Computer Networks 124 (2017) 46–60

Contents lists available at ScienceDirect

Computer Networks

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

Throughput optimization of TCP incast congestion control in large-scale datacenter networks

Lei Xu[a], Ke Xu[a,*], Yong Jiang[b], Fengyuan Ren[a], Haiyang Wang[c]

[a] Department of Computer Science & Technology, Tsinghua University, Beijing, China
[b] Graduate School at Shenzhen, Tsinghua University, Shenzhen, Guangdong, China
[c] Department of Computer Science at the University of Minnesota Duluth, MN, USA

ARTICLE INFO

Article history:
Received 25 May 2016
Revised 29 May 2017
Accepted 5 June 2017
Available online 6 June 2017

Keywords:
Datacenter networks
Transport protocol
Switch queue
Incast
Congestion control

ABSTRACT

The many-to-one traffic pattern in datacenter networks leads to Transmission Control Protocol (TCP) incast congestion and puts unprecedented pressure to cloud service providers. The abnormal TCP behaviors in incast increase system response time and unavoidably reduce the applicability of cloud-based system deployments. This paper proposes Receiver-oriented Congestion Control (RCC) to address heavy incast in large-scale datacenter networks. RCC is motivated by oscillatory queue size of switch when handling heavy incast and substantial potential of receiver in congestion control when using TCP. RCC makes effective use of centralized scheduler and Explicit Congestion Notification (ECN) at receiver. The RCC prototype is realized in network simulator 3 (ns3) which implements TCP exactly. This paper details the RCC design and evaluates its performance in diverse and heavy workloads. The evaluation results indicate that RCC has an average decreases of 47.5% in the mean queue size and 51.2% in the 99th-percentile latency in the heavy incast over TCP.

© 2017 Published by Elsevier B.V.

1. Introduction

The emergence of cloud computing as an efficient means providing computation can already be felt with the burgeoning of cloud-based applications. Such systems as iCloud, Dropbox, Facebook and Amazon EMR have enjoyed phenomenal growth over the past few years. For instance, Facebook announced that they had Hadoop clusters with 100 petabyte (PB) data across more than 50,000 servers [1]. Other systems as Azure, Dropbox, iCloud are also attracting an increasing number of users and scaling their system deployments on datacenter networks [2,3]. The trend of large-scale datacenter networks is irresistible.

The datacenter networks in this paper are regarding wired networks instead of wireless datacenter networks [4]. The ubiquitous many-to-one traffic pattern in datacenter networks poses challenges for Transmission Control Protocol (TCP). As illustrated in Fig. 1, the many-to-one traffic pattern occurs on the *partition/aggregate* architecture where many work nodes transmit data to the aggregator node. This can easily cause the *TCP incast Congestion* problem in datacenter networks [5–8]. In incast, data flows experience severe packet-drops and long Flow Completion Times (FCTs), bringing users poor Quality of Service (QoS) and enterprise revenue loss [9–11]. It is necessary to design protocols according to network environments [12].

To mitigate this problem, network researchers have proposed effective protocol designs, such as Incast congestion Control for TCP (ICTCP) and Data Center TCP (DCTCP) [13,14]. DCTCP is a pioneer of datacenter transport protocols. Using the Explicit Congestion Notification (ECN) mechanism, DCTCP suppresses queue buildups and packet-drops [13]. H. Wu et al. proposed ICTCP, which controls congestion windows by monitoring receiver throughput. It is the first attempt to control rates of flows at receiver [14]. These protocols are effective for common incast instead of heavy incast which is explained in Section 3.1.

This paper designs a novel transport protocol in congestion control for heavy incast to satisfy requirements from large-scale datacenter networks. Firstly, we investigate heavy incast and its cause. Second, we propose Receiver-oriented Congestion Control (RCC) based on TCP, a mechanism that allows receiver to dominate congestion control. RCC leverages both an open-loop congestion control, i.e., centralized scheduler, and a closed-loop congestion control, i.e., ECN, at receiver to respond to congestion. On the one hand, in RCC, ECN is deployed at receiver to achieve normal

* Corresponding author.
E-mail addresses: l-xu12@mails.tsinghua.edu.cn (L. Xu), xuke@mail.tsinghua.edu.cn (K. Xu), jiangy@sz.tsinghua.edu.cn (Y. Jiang), renfy@mail.tsinghua.edu.cn (F. Ren), haiyang@d.umn.edu (H. Wang).

http://dx.doi.org/10.1016/j.comnet.2017.06.004
1389-1286/© 2017 Published by Elsevier B.V.

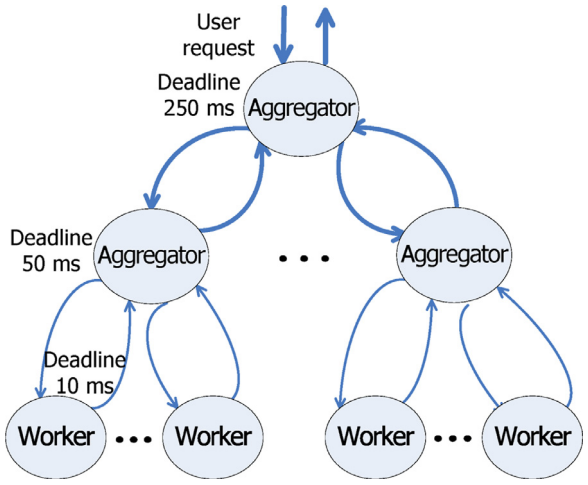

Fig. 1. Partition/aggregate architecture.

congestion control in datacenters [13].[1] On the other hand, the centralized scheduler is used to suppress the burstiness of incast. The previous work of RCC is in [15] and this paper details its design and evaluations.

