Rethinking Congestion Control Architecture: Performance-oriented Congestion Control

Mo Dong
University of Illinois at Urbana-Champaign

Qingxi Li
University of Illinois at Urbana-Champaign

Doron Zarchy
Hebrew University of Jerusalem

Brighten Godfrey
University of Illinois at Urbana-Champaign

Michael Schapira
Hebrew University of Jerusalem

ABSTRACT

After more than two decades of evolution, TCP and its end host based modifications can still suffer from severely degraded performance under real-world challenging network conditions. The reason, as we observe, is due to TCP family's fundamental architectural deficiency, which hardwires packet-level events to control responses and ignores emprical performance. Jumping out of TCP lineage's architectural deficiency, we propose Performance-oriented Congestion Control (PCC), a new congestion control architecture in which each sender controls its sending strategy based on empirically observed performance metrics. We show through preliminary experimental results that PCC achieves consistently high performance under various challenging network conditions.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Communication*

Keywords

congestion control

1. INTRODUCTION

In the roughly 25 years since its deployment, TCP's congestion control architecture has been notorious for degraded performance in numerous real-world scenarios. TCP performs poorly on lossy links, penalizes high-RTT flows, has difficulty utilizing high bandwidth-delay product (BDP) connections, can collapse in an incast [3] scenario and incurs bufferbloat [4].

Two broad avenues of research have been pursued to improve the performance of TCP. The first is for network devices to give end-hosts explicit feedback. However, because these designs require hardware and configuration changes and (for [6]) packet header changes, in-network solutions have rarely seen widespread deployment.

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2631456 .

In the second avenue of research, end-host-based modifications have addressed problems in specific network conditions, resulting in an innumerably long string of designs tweaking TCP. These include using latency rather than just loss as a signal of congestion [8], sophisticated window expansion algorithms [5] for high bandwidth-delay produce link, coordinated adjustment of the receive window across connections at the receiver to mitigate incast [10], viewing a certain amount of packet loss as unrelated to congestion [7] for lossy wireless networks, expanding the window quicker by setting up a reference RTT [2] to speed up TCP on satellite links and so on.

Although they use distinct congestion control algorithms, all end-host-based TCP variants inherit the very same TCP-based congestion control architecture: hardwiring certain packet-level events (e.g. packet-loss) directly to certain control responses (e.g. halving the window size) like a ***mapping function***. New TCP variants change the mapping function, but the *mapping*-based design itself has not changed.

Unfortunately, two critical problems remain unsolved. First, the very fact that there are such a large number of modifications indicates that each is only a point solution: they yield better performance in certain cases, but break in others, i.e. ***consistently high performance*** is not achieved. Second, and actually worse, we found through real-world experiments that in many cases the performance of these modifications is still far from optimal even in the network conditions towards which they are specially engineered.

We believe these two problems are due to the fundamental deficiency in the TCP-based congestion control architecture. The design rationale behind the mapping function architecture is to make assumptions about the packet-level events. When seeing a packet-level event, TCP *assumes* the network is in a certain state and thus tries to optimize the performance by triggering a predefined control behavior as response to that *assumed* network state. However, in real networks, it is very common that the observed packet-level events were not a result of the assumed network condition. When this assumed link breaks, TCP still mechanically carries out the mismatched control response without even being aware of the severely degraded performance. Take an event-control pair from textbook TCP for example: a packet loss halves the congestion window size. TCP assumes that the packet loss event indicates congestion in the network. If the assumptions about this packet loss event are valid, halving the window size will result in reduced loss rate and may improve performance. However, this assumption can be easily violated: packet loss can be random and unrelated to congestion and thus halving the window size will cause

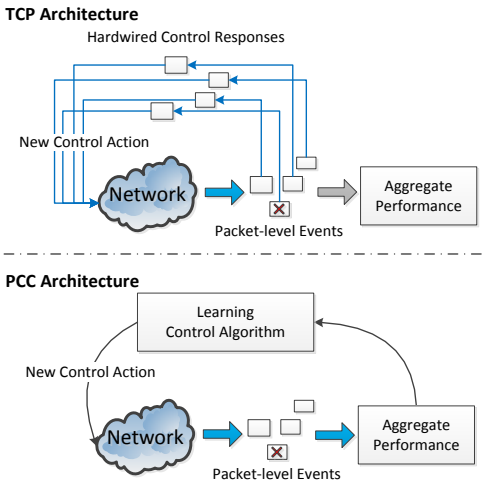

Figure 1: ***The TCP and PCC architectures compared.***

severe performance and fairness degradation. Even the recently proposed TCP exMachina [9] still follows TCP's architecture: in a computationally-intensive offline optimization, it generates a TCP-like event-control mapping function optimized for a set of assumed network conditions. This helps improve performance when the network conforms to the assumptions, but when underlying network condition become different from the input model (e.g. the number of concurrent senders exceeds the expected value), performance still degrades. Therefore, achieving consistently high performance across a range of challenging network conditions ***within the 25-year-old TCP based mapping architecture*** is fundamentally hard.

