Photon Netw Commun
DOI 10.1007/s11107-015-0526-y

An early congestion feedback and rate adjustment schemes for many-to-one communication in cloud-based data center networks

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

Received: 25 September 2014 / Accepted: 18 June 2015

Abstract Cloud data centers are playing an important role for providing many online services such as web search, cloud computing and back-end computations such as MapReduce and BigTable. In data center network, there are three basic requirements for the data center transport protocol such as high throughput, low latency and high burst tolerance. Unfortunately, conventional TCP protocols are unable to meet the requirements of data center transport protocol. One of the main practical issues of great importance is TCP Incast to occur many-to-one communication sessions in data centers, in which TCP experiences sharp degradation of throughput and higher delay. This important issue in data center networks has already attracted the researchers because of the development of cloud computing. Recently, few solutions have been proposed for improving the performance of TCP in data center networks. Among that, DCTCP is the most popular protocol in academic as well as industry areas due to its better performance in terms of throughput and latency. Although DCTCP provides significant performance improvements, there are still some defects in maintaining the queue length and throughput when the number of servers is too large. To address this problem, we propose a simple and efficient TCP protocol, namely NewDCTCP as an improvement of DCTCP in data center networks. NewDCTCP modified the congestion feedback and window adjusting schemes of DCTCP to mitigate the TCP Incast problem. Through detailed QualNet experiments, we show that NewDCTCP significantly outperforms DCTCP and TCP in terms of goodput and latency. The experimental results also demonstrate that NewDCTCP flows provide better link efficiency and fairness with respect to DCTCP.

Keywords Cloud computing · Data center networks · TCP Incast

✉ Meejeong Lee
lmj@ewha.ac.kr

Prasanthi Sreekumari
s.prasanthy@gmail.com

Jae-il Jung
jijung@hanyang.ac.kr

1 Department of Electronics and Computer Engineering, Hanyang University, Seoul, South Korea

2 Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea

1 Introduction

Cloud computing is emerging as an attractive Internet service model [1]. A data center network is the key infrastructure of cloud computing and big data, which has facilities with hundreds of thousands of servers and network equipments to provide different kinds of services and applications for government systems, academic and communications [2]. 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. With the rise of cloud computing, data center services are in high demand and large IT enterprises such as Google, Amazon, IBM and Microsoft have built their own data centers to provide cloud services, which becomes an important role in the future growth of Information and Communications Technology (ICT) industry [3,4]. Data center networks are finely designed and layered to achieve high bandwidth and low latency. As a result, the data center environment is highly different than that of the Internet, especially in terms of round trip time (RTT) [5]. The main characteristics of a data center network are high-speed links, limited-size switch buffers, low propagation delays and fault tolerance. The performance of data centers is significantly affected by the communication networks, since all the applications of data center networks are distributed in nature [6]. As a result, it is important to analyze the characteristics of data center traffic for designing efficient networking mechanisms for data centers.

Data center traffic can be classified into three types, namely elephant traffic, mice traffic and cat traffic [5]. According to recent studies [5–7], above 90 % of network traffic in a data center is contributed by transmission control protocol (TCP) due to its reliability of data transmission. In data center network, there are three basic requirements for the data center transport such as high throughput, low latency and high burst tolerance. Unfortunately, numerous deficiencies have been observed about the performance of TCP in data center networks. One of the main practical issues of great importance to the performance degradation of TCP in data center networks is TCP Incast, i.e., the drastic throughput reduction when multiple senders communicate with a single receiver [8]. When the number of servers is large, this transmission pattern may overflow the buffer at the switch, causing packet losses and timeouts.

Recently, few solutions [6–14] have been proposed for improving the performance of TCP in data center networks. Among those solutions, data center TCP (DCTCP) is the most popular protocol in academic as well as industry areas, proposed as a TCP replacement in data center environment [11]. Microsoft introduces DCTCP in Windows Server 2012 [15]. DCTCP uses a simple marking mechanism at switches based on the explicit congestion notification (ECN) [16] algorithm and a few amendments at end hosts to control the congestion window based on the congestion level of the network. Thus, DCTCP can achieve high throughput and low latency. Although DCTCP provides significant performance improvements in throughput as well as latency, it tends to provide lower throughput when the number of servers is too large. It is because when the number of servers is small, switch takes only less amount of buffer space to hold all the incoming packets [7,9,10]. On the other hand, when the number of servers is large, DCTCP is unable to mitigate the TCP Incast problem due to buffer overflow at the switch, which leads to the drop of few or more packets in the network [7]. In addition, some parameters and network configuration of DCTCP show severe oscillations in queue length and degrade the performance [10].

