Cluster Comput
DOI 10.1007/s10586-016-0555-z

A simple and efficient approach for reducing TCP timeouts due to lack of duplicate acknowledgments in data center networks

Prasanthi Sreekumari[1] · **Jae-il Jung**[2] · **Meejeong Lee**[3]

Received: 15 January 2015 / Revised: 4 March 2016 / Accepted: 10 March 2016

Abstract The problem of TCP incast in data centers attracts a lot of attention in our research community. TCP incast is a catastrophic throughput collapse that occurs when multiple senders transmitting TCP data simultaneously to a single aggregator. Based on several experiments, researchers found that TCP timeouts are the primary cause of incast problem. Particularly, timeouts due to insufficient duplicate acknowledgments is unavoidable when at least one of the last three segments is lost from the tail of a window. As a result, this type of timeouts should be avoided to improve the goodput of TCP in data center networks. A few attempts have been made to reduce timeouts, but still the problem is not completely solved especially in the case of timeouts due to insufficient duplicate acknowledgments. In this paper, we present an efficient TCP fast retransmission approach, called TCP-EFR, which is capable to reduce TCP timeouts due to lack of duplicate acknowledgments which is caused by the loss of packets from the tail of a window in data center networks. TCP-EFR makes changes in the fast retransmission and recovery algorithm of TCP by using the congestion signal mechanism of DCTCP based on instantaneous queue length. In addition, TCP-EFR controls the sending rate for avoiding the overflow of switch buffer in order to reduce the loss of packets. The results of a series of simulations in single as well as multiple bottleneck topologies using qualnet 4.5 demonstrates that TCP-EFR can significantly reduce the timeouts due to inadequate duplicate acknowledgments and noticeably improves the performance compared to DCTCP, ICTCP and TCP in terms of goodput, accuracy and stability under various network conditions.

Keywords Data center networks · TCP incast · Timeouts · Insufficient duplicate acknowledgments

1 Introduction

A data center network is the communication infrastructure used in a data center [1,2]. Recently, data centers are emerging as critical computing platforms for online services such as Web search, Webmail and advertisement [3,4]. The key goal of data center network is to provide efficient and fault-tolerant routing services to the applications of upper layer and to interconnect the massive number of data center servers [5]. The main features of a data center network are high bandwidth, low propagation delays, and small switch buffers [2]. Data centers form the backbone of the Internet and provide the platform for the deployment of diverse applications such as cloud computing, video streaming etc. Global tech companies like Google, Facebook, Amazon, Microsoft and IBM use data centers for large-scale general computations, e-commerce, Web search and storage and required to provide efficient, scalable and reliable data access and processing [4,6]. With the rise of cloud computing, data center hosting services are in high demand and have become a huge business that plays an important role in the future growth of Information and Communication Technology (ICT) industry [2,3,7].

✉ Jae-il Jung
jijung@hanyang.ac.kr

Prasanthi Sreekumari
s.prasanthy@gmail.com

Meejeong Lee
lmj@ewha.ac.kr

[1] Department of Computer Science, Grambling State University, Grambling, LA, USA

[2] Department of Electronics and Computer Engineering, Hanyang University, Seoul, South Korea

[3] Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea

Published online: 07 April 2016

The data centers are typically organized with many servers for managing the stored data and Ethernet switches for interconnecting these servers to the clients during the data transmissions period [8,9]. The small buffer size of Ethernet switches can easily overflow the buffer resulting in losses due to the simultaneous transmissions of packets from servers. This leads to the poor performance of traditional TCP [10,11]. As a result, the implementation of TCP in data center networks has to face three main challenges such as high latency for short flows [12], TCP Incast [9,10] and TCP Outcast [13]. Among them, a recently found problem TCP incast attracts a lot of attention in our research community [2,9,10,12,14–16] due to the catastrophic goodput degradation by transmitting TCP data simultaneously from multiple servers to a single client. In this type of communication pattern, the client will not send the next data request until all the requested data blocks have been received [11].

Based on several experiments, researchers found that TCP timeouts [14,16,17] are the primary cause of incast problem in data center networks. Specifically, timeouts due to insufficient duplicate acknowledgments (DUPACKs) are unavoidable when at least one of the last three segments is lost [18–21]. This type of frequent timeouts should be avoided to improve the goodput of TCP in data center networks. The reason is, after the timeouts, the servers will again send the requested data simultaneously, which results in the overflow of switch buffer and retransmission for a new round. As a result, it is an important issue of TCP performance in data center networks. A few attempts [19,22–26] has been made to avoid timeouts in data center networks, but still the problem is not completely solved especially in the case of timeouts due to insufficient DUPACKs. In this paper, we present an efficient TCP fast retransmission approach, called TCP-EFR, which is capable to reduce retransmission timeouts due to the lack of DUPACKs which is caused by the loss of packets from the tail of a window in data center networks.

TCP-EFR provides the following contributions for reducing the timeouts due to lack of DUPACKs.

- TCP-EFR introduces two schemes: DUPACK_REACTION and PKT_CONTROL.
  - **DUPACK_REACTION** It helps the sender for taking a decision whether to retransmit the lost packet immediately or not.
  - **PKT_CONTROL** It helps the sender to adjust the sending rate for avoiding the packet losses due to heavy congestion in the network.
- TCP-EFR makes a change in the fast retransmission and recovery algorithm of traditional TCP according to the two different schemes for retransmission and packet controlling.
- TCP-EFR compared its performance improvement with DCTCP, ICTCP and TCP NewReno in terms of goodput, accuracy and stability by a series of simulations in different topologies using qualnet 4.5 network simulator. The results demonstrate that TCP-EFR can reduce the timeouts due to lack of DUPACKs and achieves significant performance improvement than other TCP variants under various conditions of the network.

The remainder of the paper is organized as follows. In Sect. 2, we discuss the problem of timeouts due to insufficient DUPACKs. We present the related work in Sect. 3. Section 4 describes the details of TCP-EFR. In Sect. 5, we describe our experimental methodology and present our results. Finally, Sect. 6 concludes our work.

2 Timeouts due to insufficient DUPACKs

As shown in Fig. 1a in data center networks, multiple servers (S1 to Sn) send the requested data simultaneously to a single receiver (R). The simultaneous transmissions from thousands of servers overload the buffer resulting in packet losses due to the limited buffer space associated with the output port of the switches (S) [11]. As we know that, TCP detects packet losses either by receiving three DUPACKs or the expiration of timeouts [27,28]. In the case of TCP timeouts, the sender retransmits the unacknowledged packet and triggers the slow start algorithm by reducing the size of congestion window to one segment. If the network congestion is too heavy, the frequent timeouts degrades the performance of TCP [29]. For avoiding this situation, fast retransmission algorithm is implemented in order to recover the lost packets. When the sender receives three DUPACKs, it triggers fast retransmission algorithm and retransmit the lost packet by reducing the size of congestion window into half of the current value [30,31]. After retransmitting the lost packet the sender enters into fast recovery algorithm. This data-driven loss detection algorithm helps the TCP to improve the throughput [21].

Unfortunately, in data center networks the data-driven loss detection does not work well if some packets are loss from the tail of a window [17,22]. For example, consider the sender sends 4 packets in a window as shown in Fig. 1b. Among that the segment 2 was lost. Upon receiving the segments 3 and 4, the receiver sends two DUPACKs to the sender.

