Rate Adjustment Mechanism for Controlling Incast Congestion in Data Center Networks

13

Prasanthi Sreekumari

Abstract

Data Center Transmission Control Protocol (DCTCP) gained more popularity in academic as well as industry areas due to its performance in terms of high throughput and low latency and is widely deployed at data centers nowadays. According to recent research about the performance of DCTCP, the authors found that most of the times the sender's congestion window reduces to one segment which results in timeouts. To address this problem, we modified the calculation of sender's congestion window size for improving the throughput of TCP in data center networks. The results of a series of simulations in a typical data center network topology using Qualnet, the most widely used network simulator demonstrates that the proposed solution can significantly reduce the timeouts and noticeably improves the throughput by more than 10% compare to DCTCP under various network conditions.

Keywords

TCP • Data Centers • Timeouts

13.1 Introduction

In recent years, modern data centers host a variety of services and applications such as web search, social networks and scientific computing for various private, non-profit and government systems [1]. Figure 13.1 shows the conventional Data Center Network (DCN) architecture for data centers adapted from Cisco [2]. The typical data center consists of core, aggregation and access layers. Among that, core layer provides the high-speed packet switching for all incoming and outgoing flows of the data center. In addition, it provides connectivity to multiple aggregation modules and serves as the gateway to the campus core. The aggregation layer defines the Layer 2 domain size and has the responsibility of aggregating the thousands of sessions leaving and entering the data center [3].It also provides value added services such as load balancing, firewalling, offloading to the servers across the access layer switches [3]. The aggregation layer connects to the core layer using Layer-3 10 Gigabit Ethernet links. The traffic in the aggregation layer primarily consists of core layer to access layer and access layer to access layer. Furthermore, the access layer operates in Layer 2 or Layer 3 modes and provides the physical level attachment to the server resources.

For communication between nodes, the vast majority data centers use Transmission Control Protocol (TCP) [4]. However, recent research has shown that the TCP does not work well in the unique data center environment [5]. One of the main reasons for the TCP throughput collapse in DCN is TCP Incast congestion. Incast congestion is a catastrophic loss in throughput that occurs when the number of senders communicates with a single receiver by sending data increases beyond the ability of an Ethernet switch to buffer packets. It leads to severe packet loss and consequently frequent TCP timeouts and thereby reduces the performance of TCP.

P. Sreekumari (✉)
Department of Computer Science, Grambling State University, Grambling, LA, USA
e-mail: s.prasanthy@gmail.com

S. Latifi (ed.), *Information Technology – New Generations*, Advances in Intelligent Systems and Computing 558, DOI 10.1007/978-3-319-54978-1_13