To address the above problems of DCTCP, we propose a new data center TCP (NewDCTCP) protocol for improving the performance of TCP in data center networks in terms of goodput, latency, link efficiency and fairness. NewDCTCP mainly consists of two schemes:

- *Early congestion feedback* This scheme is used for notifying a more up-to-date congestion status to the senders by marking the packets using the mark-front strategy instead of mark-tail policy which is used by DCTCP.

The mark-front strategy helps the senders to avoid heavy congestion, queue oscillations and frequent timeouts in the network and thus reduces the performance degradation problem of TCP due to TCP Incast.

- *Rate adjustment* This scheme is used to control the sending rate by modifying congestion window size and maximum segment size based on the congestion in the network.

The QualNet-based experiments in a typical TCP Incast scenario show that the performance of NewDCTCP significantly outperforms DCTCP and TCP (NewReno) in terms of goodput and timeouts with the increase in the number of servers in many-to-one communication pattern. We also demonstrate that NewDCTCP flows provide better link efficiency and fairness with respect to DCTCP.

The remainder of the paper is organized as follows. In Sect. 2, we present a detail description about data center traffic and TCP Incast in data center networks. We then discuss the motivation including the related work and the limitations of existing ECN-based data center transport protocols in Sect. 3. In Sect. 4, we discuss the NewDCTCP algorithm in detail. The performance improvement achieved by NewDCTCP algorithm compared to DCTCP and TCP in terms of goodput, latency, link efficiency and fairness is presented in Sect. 5. Finally, Sect. 6 concludes our paper.

2 Data center traffic and TCP Incast

In this section, we present the traffic pattern of data center networks and the major problem that cause the performance degradation of TCP in data center networks.

2.1 Data center traffic

Figure 1 shows the conventional network architecture for data centers which is adapted from Cisco [17]. At the top of the hierarchy, core routers (CR) carry data requests from the clients in the Internet and are routed to aggregation switches (S) through the access routers (AR) based on the destination virtual IP addresses. The high-degree aggregation switches forward the data requests to the inexpensive top-of-rack (TOR) switches which provide 1 Gbps link connectivity to servers mounted on every rack. Each rack typically consists of tens of servers. The switches at the layer 2 contain two load balancers (LB) to provide the list of private and internal addresses of physical servers in the racks. This list defines the pool of servers that can handle requests to that virtual IP addresses, and the load balancers spread request across the servers in the pool. These days the cloud data center