Without receiving three DUPACKs the sender cannot trigger fast retransmission algorithm. As a result, the sender needs to wait for the expiration of timeouts for retransmitting the lost packet 2. These types of losses are not rare in data center networks due to the simultaneous transmission of data to the clients from large number of servers [12,13]. We conducted an experiment for checking the timeouts due to lack of DUPACKs by packet losses from the tail of a window in

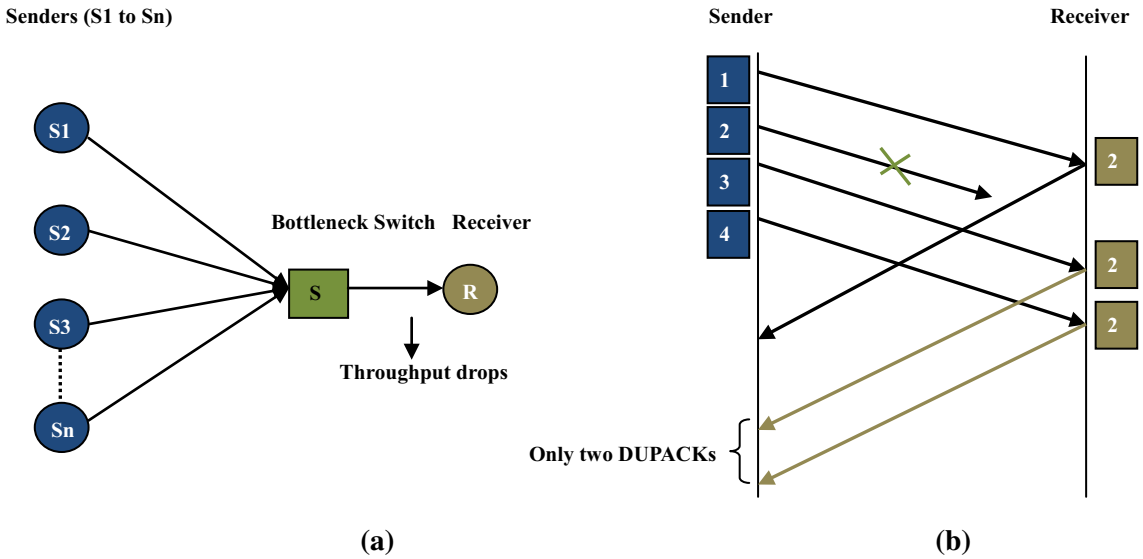

Fig. 1 **a** TCP Incast and **b** timeouts due to lack of DUPACKs

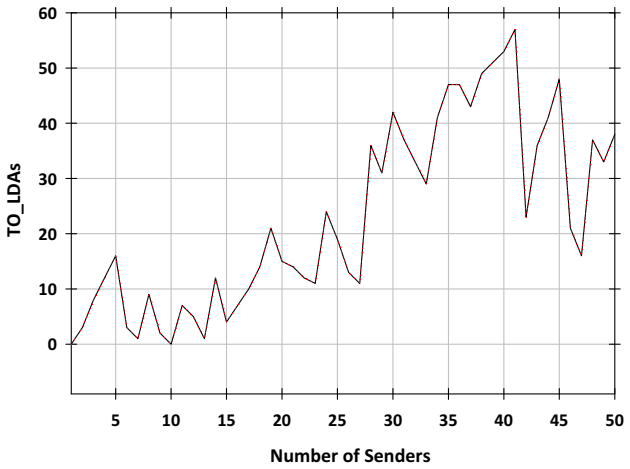

Fig. 2 Number of timeouts due to lack of DUPACKs

a single bottleneck topology as shown in Fig. 1a. The link capacity is set to 1 Gbps and the size of the bottleneck buffer is set to 100 KB. We set the packet size to 1 KB and block size 256 KB. The RTOmin is set to 10 ms. We present the result in Fig. 2. From the experimental result, we observed that when the number of senders increases, TCP timeouts (TO_LDAs) also increases due to insufficient DUPACKs. Hence, it is an important issue for reducing the timeouts due to insufficient DUPACKs as it is one of the primary causes of TCP incast problem in data center networks.

3 Related work

As we mentioned in Sect. 2, the performance degradation of TCP in data center networks are mainly due to TCP incast throughput collapse. This issue has already attracted the attention of many researchers in our research community. In this section, we present the solutions proposed recently for solving the TCP incast problem caused by the frequent timeouts due to many-to-one communication pattern in data center networks.

FGRTO In [32], the authors proposed a practical, effective and safe solution for eliminating TCP incast collapse in data center environments. The authors found that enabling microsecond-granularity retransmission timeouts by eliminating minimum retransmission timeouts can successfully avoid TCP incast collapse in data center applications. This approach can be easily deployed, since it requires only three changes to Linux TCP source code. In addition, based on the experimental results the authors recommended that for achieving full performance delayed acknowledgment (ACK) should be disabled.

DCTCP Data Center TCP (DCTCP) [25], a TCP-like protocol designed to operate with very low buffer occupancies, without the loss of throughput for data center networks [10]. The goal of DCTCP is to achieve high burst tolerance, low latency, and high throughput, primarily by reacting to congestion in proportion to the extent of congestion. DCTCP used Explicit Congestion Notification (ECN) in the network to provide multi-bit feedback from the information present in the single-bit sequence of marks to the end hosts. The simple marking scheme of DCTCP at switches helps the sources react to congestion by reducing the window by a factor that depends on the fraction of marked packets.

ICTCP Incast Congestion Control for TCP (ICTCP) [22] a systematically designed protocol to perform congestion control at the receiver side by adjusting the receiver window. The key idea of ICTCP receive window adjustment is

Table 1 Comparison of recently proposed solutions

Recent solutions	TCP stack modifications	M-PL	M-RPL	M-TPL
FGRTO	Sender and Receiver	No	Yes	No
DCTCP	Sender and Receiver	Yes	Yes	No
ICTCP	Receiver	Yes	Yes	No
IA-TCP	Receiver	Yes	Yes	No
TCP-FITDC	Sender and Receiver	Yes	Yes	No
TDCTCP	Sender and Receiver	Yes	Yes	No
ICaT	Sender and Receiver	No	Yes	Yes

to increase window when the difference ratio of measured and expected throughput is small, while decrease window when the difference ratio is large. To perform congestion control at receiver side, ICTCP uses the available bandwidth on the network interface as a quota to co-ordinate the receive window increase of all incoming connections.

IA-TCP The Incast Avoidance 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 [26]. IA-TCP was designed to operate at the receiver 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 aggregator exploits the advertisement field in the ACK header.

TCP-FITDC TCP-FITDC [19] is a delay-based TCP congestion control algorithm proposed for reacting to congestion states of conventional TCP more accurately and thereby improving the performance of TCP in data center networks. The main goal of TCP-FITDC is to achieve low latency, high throughput and fast adjustment for TCP when applied to data center networks. For achieving the goal, TCP-FITDC utilized the ECN marking scheme, defined in [25], as an indicator of network buffer occupancy and buffer overflow, as well as the queueing delay changes for an estimate of the variations in the available bandwidth.

TDCTCP An improved version of DCTCP algorithm called TDCTCP [33], specifically designed to provide high throughput without significantly increasing the end-to-end delay. TDCTCP changes were introduced in the DCTCP's congestion control algorithm and in dynamic delayed ACK calculation of timeout. Compared to DCTCP, the modified TDCTCP is able to handle the TCP Incast problem very efficiently. The modification of congestion avoidance algorithm helps the sender to adjust its sending rate and can avoid the buffer overflow. The simulation based experimental results shows that TDCTCP achieves 15 % higher throughput and improved fairness when compared to DCTCP. Using the calculation of dynamic delayed ACK timeout, TDCTCP can achieve stable throughput.

ICaT Recently, ICaT [34], presented three solutions for preventing TCP Incast throughput collapse at the beginning, continuation and the termination of a TCP connection. The three solutions are (1) admission control of TCP connections, (2) timestamp-assisted retransmission and (3) reiterated FIN packets. Among the three solutions, the authors reiterated FIN packets as a solution to tail loss. As a result, the sender can retransmit the lost packets by triggering enough DUPACKs.

Table 1 summarizes the comparative study of each solution. We compare each solution using the following four main criteria: modifications to the TCP stack, mechanism for preventing packet losses (M-PL), mechanism for recovering packet losses (M-RPL) and mechanism for avoiding frequent retransmission timeouts due to packet losses from the tail of the window (M-TPL). Although these algorithms can mitigate the problem of TCP incast issue, except ICaT, all other algorithms have no mechanism to reduce frequent retransmission timeouts due to the lack of DUPACKs. As it is not rare in data center networks, the performance of these algorithms degrades due to continuous invoking of slow start algorithms by timeouts. Eventhough ICaT has mechanism for recovering the packets from tail loss without the expiration of timeouts, ICaT does not have mechansim to prevent packet loss from retransmissions and frequent timeouts due to highly bursty traffic of multiple TCP connections in data center networks. Instead, ICaT helps the TCP sender to recover the packets after loss. In our work, we not only propose a solution to recover the lost packets due to tail loss but also provide a mechanism for avoiding the packet losses before the overflow of Ethernet switch buffer.