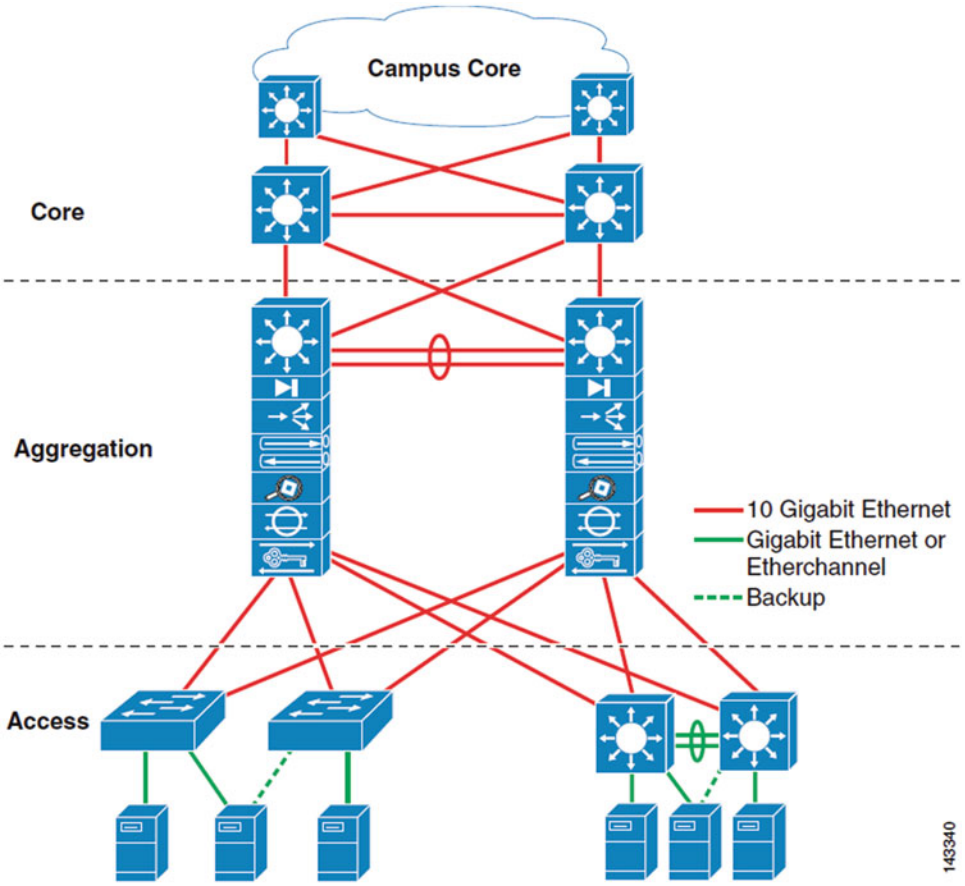

Fig. 13.1 Conventional data center architecture [2]

Recently, few attempts have been made to increase the TCP performance in DCN, but still the problem is not completely solved. Among the existing solutions, DCTCP gained more popularity in academic as well as industry areas due to its better performance in terms of throughput and latency. DCTCP uses very small buffer space compared to other existing solutions. As a result, the sender's congestion window size of DCTCP remains very small which leads to TCP timeouts. In this paper, by considering the limitation of DCTCP, we modified the congestion window adjustment scheme of DCTCP and proposed an efficient rate adjustment method which is capable to control Incast congestion and thereby increase the throughput of TCP. The results of a series of simulations in a typical data center network topology using Qualnet demonstrates that the proposed solution can significantly reduce the timeouts and noticeably improves the performance compare to DCTCP in terms of throughput under various network conditions.

The remainder of the paper is organized as follows. In Sect. 13.2, we present the related work. Section 13.3 describes the details of proposed algorithm. In Sect. 13.4, we describe our experimental methodology and present our results. Finally, Sect. 13.5 concludes our work.

13.2 Related Work

The performance degradation of TCP in data center networks mainly due to Incast congestion. This issue has already attracted the attention of many researchers in our research community. In this section, we survey the solutions proposed recently for controlling TCP Incast congestion in data center networks.

Data Center TCP (DCTCP) [6], a variant of TCP designed to operate with very small queue occupancies, without loss of throughput. DCTCP achieves the goals namely high burst tolerance, low latency and high throughput primarily by reacting to congestion in proportion to the extent of congestion. DCTCP propose a simple marking scheme by modifying the Explicit Congestion Notification [ECN] that helps the sources react to network congestion. TCP Fast [7] is a delay based congestion control algorithm proposed for controlling Incast congestion. This main aim of TCP Fast is to maintain a certain queue length at the switch buffer for each flow and keeps the total queue length below the buffer size. In that way, TCP Fast can avoids the droptails over switch buffer and thereby controls the TCP Incast congestion problem.

Incast Congestion Control for TCP (ICTCP) [8] a systematically designed protocol to perform congestion control at the receiver side by adjusting the receive window proactively before packet loss occurs. To perform congestion control at the receiver side since it knows the throughput of all TCP connections and the available bandwidth, ICTCP uses the available bandwidth on the network interface as a quota to co-ordinate the receive window increase of all incoming connections. In addition, ICTCP use the per-flow state to finely tune the receiver window of each connection on the receiver side.

Another protocol proposed for solving the TCP Incast problem is the Incast Avoidance TCP (IA-TCP). IA-TCP is a rate-based congestion control algorithm that controls the total number of packets injected into the network pipe to meet the bandwidth-delay product [9]. IA-TCP was designed to operate at the receive side like ICTCP, which controls both the window size of workers and the delay of ACKs. To control the sending rate of packets, IA-TCP used the measurement of link capacity i.e., the link rate of the interface connected to the top of the rack switch, and it is obtained from the transport layer. In addition, to control the window size of each worker that employs the standard TCP, the receiver exploits the advertisement field in the ACK header.