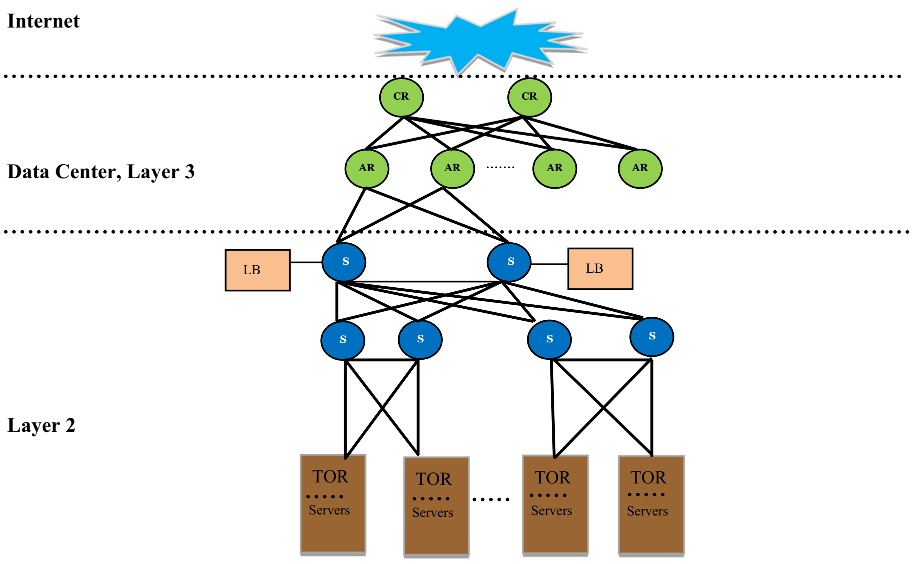


Fig. 1 Conventional architecture of data center network

Table 1 Types of data center traffic

Traffic	Size	Applications
Elephant	1–50 MB	Video-demand, Software updates
Mice	2 KB	Facebook, Google
Cat	1 MB	YouTube

applications often follow the partition/aggregate traffic pattern. Under this traffic pattern, the higher-level aggregators receive the data request from the client and then partition into several pieces and transfer to the workers via the lower-level aggregators. According to [5,14,18], the data center traffic can be categorized into three types:

1. *Elephant traffic* The large background flows form the elephant traffic. In data center network, the medium number of concurrent large flows is 1 [12].
2. *Mice traffic* It consists of latency critical flows. Majority of the traffic is mice traffic or query traffic in data center network.
3. *Cat traffic* Short message flows which are normally used to update control state on the workers. Cat traffic is time-sensitive.

Table 1 presents the different data center traffic types, message sizes and applications. As we mentioned in the above section, a data center transport needs three basic requirements such as high throughput, low latency and high burst tolerance. However, TCP protocols implemented in data center networks fail to satisfy these requirements due to the problem of TCP Incast in many-to-one communication pattern of data center networks.

2.2 TCP Incast

TCP Incast is a catastrophic drop in the network throughput that occurs when multiple senders communicate with a single receiver in high-bandwidth, low-delay networks using TCP. TCP Incast issue is firstly found in the distributed storage system PanFS [8]. Figure 2 presents a typical TCP Incast scenario of data center networks. In this many-to-one communication pattern, the client sends barrier-synchronized data requests to multiple servers via a switch.

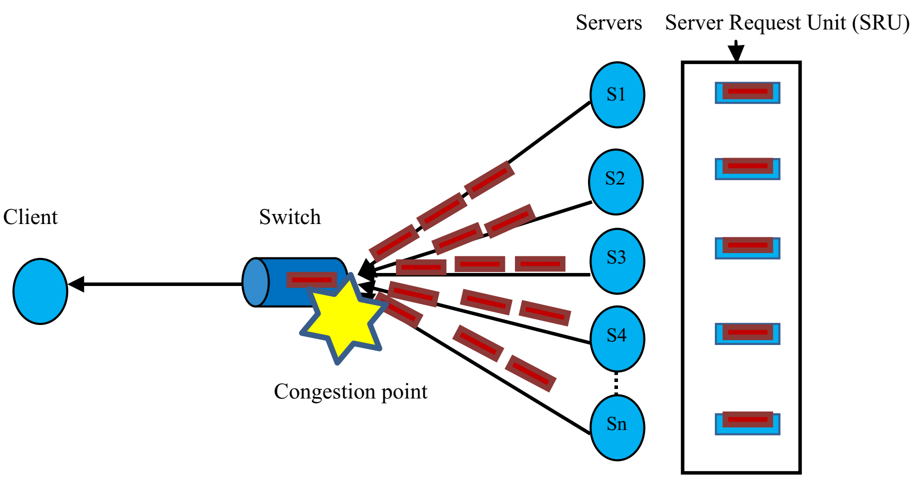
Each server stores a fragment of data block, which is referred to as server request unit (SRU). Upon receiving the data request, the servers (S) begin to transmit the requested data simultaneously and it traverse through a bottleneck link in many-to-one fashion to the client, results in the overflow of switch buffer size, leading to packet losses and retransmission timeouts. Effectively, the servers slow down the sending of requested data, leading to drastic reduction in throughput [19] which is termed as 'TCP Incast.' TCP Incast has been defined as the pathological behavior of TCP that results in gross under-utilization of the link capacity in various many-to-one communication patterns [13]. A nice summary of the preconditions for TCP Incast is stated in [20], where the preconditions are listed as follows:

- High-bandwidth, low-latency networks with small switch buffers.
- Clients that issue barrier-synchronized request in parallel.
- Servers that return a relatively small amount of data per request.