4 TCP-EFR

The primary objective of TCP-EFR is to increase the performance of TCP by reducing the frequent retransmission timeouts due to insufficient DUPACKs. One of the main reasons for the lack of DUPACKs is the loss of packets from

the tail of a window. To achieve our goal, we propose two schemes in TCP-EFR, namely, DUPACK_REACTION and PKT_CONTROL.

- **DUPACK_REACTION** In this scheme, the sender is capable to retransmit the lost packets even if the sender does not receive enough DUPACKs.
- **PKT_CONTROL** This scheme helps the senders to control the packet sending rate by reducing the size of congestion window.

In addition, we make changes to the fast retransmission and recovery algorithms of TCP NewReno as it is the most deployed protocol in data center networks. We explain the two schemes in below subsections.

4.1 DUPACK_REACTION

In TCP, the sender uses the packet loss signal for determining the congestion of the network. As we mentioned in Sect. 2, TCP uses two methods to convey packet loss due to congestion in the network. One is retransmission timeout and the other is fast retransmission by three DUPACKs. Compare to former method, the indication of packet losses due to network congestion via DUPACKs is faster for recovering the lost packets without degrading the performance. However, if the sender does not receive the packet loss signal by three DUPACKs, it waits for a longer period of time for the expiration of retransmission timer and degrades the performance of TCP. In data center networks, this type of timeouts is unavoidable due to the drop of packets from the tail of a window [18–21].

For reducing such type of timeouts, we propose a DUPACK_REACTION scheme in TCP-EFR for reacting to the packet loss via DUPACKs in the network.

In TCP-EFR, whenever the sender receives a DUPACK, it calculates the outstanding packets by using the equation in [35]. If the number of outstanding packets is less than or equal to 3, then we retransmit the lost packet immediately without waiting for two more DUPACKs. However, retransmitting the packets before receiving three DUPACKs causes false retransmissions due to network failure or packet reordering as it is not rare in data center networks [36–38]. For avoiding such situations, we use the congestion signal (CS) for determining whether the DUPACK is due to congestion in the network or not.

Many authors have pointed out that the CS through packet marking mechanism provides sufficient information about the congestion in the network [19,25,33,39]. As a result, for avoiding false retransmissions by reducing the DUPACK threshold from three to one based on the outstanding packets, we utilize the DCTCP congestion notification packet

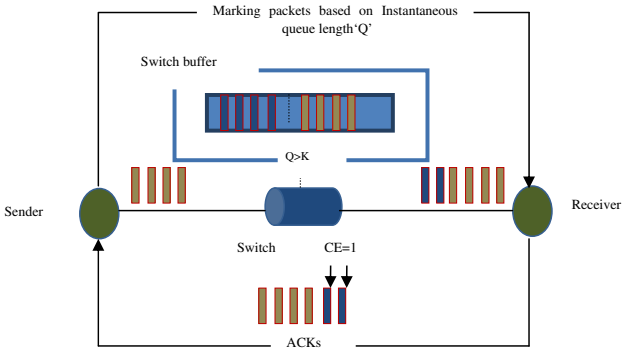

Fig. 3 TCP-EFR congestion signal mechanism

marking mechanism. DCTCP, a noted solution proposed for mitigating the TCP incast problem in data center networks, modified the packet marking mechanism of ECN and employs a very simple active queue management scheme for informing the sender about the congestion in the network. As shown in Fig. 3, if the queue occupancy is greater than a threshold value (K), TCP-EFR marks all the arriving packets based on the instantaneous queue length (Q) with the congestion experienced (CE) code point. For sending the timely congestion notification to the sender, TCP-EFR sends ACK for every packet like DCTCP, setting the ECN-Echo flag if and only if the packet has a marked CE code point. By using the packet marking mechanism of DCTCP, the DUPACK_REACTION scheme of TCP-EFR modifies the fast retransmission algorithm for reducing the timeouts due to insufficient DUPACKs.

The DUPACK_REACTION of TCP-EFR is as follows: We classify the DUPACKs into two types: DUPACK with CS and DUPACK without CS.

- **DUPACK with congestion signal** Whenever the sender receives a DUPACK with CS, it indicates that the packet loss happens due to heavy network congestion. However, the sender does not know whether the packet loss happens from the tail of a window or not. If it is from the tail of a window, the sender does not receive three DUPACKs and waits for the retransmission timeouts. For avoiding the retransmission timeouts, the sender checks the outstanding packets and if the outstanding packets is less than or equal to three the sender immediately retransmit the lost packet without waiting for two more DUPACKs. Otherwise the sender waits for three DUPACKs for recovering the lost packets like TCP NewReno. In this way, TCP-EFR avoids retransmission timeouts due to the lack of DUPACKs in heavy network congestion.
- **DUPACK without congestion signal** When the sender receives a DUPACK without CS, the sender need not

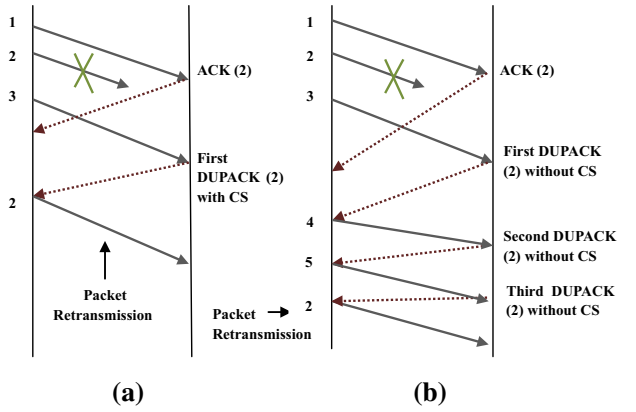

Fig. 4 Example of DUPACK_REACTION scheme of TCP-EFR. **a** DUPACK with CS, **b** DUPACK without CS

retransmit the packet immediately. Instead, it sends a new packet [35]. We assume that the newly sent packet may not lose in the network. When the sender receives a second DUPACK, again the sender sends another new packet. Upon the arrival of newly sent packets, the receiver can send third DUPACK if the packet is really lost in the network. Upon receiving the third DUPACK, the sender retransmits the lost packet. A simple example of DUPACK_REACTION is shown in Fig. 4a and b. Consider the sender sends 3 packets. Among that, the packet 2 was lost. As a result, the sender gets a DUPACK of the lost packet. As shown in Fig. 4a assume that the sender received the first DUPACK with CS. As a result the sender immediately retransmits the lost packet 2 since the outstanding packet is less than 3. On the other hand, if the sender receives a DUPACK without CS as shown in Fig. 4b, then the sender transmits a new packet 4 instead of retransmission.

Upon receiving the new packet 4, the receiver again sends a DUPACK of the lost packet 2. The sender sends the next packet 5. When the sender receives a third DUPACK of the lost packet 2, it retransmits the packet 2 immediately and follows the fast recovery procedure. With the help of DUPACK_REACTION scheme the sender can reduce the timeouts due to insufficient DUPACKS and also the sender can reduce the false retransmissions.

4.2 PKT_CONTROL

Here we explain about the PKT_CONTROL scheme of TCP-EFR. Controlling the packet sending rate is important for maintaining the queue length of the buffer. 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 [11]. As a result, switch marks lot of packets and the senders reduce their congestion window size frequently and this will leads to degrade the performance. For avoiding the rapid increase of queue length, whenever the sender received a CS via ACKs or DUPACKs, it reduces the congestion window based on four cases:

- When the first DUPACK is received with CS and the sender is not already in the fast recovery procedure, we set the size of congestion window like TCP NewReno.
- When the first DUPACK is received without CS and the sender is not already in the fast recovery procedure, we use the current value of the congestion window.
- When the sender receives a normal ACK with CS, the size of congestion window (CWND) reduces based on the fraction of marked packets like DCTCP. In DCTCP, whenever the sender receives an ACK with ECE is set to 1, the sender reduces the congestion window using the Eq. 1,

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

where α is calculated from the fraction of marked packets (F) and weight factor (g) according to the Eq. 2

$$\alpha = (1 - g)\,\alpha + g \times F \tag{2}$$

If the value of α is near zero, it indicates that the network is congested lightly. On the other hand, if the value of α is equal to one, it indicates that the network is highly congested. In the former case, DCTCP congestion window reduces slightly according to the Eq. 2.
- Whenever the sender experiences a timeout, the CWND reduction is same as TCP NewReno.