TCP-FITDC [10] is a delay-based TCP congestion control algorithm proposed for reacting to congestion states of conventional TCP more accurately. TCP-FITDC utilized the modified Congestion Experienced (CE) codepoint of packets defined in DCTCP to categorize the acknowledgments (Acks) into two classes, marked Acks and Unmarked Acks. By analyzing the differences between these two Acks, TCP-FITDC can estimate the network condition more accurately. The main goal of TCP-FITDC is to achieve low latency, high throughput and fast adjustment for TCP when applied to data center networks.

In [11], the authors introduced a new transport protocol which provides Bandwidth Sharing by Allocating Switch Buffer (SAB) to determine the congestion window of each flow. SAB has two main advantages. First, it converges fast. Second, SAB rarely loses packets. This feature can solve the goodput collapse of TCP Incast as well as the unfairness of TCP Outcast since they are both caused by large numbers of packet losses. SAB modifies the sender, switch and the receiver. In addition, SAB proposes a mechanism for adjusting the size of sender's congestion window value which is less than one. Although these algorithms can mitigate the problem of TCP Incast issue, DCTCP gained more popularity in academic as well as industry areas. However, in DCTCP, the size of congestion window reduced to one frequently which leads to delayed Ack timeouts. For addressing this issue of DCTCP, we propose a new algorithm by adjusting the size of sender's congestion window and thereby control the TCP Incast congestion in data center networks.

13.3 Proposed Algorithm

Sending rate adjustment is an important factor for controlling TCP Incast congestion and thereby improving the performance of TCP in data center networks. In data center networks, the queue length will increase rapidly in a short time due to the concurrent arrival of burst of flows from multiple senders to a single receiver [12]. As a result, switch marks the packets continuously which leads to reduce the sender's congestion window into half of its current size.

In ECN enabled TCP, whenever the sender receives an ECN marked Ack packet, it reduces the size of congestion window into half even if the network is less congested (in the case of single ECN marked Ack packet). This will degrade the performance of TCP. For avoiding the above degradation, DCTCP propose a fine grained reduction function for reducing the size of congestion window based on the value of α. Whenever the sender receives an Ack with congestion experienced code point, the DCTCP sender reduces the size of congestion window using the Eq. (13.1),

$$cwnd \leftarrow cwnd \times (1 - \alpha/2) \tag{13.1}$$

where cwnd is the size of congestion window and the value of α is calculated from the fraction of marked packets (F) and weight factor (g) according to the Eq. (13.2)

$$\alpha = (1 - g)\,\alpha + g \times F \tag{13.2}$$

If the value of α is near zero, it indicates that the network is congested lightly. In this case DCTCP reduces the size of congestion window according to Eq. (13.1). However, if the value of α is equal to one, it indicates that the network is highly congested. As a result, DCTCP reduces the sender's congestion window like normal TCP. The above adjustment of congestion window improves the DCTCP sender to control the buffer occupancy at the switch and thereby increases the throughput of data center networks. Recent study [13] shows that one of the main problems in the congestion window estimation of DCTCP is in the choice of α initialization value. If we set zero to α, the sender may suffers from frequent packet losses and retransmission timeouts. On the other hand, if we set one to α, the sender

can minimize the queuing delay but the amount of packets to be transferred is much smaller. As a result, the throughput of DCTCP will be reduced.

By considering the above limitations of DCTCP, we modified the DCTCP algorithm particularly the adjustment of sender's congestion window and propose a rate adjustment mechanism for controlling the TCP Incast congestion. When the sender receives the Ack with congestion notification, the sender checks the current network condition based on the outstanding packets and adjust the size of congestion window according to the Eq. (13.3),

$$CWstart = \alpha(CWmax\text{-}CWmin) - Cwcurrent \tag{13.3}$$

where α is the number of outstanding packets in the network, *CWcurrent* is the current size of congestion window at the time of receiving the ack packets with congestion notificatio, *CWmax* and *CWmin* are the maximum and minimum size of congestion window adjusted before receiving the congestion notification. When the sender receives the Ack of all outstanding packets, sender adjust the sending rate according to Eq. (13.4),

$$CWend = CWcurrent + \beta(CWmax\text{-}CWmin) \tag{13.4}$$

where the value of β *is 02*. *CWstart and CWend* are the starting and ending points of congestion.