3 Motivation

Recently, TCP Incast becomes a hot research topic in our research community [1]. In order to improve the TCP performance by addressing the issue of TCP Incast in data center networks, several solutions [6–14] have been proposed from the aspects of different layers including application layer, transport layer and link layer. Among those solutions, we

Fig. 2 Scenario of TCP Incast



are interested in the transport layer solutions, particularly, by leveraging the ECN [16]-based packet marking mechanism for identifying the extent of network congestion. This section presents the recent ECN-based data center transport protocols proposed for mitigating the problem of TCP Incast in data center networks and discusses the limitations of existing ECN-based data center transport protocols.

3.1 Related work

As we mentioned in Sect. 2, the main reason of TCP Incast is the frequent retransmission timeouts caused by packet losses due to network congestion. As a result, detection of network congestion is important for mitigating the problem of TCP Incast.

The packet marking mechanism of ECN is one of the efficient and most popular way for detecting the congestion status of the network. In data center networks, few protocols are proposed for improving the performance of TCP by leveraging the packet marking scheme of ECN. In this subsection, we explain the ECN-based data center transport protocols proposed for solving the problem of TCP Incast in data center networks.

3.1.1 DCTCP

Alizadeh et al. [11] proposed a TCP like host-based protocol, named DCTCP, designed to operate with very low buffer occupancies, without the loss of throughput for data center networks. 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. For detecting network congestion, DCTCP adopts the packet marking mechanism of ECN. However, the standard ECN has some major limitations [21,22]. First, echoing back ECN information from the receiver to the sender takes time. Second, the notification of congestion from ECN may not be timely enough for the sender to make the correct decision which is suitable for the current network status under all circumstances [23]. By considering the limitations of standard ECN, DCTCP employs a very simple active queue management scheme to mitigate the problem of TCP Incast. In DCTCP, if the queue occupancy is greater than a threshold value, DCTCP marks the arriving packets based on the instantaneous queue length with the Congestion Experienced (CE) codepoint rather than using the average queue length like standard ECN. For sending the timely congestion notification to the sender, DCTCP sends acknowledgment (ACK) for every packets, setting the ECN-Echo flag if and only if the packet has a marked CE codepoint. With these two changes, DCTCP has greatly improved the throughput of TCP [11]. DCTCP needs only changes in the settings of a single parameter on the switches and 30 lines of code changes to TCP, which makes it easy to deploy.

3.1.2 FITDC

Zhang et al. [24] proposed an adaptive delay-based congestion control algorithm, named TCP-FITDC, to tackle the problem of TCP Incast in data center applications. The main goal of this design is to achieve high throughput, low latency and fast adaptive adjustment for TCP when deployed in data centers. To achieve this goal, TCP-FITDC proposed two schemes: marking scheme and adjusting scheme. The first scheme is motivated by DCTCP. TCP-FITDC utilized the modified packet marking mechanism defined in [11] from ECN as an indication of network buffer occupancy and buffer overflow of the switch. If the queue length is greater than a single threshold value ‘K,’ the sender receives a marked ACK, but unmarked otherwise. Using this scheme, the sender is able to maintain the queue length of the switch by detecting the ECN-Echo bits in ACKs. The second scheme of TCP-FITDC adjusts the sender’s congestion window for controlling the sending rate based on two classes of RTT values: RTT values without ECN-Echo flag and RTT values with ECN-Echo flag. Whenever the sender receives a RTT value without ECN-Echo flag, the sender increases its congestion window by assuming that the level of switch buffer does not exceed the threshold value.

On the other hand, whenever the sender receives a RTT value with ECN-Echo flag, the sender decreases the congestion window size to reduce the buffer length of the switch.