5 Performance evaluation

We evaluate the performance of TCP-EFR in terms of goodput, accuracy and stability through Qualnet simulations [40] using single as well as multiple bottleneck links. We compare the performance of TCP-EFR with DCTCP, ICTCP and TCP (NewReno) with and without background traffic. Among the different TCP variants proposed for data center networks, DCTCP and ICTCP are the noted solutions designed for improving the performance of TCP [2,6,8,9,13]. In addition, we used TCP NewReno as it is the most widely deployed transport protocol in data center networks. More in detail, in Sect. 5.1, we explain about the performance metrics used for our simulations, while in Sect. 5.2 we describe about the single bottleneck topology, the parameters and results obtained from our evaluation. Further the evaluation set up and the

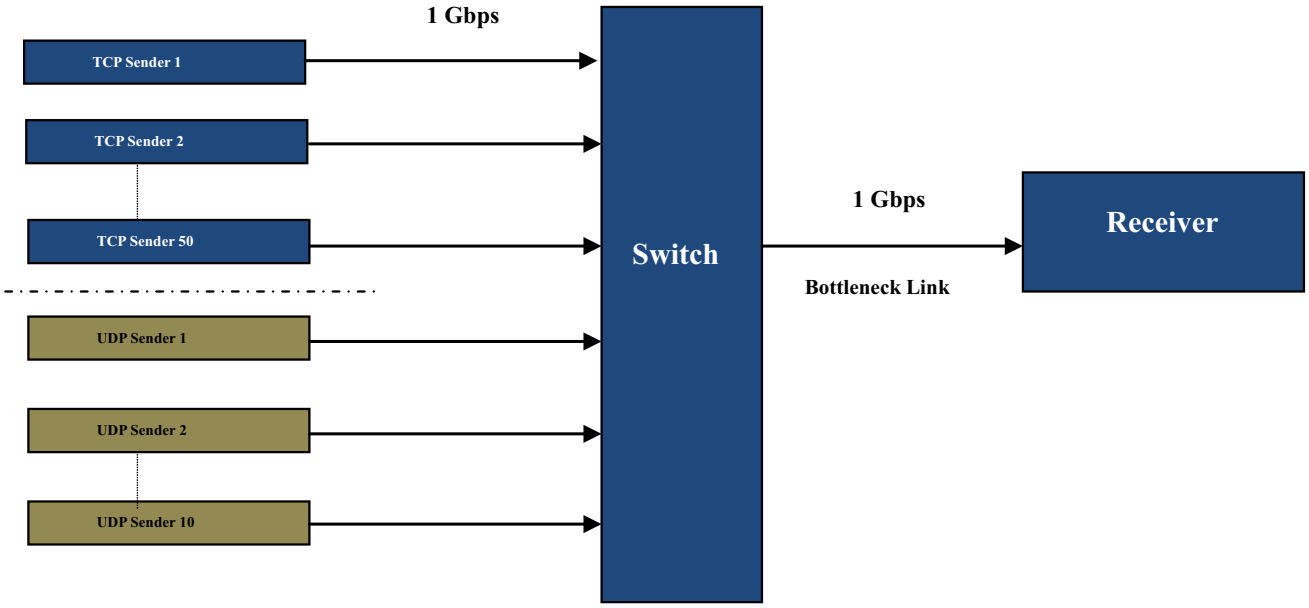

Fig. 5 Single bottleneck topology

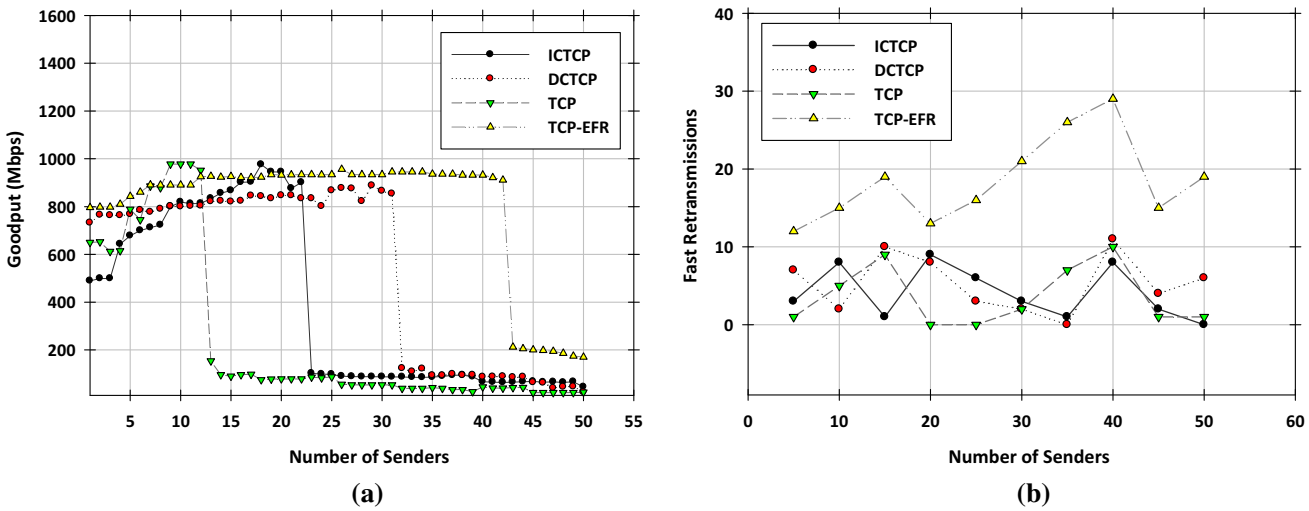

Fig. 6 Comparison of goodput in **a** and comparison of fast retransmissions in **b**

results using multiple bottlenecks topology is presented in Sect. 5.3.

5.1 Performance metrics

To evaluate the ability of TCP-EFR for improving the performance of TCP in data center networks by reducing the frequent retransmission timeouts from lack of DUPACKs, we use the following main performance metrics:

5.1.1 Goodput

It is calculated as the ratio of the total data transferred by all the servers to the client and the time required to complete the transmission of data [24].

5.1.2 Accuracy

Accuracy is one of the most important metric for improving the goodput performance. If accuracy increases, goodput also increases. For evaluating the performance of TCP TCP-EFR, we check the accuracy of detecting timeouts (A_{TO_LDA}) which happened due to the lack of DUPACKs as shown below.

$$A_{TO\_LDA} = (N_{PR\_TO}/Total_{TO\_LDA})^{*}100 \quad (3)$$

Where N_{PR_TO} is the number of packets recovered from timeouts and Total $_{TO\_LDA}$ is the total number of timeouts due to insufficient DUPACKs occurred in our simulation.

5.1.3 Stability

We use the standard deviation of the accuracy of packet loss recovery using multiple bottleneck topology. To calculate the stability, we use the equation used in [41].

$$S = \sqrt{1/n \sum_{i=1}^{n} (Ai - A)^2} \quad (4)$$

where Ai is the accuracy measured in a simulation scenario i, A is the average of accuracies measured in a set of simulation scenarios and n is the number of simulation scenarios.

5.2 Single bottleneck topology

We first evaluate the performance of TCP-EFR with TCP, DCTCP and ICTCP in a single bottleneck topology. In this section, we explain the topology and evaluation set up used for our simulations. Figure 5 shows our single bottleneck topology which consists of 60 senders, one switch and one receiver. Among that, 50 senders use TCP flows and 10 senders use UDP flows. The link capacity is set to 1 Gbps, the switch buffer size is set to 100 KB per port and the sender SRU size is set to 64 KB. Otherwise we specified. The RTOmin is set 10 ms, link delay is set to 25 µs making the base RTT 100 µs and the size of the UDP background flow is set to 10 Mbps. For the key parameters of DCTCP and TCP-EFR, we set the weighted averaging factor to 0.0625 and the value of packet marking threshold 'K' to 20 packets [42]. In our simulations, the application sends the data as fast as possible to the transport layer and the transmission is completed when the receiver receives the requested data successfully. In Sects. 5.2.1 and 5.2.2, we present the results obtained from our simulations using our performance metrics without background traffic and in the presence of background traffic.

