CS540 - Paper Review Report # XII

Hailu Belay Kahsay - 20155624

Title: Data Center TCP (DCTCP)

Author: Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, Murari Sridharan

It is stated in the paper that today's state-of-the-art TCP protocol falls short for cloud data centers with diverse applications, which mix workloads that require small predictable latency with others requiring large sustained throughput. The paper measures data center traffic, over 2 months from over 6000 servers, in product environment and analyzes three different kinds of performance impairments: incast, queue buildup and buffer pressure. To address these issues, the authors propose data center TCP(DCTCP), a TCP-like protocol for data center networks, which uses Explicit Congestion Notification (ECN), in the network to provide multi-bit feedback to end hosts, and achieves better performance with less buffer space.

The new protocol attempts to improve the network latency required by some application, the protocol needs to be able to sustained high burst traffic and high utilization so that applications could continuously update their internal structures. By letting switches to notify senders when their queue size exceeds a pre-set limit, the proposed protocol allows senders to dial back based on the number of marked packets exceeding the limit. This allows for shorter queue times, leading to fewer packet drops, the bane of TCP.

The authors also mention other studies which either approach the problem differently (doesn't focus on reducing packet loss) or require more complex hardware (better than commodity switches) such as: Quantized Congestion Notification(QCN) and several TCP variants that aim to reduce queue lengths at routers: delay-based congestion control (Vegas and Compound TCP(CTCP)), explicit feedback schemes (RCP, VCP, and BMCC), and AQM schemes (RED and PI), improving TCP performance on paths with high bandwidth-delay product(HTCP, CUBIC TCP, and FAST TCP); but DCTCP provides real-time feedback.

The authors found that DCTCP delivers the same or better throughput than TCP, while using 90% less buffer space. Unlike TCP, DCTCP also provides high burst tolerance and low latency for short flows. In handling workloads derived from operational measurements, the authors found that DCTCP enables the applications to handle 10X the current background traffic, without impacting foreground traffic. Further, a 10X increase in foreground traffic does not cause any timeouts, thus largely eliminating incast problems.

Discussion Points:

- Is it possible to use DCTP outside data center scenarios? Can we use it to replace traditional TCPs entirely in the transportation layer?
- What would be the implications of using DCTCP on wide area networks?
- Is it possible to integrate DCTCP with SDN controllers to help provide more useful information about intended/ongoing flows? How will it improve the network performance?
- Security issues: A malicious sender can ignore ECN to keep sending packet in a high speed, which may cause other senders received ACK packets with CE bits. The other senders may slow down their speed and the malicious sender can take more and more bandwidth. How to address these issues?

Title: Virtualized Congestion Control

Author: Bryce Cronkite-Ratcliff, Aran Bergman, Shay Vargaftik, Madhusudhan Ravi, Nick McKeown, Ittai Abraham, Isaac Keslassy

In the paper it is indicated that large data center companies are facing challenges to deploy new congestion control algorithms in a datacenter network. In particular, it is not trivial to deploy new algorithms such as DCTCP, TIMELY into a multi-tenant datacenter network where legacy tenants are not always able to upgrade to use new algorithms.

Multi-tenant datacenters are a crucial component of today's computing ecosystem. Multi-tenant datacenters are successful because tenants can seamlessly port their applications and services to the cloud. Virtual Machine (VM) technology plays a key role by allowing end users to run a diverse set of software(applications) in a wide variety of operating systems and configurations. Since VMs are setup and managed by different entities(flexibility), admins can't control VM TCP stacks. This flexibility, however, comes at the cost of dealing with out-dated, inefficient, or misconfigured TCP stacks implemented in the VMs. This leads to two problems: large queue latency and TCP unfairness.

Instead of upgrading tenants, implementing a translation algorithm between legacy algorithms and new algorithms inside a vSwitch of a hypervisor can solve the problem. In the paper, the authors proposed Virtualized Congestion Control(VCC), a new and improved congestion control algorithm into multi-tenant datacenters, that enables the datacenter owner to introduce a new congestion control algorithm in the hypervisors, without having to worry about TCP-friendliness with nonparticipating virtual machines. Internally, the hypervisors translate between the new congestion control algorithm and the old legacy congestion control, allowing legacy applications to enjoy the benefits of the new algorithm, by changing a bit in TCP header, buffering packets, faking ACK packets, or implementing TCP proxy.

It is also stated that enabling ECN (Explicit Congestion Notification) bit is not always possible in legacy VMs. As a result, legacy non-ECN flows perform poorly because non-ECN packets can be dropped at a switch when competing with ECN packets. By translating non-ECN packets to ECN packets can solve the problem. On transmit path, hypervisor marks packets to be ECN enabled. On receive path, upon receiving ECE packet, the hypervisor changes the window size accordingly. Further, vCC also enables application-specific congestion control algorithms to be applied to particular flows.

The authors have implemented proof-of-concept systems for virtualized congestion control in the Linux kernel and in VMware's ESXi hypervisor, achieving improved fairness, performance, and control over guest bandwidth allocations.

Discussion points:

- There are companies that do TCP header modifications. How is this work different from middleboxes?
- What are the limitations of header modifications?
- What happens with real-time application(UDP flow)?