13.4 Working Rationale of Proposed Algorithm

We used the DCTCP slow start, fast retransmission and retransmission timeout algorithms. However, we modified only the sending rate when the sender receives the congestion notification via explicit congestion notification algorithm. When the sender receives the congestion notification through the Ack packets with ECE = 1, the sender starts the congestion point and reduces the size of congestion window according to Eq. (13.3). Once the sender receives all the outstanding packets in the network, the sender changed the window size based on Eq. (13.4). Instead of reducing the congestion window into half of the current size, our proposed rate adjustment can utilize the buffer space efficiently.

13.5 Performance Evaluation

In this section, we present the performance of our proposed protocol through comprehensive simulations using Qualnet simulator [14]. We compare the performance of our algorithm with DCTCP as it is the most popular data center transport protocol. We implemented DCTCP in Qualnet using the source code we got from [15]. Our main goal of this work is to increase the performance of TCP by controlling the Incast congestion in data center networks. For achieving our goal, we evaluate the performance of our algorithm in a typical network topology for Partition/Aggregate cloud applications as shown in [16].

As we did in our previous work [16], to simulate the Incast scenario, we used 10 servers connected to a single client via a switch. The link capacity is set to 1 Gbps and link delay is set to 25 μs, RTT 100 μs and RTO min which is equal to 10 ms. The buffer size is set to 64 KB and 256 KB. We vary the SRU size from 10 KB to 128 KB. The marking threshold value 'K' is set according to [6, 17] for 1 Gbps link capacity. The value of the weighted averaging factor 'g' for DCTCP is set to 0.0625 for buffer size 256 KB and 0.15 for 64 KB. An FTP-generic application is run on each source for sending the packets as quickly as possible. We repeated the experiments for 100 times.

To evaluate the performance of our algorithm with DCTCP, we use three important performance metrics namely, goodput, fast retransmissions and flow completion time. First, we calculated the goodput as the ratio of the total data transferred by all the servers to the client and the time required to complete the data transfer. Second, we evaluated the total number of fast retransmission occurred in DCTCP and proposed algorithm and finally we evaluated the flow completion time of both algorithms. We present the results of our evaluation of proposed algorithm with DCTCP in terms of goodput, flow completion time, and fast retransmissions using a single bottleneck TCP Incast scenario. Figure 13.2 shows the performance of our algorithm compared with DCTCP in terms of goodput. The buffer size we set for this simulation is 64 KB with SRU sizes 64 KB and 128 KB. From the result, we observe that even we used a smaller buffer size, the performance of proposed algorithm achieved higher throughput compared to DCTCP. Figure 13.3 presents the comparison of fast retransmissions of 50 senders. From the result, we observed that DCTCP has higher retransmissions than proposed algorithm. One of the main reason of our algorithm gains less fast retransmission is, the efficient utilization of buffer which leads to reduce the