5.2.1 Performance without background traffic

First, we present the performance of TCP-EFR in the absence of background traffic. For this experiment, we use 50 TCP senders for sending data to the receiver. We compared the goodput of ICTCP, DCTCP, TCP and TCP-EFR as shown in Fig. 6a. From the results, we can see that the goodput of TCP-EFR greatly improves compared to other TCP variants especially when the number of sender increases. For instance, when there are 32 senders, the goodput of TCP and ICTCP variants are less than 100 Mbps while that of DCTCP is 123 Mbps.

However, the performance of TCP-EFR reaches to 945 Mbps. The reason for this achievement is the loss recovery mechanism of TCP-EFR compared to TCP, ICTCP and DCTCP. In addition, we observed that as the network congestion becomes heavier the performance of TCP-EFR reduces to less than 250 Mbps after 40 servers due to more timeouts. But, it still achieves better performance compared to ICTCP, DCTCP and TCP. Figure 6b presents the performance of TCP-EFR in terms of fast retransmissions. Compared to other variants, TCP-EFR has the best performance since it has an efficient mechanism to recover the lost packets by fast retransmissions and thereby reduce the frequent retransmission timeouts. When the number of senders increases to 40, TCP-EFR has more than 20 fast retransmissions compared to other variants. Next, we evaluate the query completion time using the same parameters. In Fig. 7, we observe that the query completion time of TCP and other variants increases with increase in the number of senders. When the number of sender increases, the many-to-one traffic pattern of data center networks often causes heavy network congestion which results in high packet loss ratio. In this figure, we find that TCP-EFR takes only less time compared to DCTCP, ICTCP and TCP. This is because TCP-EFR is able to maintain the queue length by using the PKT_CONTROL scheme and thus reduce the number of packet losses. In the case of 50 senders, the query completion time of TCP-EFR is less than 40 ms while that of ICTCP and TCP are above 50 ms.

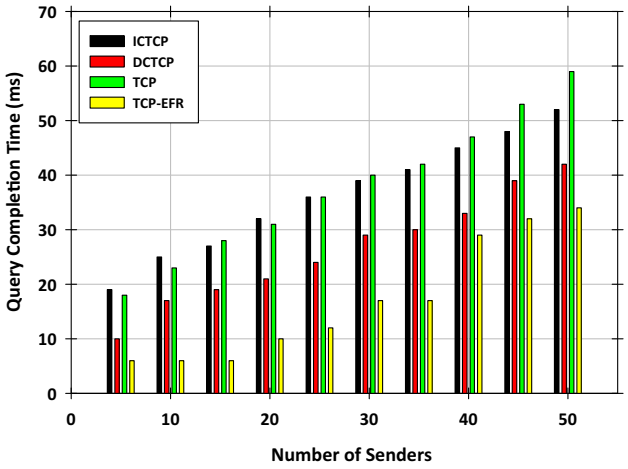

Fig. 7 Comparison of query completion time

5.2.2 Performance with background traffic

We conduct simulations to evaluate the performance of TCP-EFR in the presence of background traffic. For this evaluation, we use 50 TCP senders and 10 UDP senders in our single bottleneck topology. Rest of the parameters is same as we used in the above subsection. Figure 8 shows the number of timeouts due to the lack of DUPACKs in the presence of background traffic. We observe that the timeouts of TCP-EFR is much less than that of DCTCP, ICTCP and TCP. Upto 20 senders TCP-EFR has almost zero timeouts due to its effi-

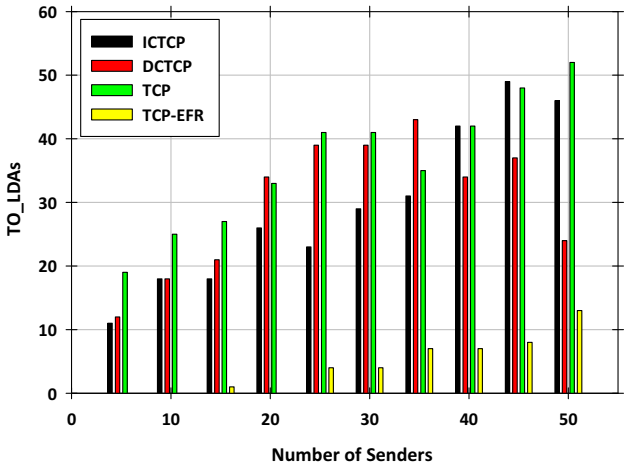

Fig. 8 Comparison of timeouts due to insufficient DUPACKs

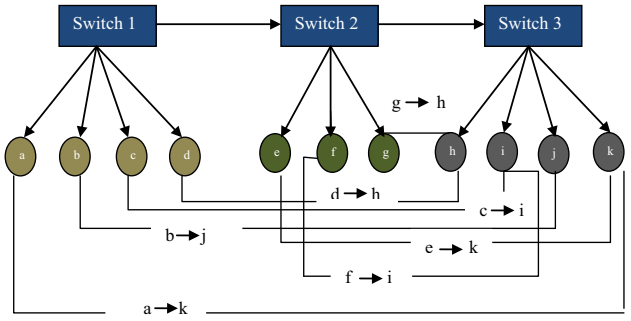

Fig. 10 Multiple bottleneck topology

cient retransmission mechanism. However, DCTCP, ICTCP and TCP have no mechanism to recover the lost packet without timeouts if the losses happen from tail of a window and they suffer from frequent timeouts. Although TCP-EFR can recover the packets from new retransmission mechanism, TCP-EFR also suffers some timeouts when the number of senders increases. However, compared to other TCP variants TCP-EFR has less number of timeouts.

Figure 9a shows the performance of TCP-EFR in terms of accuracy with varying number of senders with background traffic (WBT) and without background traffic (WoBT). We should note that, upto 20 senders TCP-EFR has 100 % of accuracy for recovering the packet losses from tail of a window. However, the accuracy is decreasing when the number of sender increases. The main reason is the loss of multiple packets from a window. When the sender reaches 40, the accuracy is close to 80 % in terms of without background traffic and less than 45 % in terms of the presence of background traffic. On the other hand, more than 40 % accuracy achieved in the absence of background traffic at 50 senders while less than 30 % accuracy achieved in the presence of background traffic at the same number of senders.

In Fig. 9b, we present the accuracy of TCP-EFR with background traffic in terms of number of different flows which varies from 20 to 100. When the number of flows increases the accuracy is decreasing. However, TCP-EFR achieves more than 40 % of accuracy even when the number of flows reaches 100. In the next section, we present the performance of TCP-EFR and all other TCP variants using multiple bottleneck topology.