3.1.3 TDCTCP

In [6], Das et al. designed a data center transport protocol, named TDCTCP, 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 the dynamic delayed ACK calculation of TCP retransmission timer. First, TDCTCP modified the congestion control mechanism of DCTCP algorithm to control the congestion window in the congestion avoidance state according to the level of congestion in the network. Second, TDCTCP resets the value of congestion indicator to 0 after every delayed ACK timeouts for avoiding the stale value of congestion indicator. Third, for better adapting to the network conditions, TDCTCP calculates the delayed ACK timeouts dynamically. These modifications help the senders to react better to the current congestion state and provide better throughput.

3.1.4 FaST

In [25], Hwang et al. proposed a fine-grained and scalable (FaST) congestion control scheme using virtual congestion window for cloud data center applications. Through analysis, FaST revealed the scalability limits of the legacy approaches and observed that the congestion control should be fine-grained. FaST modified the DCTCP algorithm for improving the scalability by reducing the segment size according to virtual congestion window size. If the size of the minimum congestion window falls below one packet, FaST adjusts the amount of sending data by reducing the segment size to effectively mitigate network congestion. By doing this, FaST achieves low query completion times for the short flows generated by cloud applications, while still showing comparable average throughput for background traffic. In addition, this approach is simple to implement and the actual deployment is easy, as it requires only a small modification at the server side. Moreover, the authors realized that most of the TCP implementations could have a problem avoiding the silly window syndrome function which is generally used to avoid very small sending windows. In FaST, this function is turned off only for partition/aggregate cloud applications.

3.2 Limitations of existing ECN-based data center transport protocols

The existing ECN-based data center transport protocols used the technique of marking the incoming packets that just entered the queue when the queue length reaches the specified threshold value. This type of packet marking policy is called 'mark-tail.' One of the major limitations of this technique is its discrimination against new flows [26]. For example, consider the buffer of the congested switch/router is already occupied by the old flows and at that time, a new flow joins in the network. With the mark-tail technique, the router will mark all the incoming packets from the new flow rather than marking the packets which are already in the buffer. As a result, the umarked packets of old flows lead to increase the size of congestion window, resulting in the usage of more bandwidth, and the new flow has to backoff due to the reduction in congestion window size by the marked packets. This causes a 'lock-out' phenomenon [26,27].

That is, in some situations, packets mark or drop from the tail allows a single connection or a few flows to monopolize switch buffer space preventing other connections from accommodating packets in the queue [28]. Another limitation is the discrimination against connections with lower RTT flows [9]. For avoiding the limitations of existing ECN-based data center transport protocols, NewDCTCP uses mark-front policy for marking packets by modifying the mark-tail policy of DCTCP as it is the most popular transport protocol for data center networks. As shown in Fig. 3, in mark-front policy, when the queue length is greater than the threshold value 'K,' the switches mark the packets from the front of the queue rather than marking the incoming packets from the tail of the queue. This marking technique helps the sender to receive faster and up-to-date congestion signals than existing ECN-based protocols and thus avoids packet losses and frequent retransmission timeouts and thereby mitigate the problem of TCP Incast in data center networks. In the next section, we explain NewDCTCP in detail.

4 NewDCTCP algorithm

Based on the above discussions in Sect. 3, we design a new data center transport protocol called NewDCTCP for improving the performance of TCP by supporting many-to-one communication in data center networks. The primary objective of NewDCTCP is to mitigate the problem of TCP Incast by reducing the frequent retransmission timeouts and the loss of packets due to network congestion. For achieving our goal, we contribute two new schemes in NewDCTCP such as (1) early congestion feedback and (2) rate adjustment at the sources. The former scheme is the most important part of NewDCTCP which is used for conveying the incipient