To solve these problems, we rethink congestion control architecture design and propose Performance-oriented Congestion Control (PCC). PCC *makes control decisions directly based on empirically observed performance outcomes* and we show through preliminary but large scale experiments that it is a promising path towards a congestion control architecture that achieves consistently high performance over a range of challenging network scenarios.

2. ARCHITECTURE OVERVIEW

Unlike the TCP family's congestion control architecture, PCC does not use any predefined packet-level event to trigger certain control behavior. Instead, PCC, as shown in Fig. 1, optimizes applications' performance objectives based on real performance metrics: it picks a sending rate, continuously observes the resulting effect on its performance such as loss rate, throughput, and RTT; and feeds these performance values into a utility function to produce a numerical performance value reflecting the application's objective such as "high throughput and low loss rate". The sender then runs an online learning control algorithm to selfishly adjust its sending rate to maximize this utility over time. Intuitively, this performance oriented congestion control architecture is more robust than traditional TCP's event-control design because it makes fewer assumptions. Rather than using packet-level events to trigger hardwired responses, PCC *observes meaningful performance* and *learns the empirically best strategy* to optimize this performance.

Though PCC has no notion of fairness in its design, it turns out that PCC's selfish optimization need not equate to loss of stability, convergence, or fairness. When choosing a certain kind of utility function expressing the widely-applicable goal of *high throughput and low loss rate*, competing PCC senders will provably converge to a fair share point (unique Nash Equlibrium).

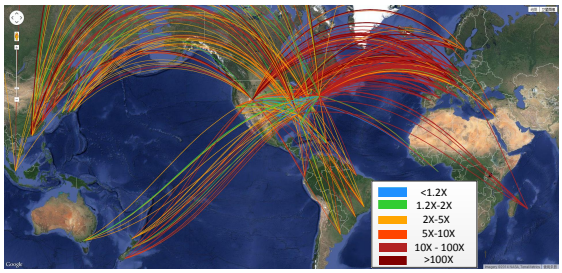

Figure 2: ***Large scale Internet experiment demonstrating PCC's performance improvement over TCP CUBIC***

3. IMPLEMENTATION AND EVALUATION

We built a PCC prototype on top of UDP and evaluated it in experiments spanning PlanetLab, the GENI network, Emulab, and a local testbed. These experiments, ***carried out without any tweaking of control algorithm***, show PCC significantly beating or at least matching specially engineered TCPs on challenging network environments: **a.** provisioned very high capacity backbone networks for scientific data transfer, **b.** incast in data center networks, **c.** lossy satellite links, **d.** links with tiny buffers, **e.** links with bufferbloat, **f.** unfairness of RTT, **g.** the wild and complex conditions of the real Internet, **h.** unusually high loss rate and **i.** latency sensitive applications requiring high throughput and low latency. We also show that PCC's convergence is more stable than TCP even though it has no explicit concern for fairness in its design. Due to the space limit, we only briefly expand two interesting examples here.

High performance satellite communication: satellite links serve critical missions but have packet loss and excessively high latency. State-of-the-art solutions use relay nodes [1] and the purpose-built TCP Hybla [2]. PCC achieves $17\times$ higher throughput than Hybla on links with satellite characteristics, reaching 90% capacity.

5X faster data delivery over Internet: In our global-scale evaluation over 480 sender-receiver pairs across the commercial Internet (Fig. 2), PCC outperforms high BDP optimized TCP CUBIC by $5.52\times$ in the median and achieves $\geq 10\times$ for 41% of source-destination pairs.

Acknowledgement: This work was supported by National Science Foundation grant CNS 1149895.

4. REFERENCES

[1] TCP accelarate on satellite link. `http://goo.gl/E6q6Yf`.
[2] C. Caini and R. Firrincieli. TCP Hybla: a tcp enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking*, 2004.
[3] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph. Understanding tcp incast throughput collapse in datacenter networks. *Proc. ACM SIGCOMM Workshop on Research on Enterprise Networking*, 2009.
[4] J. Gettys and K. Nichols. Bufferbloat: Dark buffers in the internet. December 2011.
[5] S. Ha, I. Rhee, and L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *Proc. ACM SIGOPS*, 2008.
[6] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. *Proc. ACM SIGCOMM*, August 2002.
[7] S. Liu, T. Başar, and R. Srikant. Tcp-illinois: A loss-and delay-based congestion control algorithm for high-speed networks. *Performance Evaluation*, 65(6):417–440, 2008.
[8] D. Wei, C. Jin, S. Low, and S. Hegde. FAST TCP. *IEEE/ACM Trans. Networking*, December 2006.
[9] K. Winstein and H. Balakrishnan. Tcp ex machina: Computer-generated congestion control. *Proc. ACM SIGCOMM*, 2013.
[10] H. Wu, Z. Feng, C. Guo, and Y. Zhang. ICTCP: Incast congestion control for TCP in data center networks. *Proc. CoNEXT*, 2010.