Computer Networks 127 (2017) 352–367

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

More load, more differentiation — Let more flows finish before deadline in data center networks

Han Zhang[a,c], Xingang Shi[b,c,*], Yingya Guo[a,c], Zhiliang Wang[b,c], Xia Yin[a,b]

[a] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[b] Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
[c] Tsinghua National Laboratory for Information Science and Technology (TNLIST), Beijing, China

ARTICLE INFO

Article history:
Received 24 February 2017
Revised 21 August 2017
Accepted 22 August 2017
Available online 30 August 2017

Keywords:
Data center
Congestion
Rate control
Deadline
Flow completion time

ABSTRACT

Data center network has become an important facility for hosting various online services and applications, and thus its performance and underlying technologies are attracting more and more interests. In order to achieve better network performance, recent studies have proposed to tailor data center network traffic management in different aspects, devising various routing and transport schemes. In particular, for applications that must serve users in a timely manner, strict deadlines for their internal traffic flows should be met, and are explicitly taken into consideration in some latest flow rate control or scheduling algorithms in data center networks. In this paper, we advocate that when designing such deadline-aware rate control schemes, a simple principle should be followed: flows with different deadlines should be differentiated in their bandwidth allocation/occupation, and the more traffic load, the more differentiation should be made. We derive sufficient and necessary conditions for a flow rate control scheme to follow this principle, and present a simple congestion control algorithm called Load Proportional Differentiation (LPD) as its application. We have evaluated LPD under different topologies and load scenarios, both by simulation and in real testbed. Compared with D^2TCP, the state-of-art window-based deadline-aware congestion control schema, LPD often reduces the number of flows missing their deadlines by more than 25%. Compared with Karuna, the state-of-art deadline-aware rate control method, LPD only performs about 5% worse on average, but under heavy congestion, LPD performs about 5%–10% better than it. Indeed, the more load more differentiation is a general principle and it can also be used for the optimization of other object. Specifically, we consider minimizing the average flow completion time. Compared with the window-based protocol L^2DCT, LPD can reduce flow completion time by 30% and compared with the state-of-art scheduling method pFabric, it only underperforms by 20% on average.

1. Introduction

Nowadays, more and more services and applications such as web search and social networking are hosted in data centers [1–5]. Since their performance, especially the network performance such as data transfer latency and throughput, directly affect the service quality experienced by end users, network protocols have to be specially designed to fully recognize the unique characteristics of data centers, and fully utilize the resources provided by the underlying infrastructure. In data center networks (DCNs), applications typically require low latency for short flows, and require high burst tolerance and high throughput for large flows [2,6]. To meet these requirements, recent studies have proposed to tailor traffic management in different aspects, and devised various transport schemes for DCNs.

In particular, an important class of applications called Online Data-Intensive (OLDI) [7,8] applications are popularly deployed in data centers. Such applications must serve users in a timely manner since even one more millisecond of latency may considerably affect the revenue [9], and they typically need to query and aggregate results from a large number of nodes. To meet strict response deadlines, applications may send out incomplete responses to users even when some internal query results are not ready yet. This further requires applications' internal traffic flows to meet their respective deadlines as much as possible, since fewer missed deadlines means better response quality, and hence more revenue. Old deadline-agnostic transport schemes like TCP or DCTCP [6] do not work well for these applications, and deadline has been explicitly taken into consideration in modern data center flow rate control [8,10] or flow scheduling [11] algorithms.

* Corresponding author at: Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China.
E-mail address: shixg@cernet.edu.cn (X. Shi).

http://dx.doi.org/10.1016/j.comnet.2017.08.020
1389-1286/© 2017 Elsevier B.V. All rights reserved.

In this paper, we first focus on the design of deadline-aware flow rate control mechanisms for DCNs. We start our work based on D^2TCP [8]. By careful investigations on D^2TCP's performance under different traffic load scenarios, we notice that under high traffic load, it degenerates to DCTCP and becomes nearly deadline-agnostic, which is troublesome since traffic bursts in the aggregation stage of OLDI applications often cause link congestion. With this in mind, we argue that when the network load is heavier, relatively more network bandwidth should be allocated to flows with tighter deadlines, and propose a simple deadline-aware flow rate control principle: *flows with different imminence should be differentiated in their bandwidth allocation/occupation, and the more traffic load, the more differentiation should be made*.