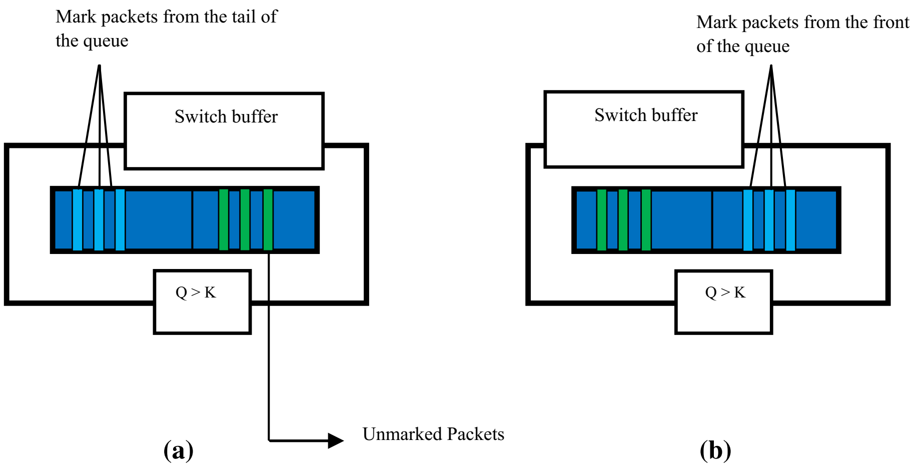

Mark packets from the tail of the queue
Switch buffer
Q > K
Unmarked Packets
(a)
Mark packets from the front of the queue
Switch buffer
Q > K
(b)

Fig. 3 Marking strategies **a** mark-tail and **b** mark-front

congestion signals to the sources as early as possible, and the latter part is used for controlling the sending rate for reducing the queue length at the switches based on the congestion level of the network. We now present our two schemes in the following subsections.

4.1 Early congestion feedback

To address the important limitations of existing ECN-based congestion notification mechanism of data center transport protocols, we propose an efficient packet marking scheme called ‘Early Congestion Feedback (ECF),’ which is able to inform the senders about the up-to-date congestion status of the network. In this scheme, we modified the packet marking mechanism of DCTCP by adopting the mark-front technique [27] instead of using the mark-tail strategy. In our ECF mechanism using mark-front strategy, the switches mark the data packets which are already in the buffer starting from the front of the queue rather than marking the incoming packets from the tail of the queue as shown in Fig. 3. Following are the advantages of ECF mechanism of NewDCTCP.

1. It helps to deliver congestion notification more faster than DCTCP packet marking mechanism.
2. It helps to control the buffer occupancy at the switch.
3. It helps to reduce the frequent retransmission timeouts.
4. It helps to improve link efficiency and fairness by avoiding the lock-out phenomenon.

As shown in Fig. 4, when the instantaneous queue length (Q) is greater than the threshold value ‘K,’ switches mark the packets which are already in the buffer starting from the front of the queue with CE codepoint and transfer to the receiver. The receiver informs the sender via ECN-Echo flag. As soon as the sender receives the ECN-Echo notification, it reduces the size of congestion window according to our rate adjustment scheme and sends the first new data packet.

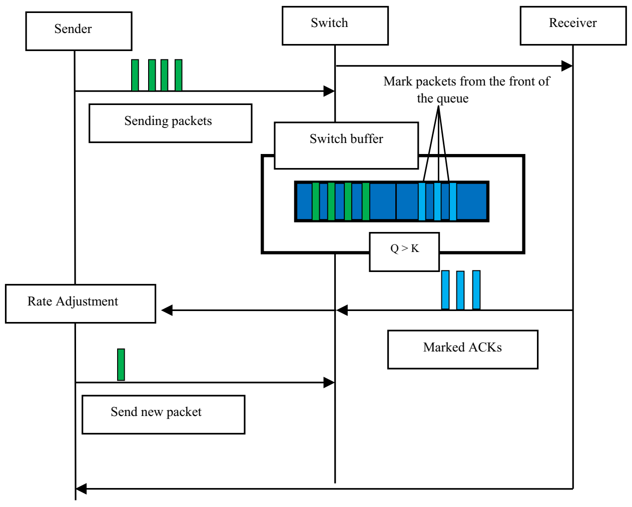


Fig. 4 Early congestion feedback of NewDCTCP

This faster and earlier congestion feedback helps the senders to reduce packet losses from sending more packets to the congested network and thereby increase the throughput as well as decrease the queuing delay.

4.2 Rate adjustment

In addition to earlier congestion notification, sending rate adjustment is another important factor for 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 [10]. As a result, switch marks lot of packets and the senders reduce their congestion window size frequently and reduce the performance. 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 proposes a fine-grained reduction function for reducing the size of congestion window based on the value of α.