5.3 Multiple bottleneck topology

Our previous simulation results are based on single bottleneck topology in which all the senders and the receiver are communicated by a single switch. In this section, we compare the performance of TCP-EFR in multiple bottleneck topology which consists of three switches, seven senders and four

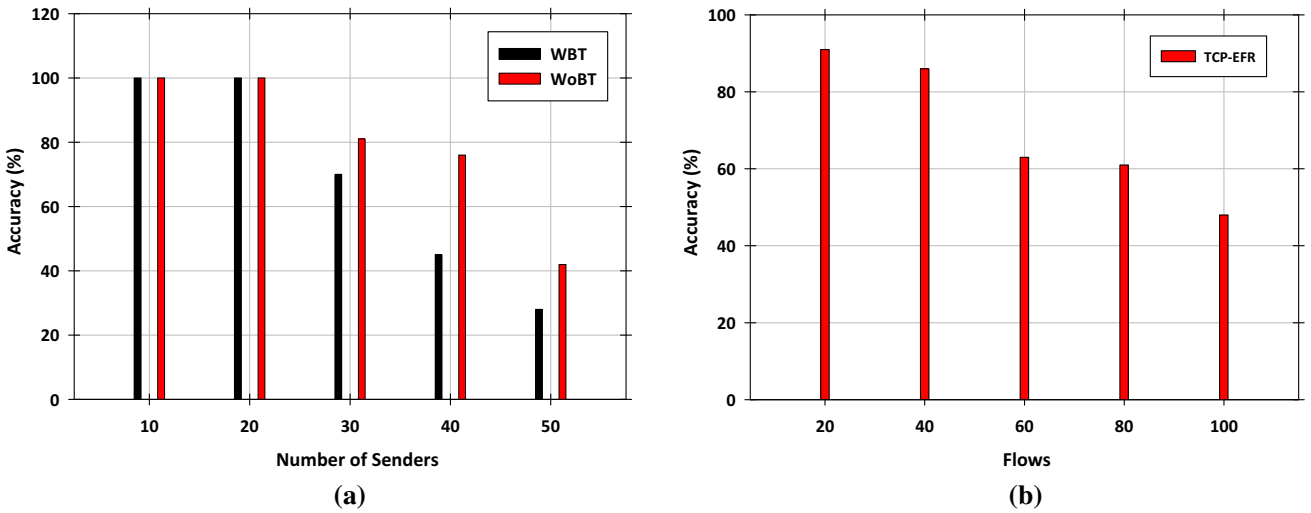

Fig. 9 Comparison of accuracy of TCP-EFR . **a** Number of senders and **b** number of flows

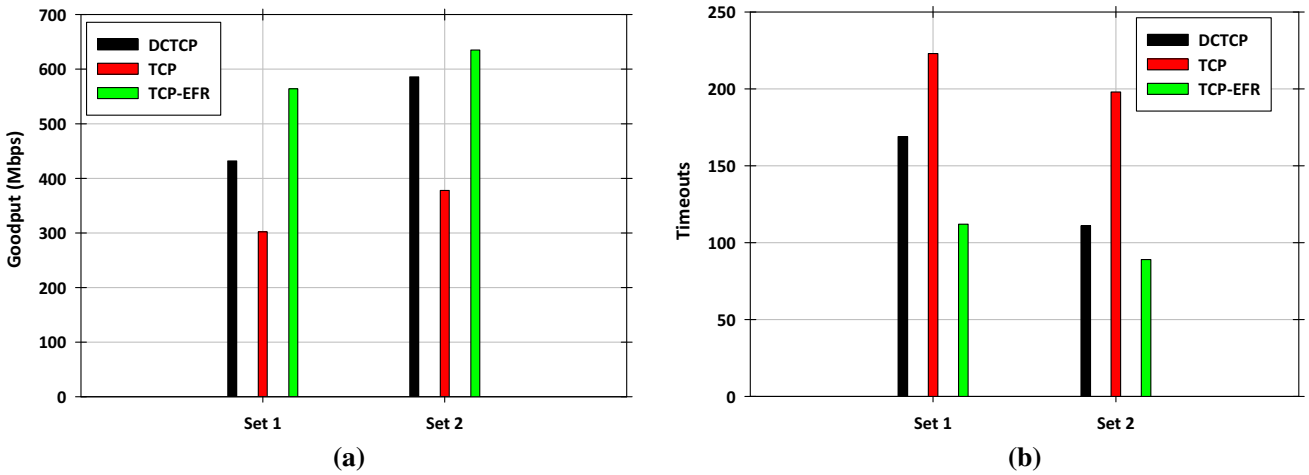

Fig. 11 Comparison of goodput in **a** and **b** presents the results of timeouts

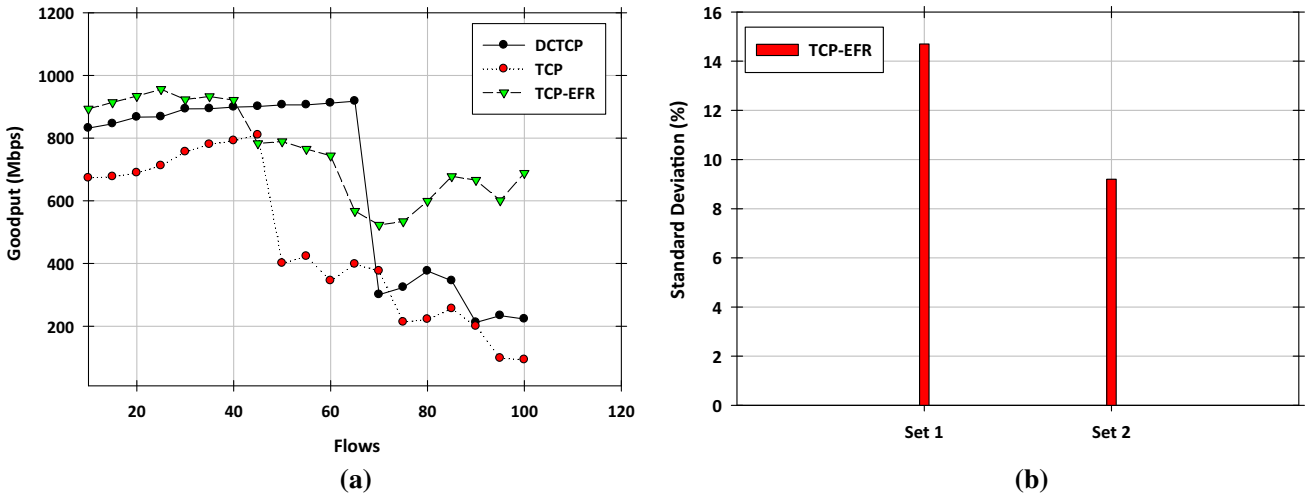

Fig. 12 Comparison of goodput in **a** and **b** shows the stability of TCP-EFR

receivers as shown in Fig. 10. Among that, switch 1 consists of four senders (a, b, c and d), switch 2 consists of 3 senders (e, f and g) and switch 3 consists of 4 receivers (h, i, j and k). We compare the results into two sets: set 1 consists of flows from the senders a→k, b→j, c→i and d→h whereas set 2 consists of flows from the senders e→k, f→i, and g→h.

The capacity of each link is 1 Gbps, link delay we set to 25 μs, and the size of switch buffers are 128 KB. The packet size we set to 1 KB. The senders of set 1 communicate with the receivers through the switches 1, 2 and 3 whereas the senders from set 2 communicate with the receivers through the switches 2 and 3. Using multiple bottleneck topology, we compared the performance of TCP-EFR with DCTCP and TCP.

Figure 11a presents the comparison of goodput measured from sets 1 and 2. Compared to set 1, the senders in the set 2 achieve better goodput. The main reason is differences in the number of hops between the senders and the receivers. If hops increase, RTT also increases. However, TCP-EFR outperforms DCTCP and TCP in both sets. In set 1, TCP-EFR achieves more than 600 Mbps while the goodput of DCTCP and TCP are less than 450 Mbps.