Integrating the open- and closed-loop congestion controls at receiver is challenging for two reasons. First, the congestion control at receiver has to be compatible with that of TCP. Further, they belong to different types in terms of congestion mechanisms. This paper shows that by arranging the above two congestion controls in a reasonable order, the receiver coordinates different congestion decisions effectively.

Our contributions are as follows:

- This paper teases out the factors impacting transport performance and identifies the root reasons of incast congestion in large-scale datacenter networks.
- We design the receiver congestion control and combine open- and closed-loop congestion controls at receiver. To the best of our knowledge, this is the first attempt to improve transport protocols by combining two congestion controls at receiver.
- We provide a prototype of RCC and evaluate its performance in network simulator 3 (ns3).[2]

The rest of this paper is organized as follows. Section 2 offers related work; Section 3 describes the motivations of RCC; Section 4 details the design of RCC. Section 5 details the analysis of RCC factors. Section 6 further evaluates the performance of RCC in terms of heavy incast, etc. Finally, Section 7 concludes the paper.

2. Related work

Congestion control protocols have been developed for many years with the evolution of the Internet. Network researchers have been improving transport protocols to suit different circumstances all the time. The majority of protocols spring up based on Transmission Control Protocol (TCP) [16]. The state-of-the-art TCP makes effective use of bandwidth by adjusting window sizes according to the Additive-Increase and Multiplicative-Decrease (AIMD) approach. The following presents typical TCP-based protocols in brief.

TCP Reno is proposed to improve TCP throughput when encountering a packet loss [17]. Further, TCP NewReno adds an algorithm for partial Acknowledgment (ACK) during the fast recovery phase for the same aim [18]. TCP Vegas achieves better throughput than TCP Reno. It employs several novel techniques, e.g., Spike Suppression and Congestion Detection by throughput [19]. To meet the challenges from large Bandwidth and Delay Product (BDP) networks, CUBIC is introduced with a window growth function, which is a cubic function of the elapsed time from the last congestion event [20]. Another direction of TCP development is code-based TCP which is beyond the scope of this paper [21,22]. The above protocols mainly focuse on Internet backbone networks rather than specified ones such as datacenter networks.

A chunk of protocols have been proposed for datacenter networks. DCTCP detects congestion degree by ECN packets and makes corresponding congestion decisions [13]. DCTCP has suppressed queue buildups of switches and mitigated congestion. D^3 has introduced deadline into congestion solutions, and it works well with deadline flows [23]. Further, D^2TCP adds flow deadline to DCTCP congestion window control [24]. L^2DCT advances D^2TCP by deploying deadline into not only window decreasing but also increment [25]. PDQ is proposed completely for flows with priorities, achieving an excellent datacenter transmission [26]. Meanwhile, pFabric employs priority switch queues and simplified TCP to achieve outstanding datacenter transport [27]. PASE synthesizes existing transport strategies to suit the need of datacenter networks [28]. Recently, TIMELY is designed based on the Round Trip Time (RTT) mechanism [29].

Receiver-oriented mechanisms have been proposed for several years. There are typical implementations for different contexts [14,30,31]. ICTCP utilizes throughput detection at receiver to improve transport efficiency of datacenter networks in [14]. TCP-Real leverages receiver's ability of decoupling packet loss from window adjustments to avoid unnecessary back-offs [30]. TCP-RTM makes receivers ignore unimportant packet loss for multimedia applications in [31]. Recent work [32] also leverages receiver's ability on detecting actual throughput on the attached link. These procotols reflect the feasibility to deploy congestion control at receiver and receiver's potential of congestion control.

A notable receiver-oriented proposal to cope with incast congestion is PAC [33]. Through a proactive ACK control, PAC is able to cope with heavy incast in datacenter networks. PAC has proved that receiver-oriented method is effective in solving incast problems. The main difference between PAC and RCC is that PAC does not touch TCP-related protocols while RCC is completely a TCP-related protocol.

The following section describes the motivations and objectives of RCC.

3. Motivations

3.1. Queue buildup and packet loss

To analyze heavy incast, this section mimics incast with 50 and 80 senders sending short flows to the same receiver. All the hosts are connected to an 81-port switch whose queue size is 128 KB (i.e., 80 1500-B packets). The starting time of 80 senders complies with exponential distribution with the mean of 15 ms. Hence they are called as randomized senders. Each sender sends 32 KB flow. The link delay is 40 μs and the link bandwidth is 1 Gbps. The real time size of the output queue in the last-hop switch are plotted for three protocols, TCP, DCTCP and ICTCP, in Fig. 2(a) and (b).

In the randomized scenarios in Fig. 2(a) and (b), the queue oscillations are severe. With the increasing number of senders, i.e., 80 senders, oscillation occurs more frequently and fiercely in Fig. 2(b). These two figures show that with increasing senders,

[1] Congestion Experienced (CE) bits in IP headers and ECN-Echo (ECE) bits in TCP headers have been used to convey congestion information in ECN packets. We use the term ECN packets to describe the packets that are marked either with the ECE code point in TCP headers or with the CE code point in IP headers.

[2] The C++ codes of RCC in ns3 are fully accessible at https://github.com/thuxl/rdtcp.

Download English Version:

https://daneshyari.com/en/article/4954633

Download Persian Version:

https://daneshyari.com/article/4954633

Daneshyari.com