In DCTCP, whenever the sender receives an ACK with ECE is set to 1, the sender reduces the congestion window (cwnd) using Eq. (1),

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

where α is calculated from the fraction of marked packets (F) and weight factor (g) according to 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 slightly reduces according to Eq. (1). However, in the latter case, DCTCP congestion window reduces 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 [29] shows that one of the main problems in the congestion window estimation of DCTCP is in the choice of α initialization value. If the value of α is set to zero, the sender can send as much as packets to the receiver which leads to packet losses and retransmission timeouts. On the other hand, if the value of α is set to one, the sender can minimize the queuing delay but the amount of packets to be transferred is much smaller. This will effect the throughput of DCTCP sender. As a result, the initial value of α depends on the applications that use DCTCP. Another problem is resetting the value of α after retransmission timeout. For solving this problem, recently, TDCTCP [6] modified the window adjustment of DCTCP for avoiding the stale value of congestion indicator. However, both DCTCP and TDCTCP needed additional calculations for estimating the size of congestion window at the sender side.

By considering the limitations of DCTCP window adjustment algorithm, we propose a simple and efficient mechanism for adjusting the sending rate of senders. In NewDCTCP, when the sender receives an unmarked ACK packet, the sender proceeds like TCP NewReno. On the other hand, if the sender receives a marked ACK packet, the sender estimates the congestion window size based on the formula (3),

$$\text{cwnd} = \max\,(\text{cwnd}/2, \alpha \times \text{mss}) \tag{3}$$

$$\alpha = \text{number of packet sent} - \text{number of marked ACK packets} \tag{4}$$

where mss is the maximum segment size, and α is the number of unmarked ACKs that were received in the last window of data.

Using Eq. (3), the congestion window slightly reduces if the number of unmarked ACKs is greater than half of the sent packets. Otherwise, the sender assumes that the congestion is severe and reduces the window into half for the safety side by considering the sudden increase in queue length due to burst of flows. This adjustment of congestion window helps the sender to control the queue length without using any additional calculations like DCTCP and TDCTCP at the sender side and helps to improve the performance in terms of goodput and delay even in the presence of large number of servers in the network. Moreover, we considered the situation that congestion window reduces its value to less than one packet. This situation is unavoidable in large-scale data centers [9]. If the congestion window consists of one mss and the sender receives a marked ACK packet, then NewDCTCP reduces its congestion window according to RFC 3168 [30].

4.3 Working rationale of NewDCTCP

In this section, we describe the working rationale of our NewDCTCP algorithm as shown in Fig. 5. We modified the DCTCP packet marking mechanism at the switch side and the response to marked packets at the sender side. We explain the algorithms at sender, switch and receiver sides in detail.

4.3.1 Sender side

Whenever the sender receives marked ACK packets, the sender controls the sending rate by invoking the rate adjustment scheme of NewDCTCP. Otherwise, the sender behaves same as TCP congestion control algorithms.