On the other hand, in set 2 TCP-EFR maintains the goodput of 635 Mbps while that of DCTCP and TCP have 586 Mbps, 378 Mbps, respectively. Figure 11b shows the number of timeouts happens in sets 1 and 2. We can clearly see that the timeouts of TCP-EFR is less than 150 compared to other variants in both sets. It means that TCP-EFR can continuously send more packets without suffering the frequent invoke of slow start algorithm due to timeouts. This reflects in the goodput of TCP-EFR. In the case of other variants, they suffer more than 200 timeouts in set 1 and more than 100 timeouts in set 2 which results in the degradation of goodput. Figure 12a shows the comparison of goodput in terms of different number of flows from the senders in set 2. From the results, we can see that after 80 flows the goodput of all vari-

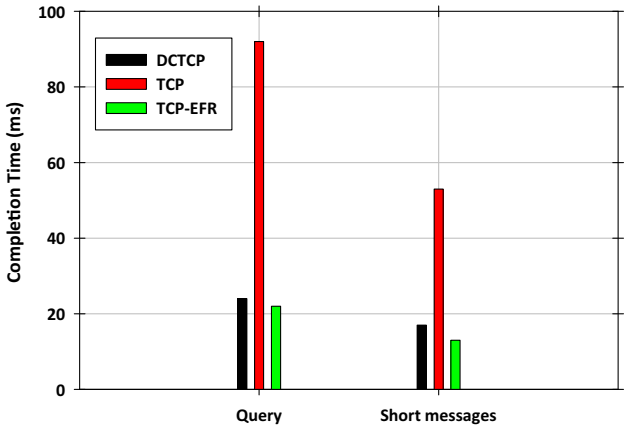

Fig. 13 Comparison of completion times

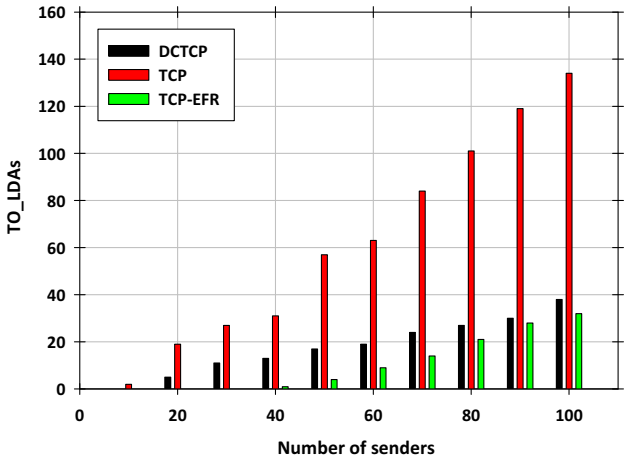

Fig. 14 Comparison of timeouts

ants except TCP-EFR falls below 400 Mbps while TCP-EFR outperforms with goodput more than 600 Mbps.

Figure 12b presents the stability of TCP-EFR in sets 1 and 2 of the multiple bottleneck topology. From the results, we observed that set 2 is more stable than set 1. This is because; in set 1 TCP-EFR suffers more packet losses. As a result, the stability of accuracy fluctuates in this network condition and drops the stability.

We conducted more simulations for evaluating the timeouts and the completion times of query and short messages by generating practical traffic similar to the traffic characteristics described in [25]. For this experiment, we used a network topology that consists of one aggregation switch and 10 top-of-rack switches, each rack has 10 servers. We set the link rate of each rack switches to 1 and 10 Gbps for the aggregation switch. The delay of the links was set to 25 μs and we deployed a large buffer for the aggregation switch by assuming that the rack switches are shallow buffered commodity switches. The threshold value 'K' we set for TCP-EFR and DCTCP are 20 for 1 Gbps and 65 for 10 Gbps and the RTOmin was set to 10 ms. Figure 13 shows the comparison of average completion times of query and short messages. Compared to TCP, DCTCP and TCP-EFR achieves good performance. However, the completion times of TCP-EFR shows very low than the completion times of DCTCP due to its ability for avoiding the timeouts due to tail packet losses and the mechanism for controlling the sending packets. Figure 14 presents the average number of total timeouts occurred due to tail packet losses during simulations. When the number of senders increases, TCP-EFR shows timeouts. However, the number of timeouts is lesser compared to DCTCP and TCP.

6 Conclusion

TCP timeouts are the root cause of incast problem in data center networks. In data center networks more than 30 % of packet losses are caused from the tail of a window. This results in lack of DUPACKs for triggering fast retransmissions and cause frequent timeouts. For reducing such types of timeouts, in this paper, we presented a simple and efficient solution called TCP-EFR which consists of two schemes for reducing the timeouts due to lack of DUPACKs and controlling the sending rate for maintaining the queue length of the buffer in order to avoid more timeouts. The extensive simulation using Qualnet 4.5 show that TCP-EFR can effectively improve the goodput by reducing the timeouts compared to the existing solutions, namely, DCTCP, ICTCP and TCP. In addition, the accuracy of TCP-EFR is better for recovering the lost packet via DUPACKs rather than waiting for timeouts. Furthermore, the performance of TCP-EFR in terms of stability also shows satisfactorily improvement in multiple bottleneck topology.

Acknowledgments This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future Planning (NRF-2015R1A2A1A15056298) and also supported by the ICT R&D program of MSIP/IITP. [B0126-15-1051, A Study on Hyper Connected Self-Organizing Network Infrastructure Technologies for IoT Service].

References

1. Kant, K.: Data center evolution: a tutorial on state of the art, issues, and challenges. Comput. Netw. **53**(17), 2939–2965 (2009). doi:10.1016/j.comnet.2009.10.004. ISSN 1389–1286
2. Ren, Y., Zhao, Y., Liu, P., Dou, K., Li, J.: A survey on TCP incast in data center networks. Int. J. Commun. Syst. **27**, 1160–1172 (2012)
3. Bari, M.F., Boutaba, R., Esteves, R., Granville, L.Z., Podlesny, M., Rabbani, M.G., Qi, Z., Zhani, M.F.: Data center network virtualization: a survey. IEEE Commun. Surv. Tutor. **15**(2), 909–928 (2013). Second Quarter 2013
4. Kiriha, Y., Nishihara, M.: Survey on data center networking technologies. IEICE Trans. Commun. **E96–B**(3), 713–721 (2013)

