the same time. If AC/DC is unable to simultaneously support two distinct congestion control algorithms, its utility for general purpose datacenters is weakened.

3. Scaling the Number of Supported Tenants

On a virtualized datacenter node, multiple VM instances from different tenants will run atop the same vSwitch. Though AC/DC's design has been greatly influenced by the need to support multi-tenant systems, the authors do not present evaluation of AC/DC's ability to scale to support a large number of tenants on the same vSwitch. Such analysis would provide a better sense of AC/DC's applicability as a production datacenter service. If AC/DC cannot support a large number of concurrent VMs on a single vSwitch, its utility as a production service decreases significantly.

4. Integration of AC/DC with Load-Balancing Systems

A possible extension of AC/DC would be to integrate AC/DC with systems designed to provide load balancing for datacenter networks. Systems have been proposed to provide datacenter network load balancing based on congestion awareness. One such system, CONGA [1], provides congestion-aware load balancing without enforcing specific transport protocol usage among tenants. While CONGA uses congestion awareness to inform load balancing decisions, it does not explicitly attempt to mitigate congestion. Using AC/DC with CONGA could be an interesting way to provide both load balancing and congestion avoidance. Combining the two systems may provide additional benefits, as AC/DC's congestion measurements could potentially be used by CONGA to determine the global levels of congestion among links. Integrating AC/DC with CONGA could require significant changes to each system, as CONGA is designed to be agnostic to transport protocols being used, while AC/DC currently supports only TCP.

References

[1] Alizadeh, M., et al., "CONGA: Distributed Congestion-Aware Load Balancing for Datacenters," *Proc. of ACM SIGCOMM '14*, 44(4):503-514, Oct. 2014.

[2] Alizadeh, M., et al., "Data Center TCP (DCTCP)." *In SIGCOMM*. ACM, 2010.

[3] Ousterhout, J. et al., "The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM," *SIGOPS Oper. Syst. Rev.* 43, 4 (January 2010), 92-105.