To demonstrate the effectiveness of this principle, we have also designed a simple congestion window based rate control algorithm called Load Proportional Differentiation (LPD) as its application. The basic idea of LPD is that, when using deadline to regulate the congestion window, traffic load is introduced as a multiplicative factor, so that the extent of regulation, as well as the difference of that extent, is proportional to traffic load. Using typical data center topologies and various traffic load scenarios, we evaluate LPD's performance and compare it with those achieved by some latest data center flow rate control algorithms, including DCTCP [6], D^2TCP [8], and L^2DCT [12], Karuna [2]. LPD nearly always outperforms them in enabling more flows to meet their deadlines. In typical scenarios, compared with D^2TCP, it can reduce the number of flows missing their deadlines by more than 25%. Also, compared with the best known deadline-aware method Karuna, LPD performs about 5% worse than it on average. But for some heavy congestion cases, because of the principle of more load, more differentiation, LPD performs about 10% better than Karuna. Besides, we also show our principle is quite general, since its realizations in forms other than LPD can achieve comparable results, and it can be easily adapted for other tasks, such as reducing flow completion time. We modify LPD to let it fit to reduce flow completion time and then compare its performance with the state-of-art window-based protocol L^2DCT and find it can reduce flow completion time by about 30%. Even compared with state-of-art schedule method pFabric [13], LPD performs about 20% worse than it on average. We have modeled LPD and implemented LPD in both ns2 [14,15] and Linux kernel 3.2.61 [16].

The key contributions of this paper are:

- We uncover scenarios where modern deadline-aware flow rate control mechanisms in DCNs become less effective.
- We propose a simple principle for designing deadline-aware flow rate control mechanism in DCNs, and mathematically derive the sufficient and necessary conditions for a scheme to follow this principle.
- We propose the LPD algorithm as an application of our principle and build a fluid model to analyze its performance with different parameters.
- We implement LPD in both ns-2 and linux kernel 3.2.61 and thoroughly evaluate its performance against latest deadline-aware as well as flowsize-aware rate control mechanisms. Simulation code of LPD can be downloaded at [15] and linux kernel source code of LPD can be found at [16].
- We also show the generality of our principle by other forms of rate control algorithms and applications.

The rest of our paper is organized as follows. We first introduce some necessary background and related work in Section 2. Then in Section 3, we describe the more load, more differentiation principle, which is motivated by careful investigations on D^2TCP. The LPD algorithm is presented and analyzed using fluid model in Section 4, and is evaluated in Section 5 both by simulation and in real testbed. Some further discussions are given in Section 6, and finally, Section 7 concludes the paper.

2. Background and related work

The ever-increasingly used data center networks, although often have ultra high bandwidth and low latency links, still face frequent network congestions [17,18]. As a result, throughput collapse or high latency caused by congestion may seriously affect the applications which use DCN as their underlying facility. In particular, a class of Online Data-Intensive (OLDI) applications [7] are very sensitive to the service responsiveness, and often require their internal traffic flows to complete before certain hard deadlines. For these applications, network congestion control or scheduling algorithms have to be specifically tailored, so that flows, especially those short and bursty query flows, can finish before their deadlines.

Among many TCP-like transport schemes in DCNs, DCTCP [6] proposes to mark packets on a switch when its instantaneous queue length exceeds a certain threshold k. The endpoints then estimate the extent of congestion by the marked packets, and throttle flow rates in proportion to that extent. For a congestion level estimation α, the multiplicative decrease algorithm for the congestion window w now becomes $w = w \times (1 - \alpha/2)$ instead of being halved in TCP. By setting an appropriate k, DCTCP elegantly reduces the queue length and its variability on switches, and can reduce the queueing delay and network congestion. Extensive experiments show that DCTCP effectively provides high burst tolerance and low latency for short flows. However, its deadline-agnostic fair sharing of bandwidth among flows with different deadlines make it less effective for the deadline-sensitive OLDI applications, such as web query and advertisement [8].

To this end, deadline-aware protocols have been proposed to explicitly take flow deadline into consideration when allocating bandwidth, which is either computed explicitly by switches [10,11], or adjusted implicitly by varying the congestion window size on end points [8].

D^3 [10] computes the required bandwidth on the switches in a centralized fashion, and grants a sender's request for bandwidth accordingly. Although it is the first known deadline-aware transport protocol and improves upon DCTCP, it is shown to work bad in some race conditions where far-deadline requests arrive slightly ahead of near-deadline requests due to its greedy and first-come-first-service policy [8]. On the other hand, D^3 requires switch hardware change, and cannot coexist with TCP, due to its request-reply mechanism for bandwidth allocation.

D^2TCP improves on DCTCP by adjusting an endpoint's congestion window more intelligently. It defines a deadline imminence factor $d = T/D$, where T is the time needed for a flow to complete its transmitting under a deadline-agnostic manner, while D is the time remaining until its deadline expires. Both the network congestion extent α and the deadline imminence factor d are used in the multiplicative decrease of D^2TCP's congestion window as $w = w \times (1 - \alpha^d/2)$, where α^d is the well-known gamma-correction function [19], and is used as a penality function here. In this way, D^2TCP not only possesses DCTCP's nice property of keeping queue length steady and small, but also favors flows with tighter deadlines. It can co-exist with TCP, and is easy to implement in real data centers.

PDQ [11] uses preemptive flow scheduling at switches to assist bandwidth allocation. Unlike D^3, PDQ works in a distributed fashion, and its preemptive scheduling can deal with the race conditions that threaten D^3. PDQ can emulate different scheduling policies like Earliest Deadline First (EDF) or Shortest Job First (SJF). However, SJF is the just optimal solution to minimize flow completion time over the single link, for complex link it is far