5. Shang, Y., Li, D., Xu, M.: Energy-aware routing in data center network. In: Proceedings of the First ACM SIGCOMM Workshop on Green networking (Green Networking '10), pp. 1–8. ACM, New York (2010). doi:10.1145/1851290.1851292
6. Li, D., Xu, M., Liu, Y., Xie, X., Cui, Y., Wang, J., Chen, G.: Reliable multicast in data center networks. IEEE Trans. Comput. **63**(8), 2011–2014 (2014). doi:10.1109/TC.2013.91
7. Chen, K., Guo, C., Wu, H., Yuan, J., Feng, Z., Chen, Y., Lu, S., Wu, W.: DAC: generic and automatic address configuration for data center networks. IEEE/ACM Trans. Netw. **20**(1), 84–99 (2012). doi:10.1109/TNET.2011.2157520
8. Yu, Y., Fang, S., Aung, K.M.M., Foh, C.H., Li, H., Zhu, Y.: A layer 2 multipath solution and its performance evaluation for Data Center Ethernets. Int. J. Commun. Syst. **27**, 2555 (2013). doi:10.1002/dac.2488
9. Zhang, Y., Ansari, N.: On architecture design, congestion notification, TCP incast and power consumption in data center. IEEE Commun. Surv. Tutor. **15**(1), 39–64 (2013)
10. Zhang, J., Ren, F., Yue, X., Shu, R., Lin, C.: Sharing bandwidth by allocating switch buffer in data center networks. IEEE J. Sel. Areas Commun. **32**(1), 39–51 (2014). doi:10.1109/JSAC.2014.140105
11. Wu, W., Crawford, M.: Potential performance bottleneck in Linux TCP. Int. J. Commun. Syst. **20**, 1263–1283 (2007). doi:10.1002/dac.872
12. Liu, C.H., Kind, A., Liu, T.: Summarizing data center network traffic by partitioned conservative update. IEEE Commun. Lett. **17**(11), 2168–2171 (2013). doi:10.1109/LCOMM.2013.091913.130094
13. Qin, Y., Shi, Y., Sun, Q., Zhao, L.: Analysis for unfairness of TCP outcast problem in data center networks. In: 2013 25th International Teletraffic Congress (ITC), pp. 1–4 (2013). doi:10.1109/ITC.2013.6662965
14. Tahiliani, R.P., Tahiliani, M.P., Sekaran, K.C.: TCP variants for data center networks: a comparative study. In: Proceedings of the 2012 International Symposium on Cloud and Services Computing (ISCOS '12), pp. 57–62. IEEE Computer Society, Washington (2012). doi:10.1109/ISCOS.2012.38
15. Chen, Y., Griffith, R., Liu, J., Katz, R.H., Joseph, A.D.: 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. ACM, New York (2009). doi:10.1145/1592681.1592693. http://doi.acm.org/10.1145/1592681.1592693
16. Phanishayee, A., Krevat, E., Vasudevan, V., Andersen, D.G., Ganger, G.R., Gibson, G.A., Seshan, S.: Measurement and analysis of TCP throughput collapse in cluster-based storage systems. In: Baker, M., Riedel, E. (eds.) Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08), p. 14. USENIX Association, Berkeley (2008). Article 12
17. Lee, J., Cha, H., Ha, R.: A time-dominated TCP congestion control over heterogeneous networks. Int. J. Commun. Syst. **21**, 1325–1345 (2008). doi:10.1002/dac.957
18. Zhang, J., Ren, F., Tang, L., Lin, C.: Taming TCP incast throughput collapse in data center networks. In: Proceedings of 21st International Conference on Network Protocols (2013)
19. Zhang, J., Wen, J., Wang, J., Zhao, W.: TCP-FITDC: an adaptive approach to TCP incast avoidance for data center applications. 2013 International Conference on Computing, Networking and Communications (ICNC), pp. 1048–1052 (2013)
20. Shukla, S., Chan, S., Tam, A.S.W., Gupta, A., Yang, X., Chao, H.J.: TCP PLATO: packet labelling to alleviate time-out. IEEE J. Sel. Areas Commun. **32**(1), 65–76 (2014). doi:10.1109/JSAC.2014.140107
21. Zhang, J., Ren, F., Tang, L., Lin, C.: Modeling and solving TCP incast problem in data center networks. IEEE Trans. Parallel Distrib. Syst. **99**, 1 (2015). doi:10.1109/TPDS.2014.2310210
22. Wu, H., Feng, W., Guo, C., Zhang, Y.: ICTCP: incast congestion control for TCP in data-center networks. IEEE/ACM Transactions on Networking **21**(2), 345–358 (2013). doi:10.1109/TNET.2012.2197411
23. Ming, L., Lukyanenko, A., Tarkoma, S., Yla-Jaaski, A.: MPTCP incast in data center networks. Communications **11**(4), 25–37 (2014). doi:10.1109/CC.2014.6827566
24. Wang, G., Ren, Y., Dou, K., Li, J.: IDTCP: an effective approach to mitigating the TCP incast problem in data center networks. Inf. Syst. Front. **16**(1), 35–44 (2014). doi:10.1007/s10796-013-9463-4
25. Alizadeh, M., Greenberg, A., Maltz, D.A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., Sridharan, M.: Data center TCP (DCTCP). SIGCOMM Comput. Commun. Rev. **40**(4), 63–74 (2010). doi:10.1145/1851275.1851192. http://doi.acm.org/10.1145/1851275.1851192
26. Hwang, J., Yoo, J., Choi, N.: IA-TCP: a rate based incast-avoidance algorithm for TCP in data center networks. In: 2012 IEEE International Conference on Communications (ICC), pp. 1292–1296 (2012)
27. Ko, E., An, D., Yeom, I., Yoon, H.: Congestion control for sudden bandwidth changes in TCP. Int. J. Commun. Syst. **25**, 1550–1567 (2012). doi:10.1002/dac.1322
28. Hashimoto, M., Hasegawa, G., Murata, M.: An analysis of energy consumption for TCP data transfer with burst transmission over a wireless LAN. Int. J. Commun. Syst. (2014). doi:10.1002/dac.2832
29. Hou, T.-C., Hsu, C.-W., Wu, C.-S.: A delay-based transport layer mechanism for fair TCP throughput over 802.11 multihop wireless mesh networks. Int. J. Commun. Syst. **24**, 1015–1032 (2011). doi:10.1002/dac.1207
30. http://tools.ietf.org/html/rfc2001
31. http://tools.ietf.org/html/rfc6582
32. Vasudevan, V., Phanishayee, A., Shah, H., Krevat, E., Andersen, D.G., Ganger, G.R., Gibson, G.A., Mueller, B.: Safe and effective fine-grained TCP retransmissions for datacenter communication. In: Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication (SIGCOMM '09), pp. 303–314. ACM, New York (2009). doi:10.1145/1592568.1592604
33. Das, T., Sivalingam, K.M.: TCP improvements for data center networks. 2013 Fifth International Conference on Communication Systems and Networks (COMSNETS), pp. 1–10 (2013)
34. Tam, A.S.W., Xi, K., Xu, Y., Chao, H.J.: Preventing TCP incast throughput collapse at the initiation, continuation, and termination. In: Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service (IWQoS '12), p. 9. IEEE Press, Piscataway (2012). Article 29
35. Sreekumari, Prasanthi, Chung, Sang-Hwa: TCP NCE: a unified solution for non-congestion events to improve the performance of TCP over wireless networks. EURASIP J. Wirel. Commun. Netw. **2011**, 23 (2011)
36. Dixit, A., Prakash, P., Hu, Y.C., Kompella, R.R.: On the impact of packet spraying in data center networks. 2013 Proceedings IEEE INFOCOM, pp. 2130–2138 (2013). doi:10.1109/INFCOM.2013.6567015
37. Dixit, A., Prakash, P., Kompella, R.R.: On the efficacy of fine-grained traffic splitting protocolsin data center networks. SIGCOMM Comput. Commun. Rev. **41**(4), 430–431 (2011). doi:10.1145/2043164.2018504. http://doi.acm.org/10.1145/2043164.2018504
38. Gill, P., Jain, N., Nagappan, N.: Understanding network failures in data centers: measurement, analysis, and implications. Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11). ACM, New York (2011). doi:10.1145/2018436.2018477. http://doi.acm.org/10.1145/2018436.2018477
39. Vamanan, B., Hasan, J., Vijaykumar, T.N.: Deadline-aware datacenter TCP (D2TCP). In: Proceedings of the ACM SIGCOMM

2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '12), pp. 115–126. ACM, New York (2012)
40. http://web.scalable-networks.com/content/qualnet
41. Park, Mi-Young: Chung, Sang-Hwa: TCP's dynamic adjustment of transmission rate to packet losses in wireless networks. EURASIP J. Wirel. Commun. Netw. **2012**, 304 (2012)
42. Jiang, C., Li, D., Xu, M.: LTTP: an LT-code based transport protocol for many-to-one communication in data centers. IEEE J. Sel. Areas Commun. **32**(1), 52–64 (2014). doi:10.1109/JSAC.2014.140106

Prasanthi Sreekumari received the B.Sc. degree in Physics from Kerala University, India, in 2000, the M.C.A. degree from Madurai Kamaraj University, India, in 2003, the M.Tech. degree from JRN Rajasthan Vidyapeeth Deemed University, India, in 2006 and the Ph.D. degree in Computer Engineering from Pusan National University, Busan, South Korea, in 2012. After receiving her Ph.D. degree, she was a postdoctoral researcher at the Department of Computer Science and Engineering, Ehwa Womans University, Seoul, South Korea, from March 2012 to August 2014. From September 2014 to February 2016, she was a Research Professor at the Department of Electronics and Computer Engineering, Hanyang University, Seoul, South Korea. Currently she is working as a Visiting Professor in the Department of Computer Science, Grambling State University, Grambling, Louisiana, USA. Her research interests include Network Protocols, Congestion Control, Data Center Networks and Wireless Networks.

Jae-il Jung received the B.S. degree in Electronic Engineering from Hanyang University, Seoul, South Korea, in 1981, the M.S. degree in Electrical and Electronic Engineering from Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1984, and the Ph.D. degree in Computer Science and Networks from Ecole Nationale Superieure des Telecommunications (ENST), Paris, France, in 1993. After receiving his M.S. degree, he was with Telecommunication Network Research Labs, Korea Telecom, from 1984 to 1997. He is currently a Professor at Hanyang University. His research interests include Wireless Networks and Vehicular IT Services, especially in VANET Networks and V2X Communications.

Meejeong Lee received the B.S. degree in Computer Science from Ewha Womans University, Seoul, South Korea, in 1987, the M.S. degree in Computer Science from University of North Carolina, Chapel Hill in 1989, and the Ph.D. degree in Computer Engineering from North Carolina State University, Raleigh, in 1994. In 1994, she joined the Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea, where she is currently a Professor. She has been engaged in research in the field of Computer Communication and Networks, and Performance Modeling and Evaluation. Her current research interests focus on Routing and Transport Protocols for Multimedia Traffic Transmission over Wireless Mobile Networks and Internet.