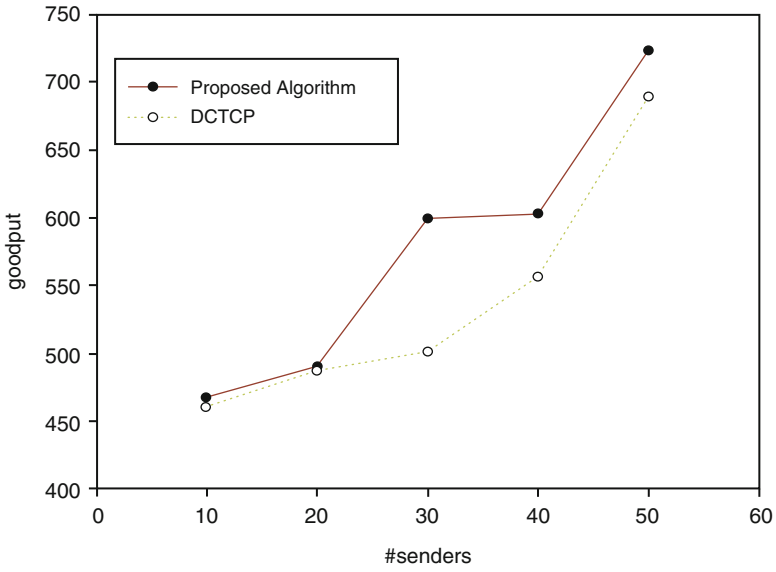

Fig. 13.2 Goodput performance

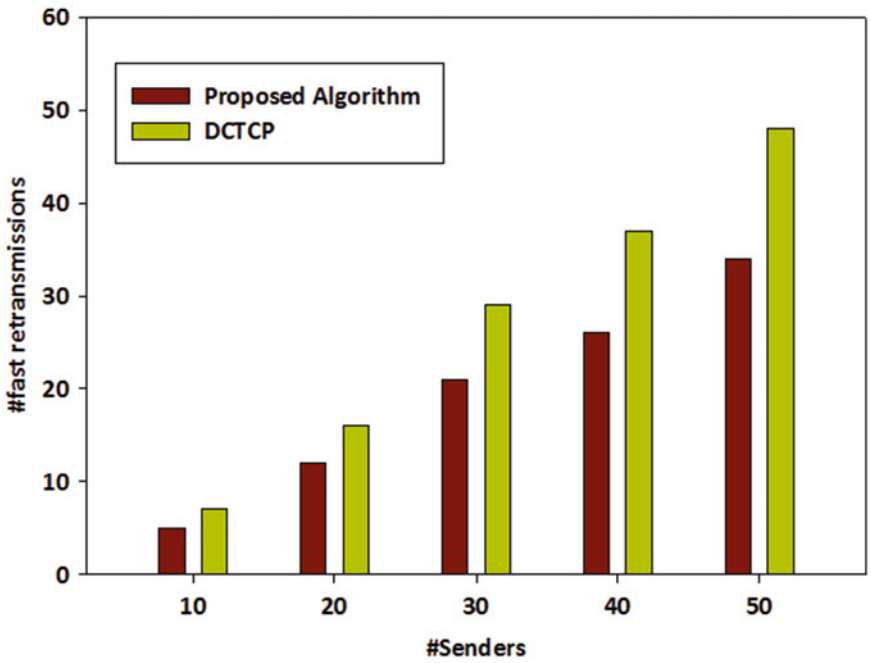

Fig. 13.3 Comparison of fast retransmissions

overflow of queue length and thereby reduce the loss of packets from the network.

Figure 13.4 depicts the query completion time of proposed algorithm compared to DCTCP. As we expected, our algorithm has less query completion time due to the efficient sending rate adjustment mechanism.

13.6 Conclusion

In this paper, we have developed a modified DCTCP protocol for improving the performance of TCP by controlling Incast congestion in data center networks. We proposed a

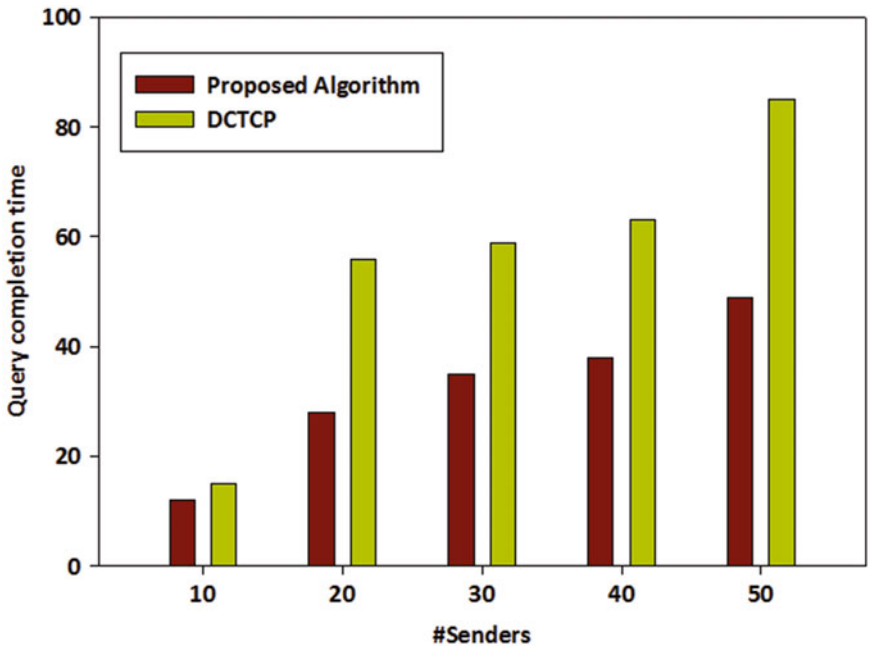

Fig. 13.4 Comparison of query completion time

new sending rate adjustment mechanism for avoiding the frequent retransmission timeouts and thereby utilize the buffer space efficiently. We conducted extensive simulation using Qualnet to validate the performance and effectiveness of our algorithm compared to DCTCP in terms of goodput, flow completion time and fast retransmissions. Our experimental results using the typical TCP Incast scenario shows that the proposed algorithm achieves significant improvement in goodput by reducing the number of fast retransmissions as well as query completion time.

References

1. Zhang, Y., & Ansari, N. (First Quarter 2013). On architecture design, congestion notification, TCP incast and power consumption in data centers. *IEEE Communications Surveys and Tutorials*, *15* (1), 39–64.
2. Cisco Data Center Infrastructure 2.5 Design Guide. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRND_2_5a_book.pdf.
3. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCInfra_1.html.
4. Chen, Y., Griffith, R., Liu, J., Katz, R. H., & Joseph, A. D. (2009). Understanding TCP incast throughput collapse in datacenter networks. In *Proceedings of the 1st ACM workshop on Research on enterprise networking (WREN '09)* (pp. 73–82). New York: ACM.
5. Kant, K. (2009). Data center evolution: A tutorial on state of the art, issues, and challenges. *Computer Networks, 53*(17), 2939–2965. ISSN 1389-1286.
6. Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., Sridharan, M. (2010). Data center TCP (DCTCP). In *Proceedings of the SIGCOMM*, New Delhi, (pp. 63–74).
7. Hwang, J., & Yoo, J. (2014). FaST: Fine-grained and scalable TCP for cloud data center networks. *KSII Transactions on Internet and Information Systems, 8*(3), 762–777. doi:10.3837/tiis.2014.03.003.
8. Wu, H., Feng, Z., Guo, C., Zhang, Y. (2010). ICTCP: Incast Congestion Control for TCP in data center networks. In *Proceedings of ACM CoNEXT*, Philadelphia.
9. Hwang, J., Yoo, J., Choi, N. (2012). IA-TCP: A rate based incast-avoidance algorithm for TCP in data center networks. In *Proceedings of the IEEE ICC*, Ottawa.
10. Zhang, J.,Wen, J.,Wang, J., Zhao, W. (2013). TCP-FITDC: An adaptive approach to TCP incast avoidance for data center applications. *International Conference on Computing, Networking and Communications (ICNC)*, San Diego, (pp. 1048–1052).
11. Zhang, J., Ren, F., Yue, X., Shu, R., & Lin, C. (2014). Sharing bandwidth by allocating switch buffer in data center networks. *IEEE Journal on Selected Areas in Communications, 32*(1), 39–51.
12. Wu, W., & Crawford, M. (2007). Potential performance bottleneck in Linux TCP. *International Journal of Communication Systems, 20*, 1263–1283. doi:10.1002/dac.872.
13. https://eggert.org/students/kato-thesis.pdf.
14. http://web.scalable-networks.com/content/qualnet.
15. http://dev.pyra-handheld.com/index.php.
16. Sreekumari, P., Jung, J., Lee, M. (2015). An early congestion feedback and rate adjustment schemes for many-to-one communication in cloud-based data center networks. *Photonic Network Communications, 31*(1), 23–35.
17. Jiang, C., Li, D., & Xu, M. (2014). LTTP: An LT-code based transport protocol for many-to-one communication in data centers. *IEEE Journal on Selected Areas in Communications, 32* (1), 52–64.