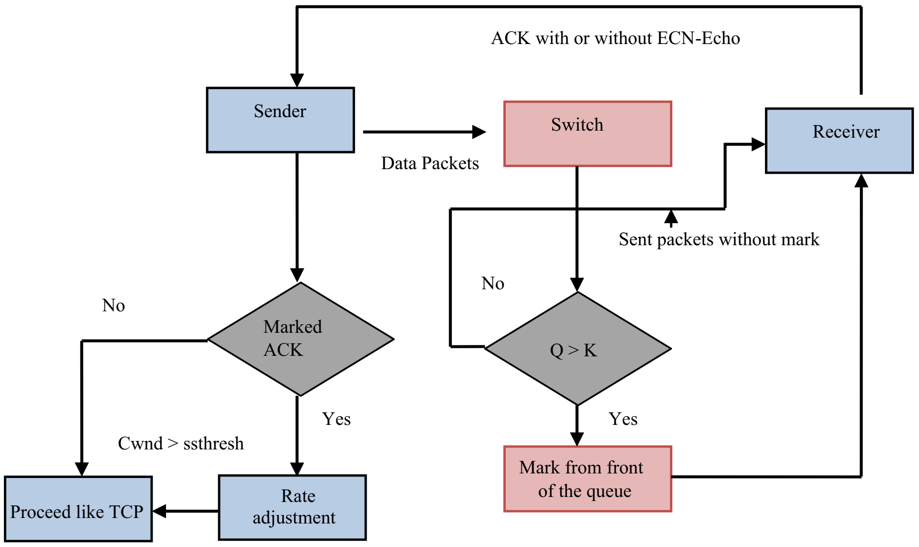
4.3.2 Switch side

At the time of receiving packets, the switch calculates the instantaneous queue length based on the single threshold value 'K' like DCTCP. If the queue length is greater than K, the switch marks the packets in the buffer from the front of the queue and sets the congestion experienced bit to the outgoing packets to the receiver.

4.3.3 Receiver side

NewDCTCP receiver is same like DCTCP receiver. That is, NewDCTCP receiver sends an ACK for every two packets if there is no congestion notification from the switch. On the other hand, if there is congestion notification, the receiver sends all marked ACK packets to the sender as shown in Fig. 6. With these simple modifications to sender as well as switch sides, we can overcome the limitations of DCTCP

Fig. 5 Working of NewDCTCP



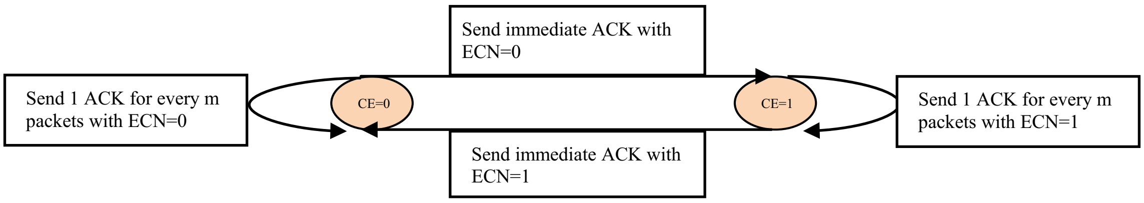

Send immediate ACK with ECN=0
Send 1 ACK for every m packets with ECN=0
CE=0
CE=1
Send 1 ACK for every m packets with ECN=1
Send immediate ACK with ECN=1

Fig. 6 ACK generation state machine [11]

and thereby increase the performance of TCP by mitigating Incast issue in data center networks.

5 Performance evaluation

In this section, we present the performance of our proposed protocol NewDCTCP through comprehensive simulations using QualNet simulator [31]. We compare the performance of NewDCTCP with DCTCP as it is the most popular data center transport protocol and with NewReno [32] as it is the widely used protocol in practice [33]. We implemented DCTCP in QualNet using the source code got from [34]. TCP NewReno is readily available in QualNet. We first describe our evaluation methodology including topology, parameter settings and performance metrics in Sect. 5.1. Then, we presented the evaluation results in Sect. 5.2.

5.1 Methodology

5.1.1 Topology

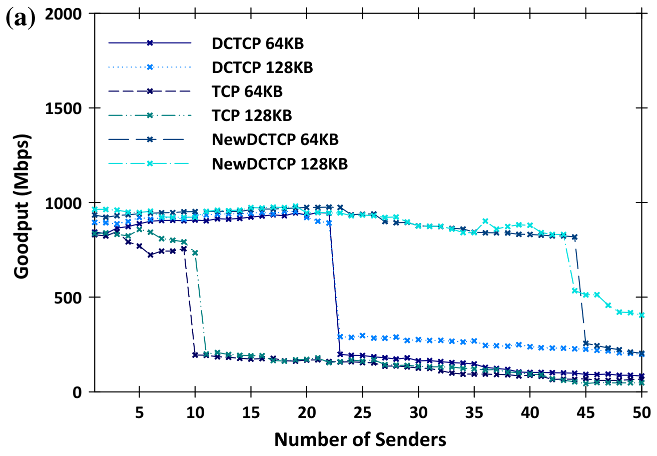
Our main goal of this work is to increase the throughput of TCP by mitigating the problem of TCP Incast which is caused by partition/aggregate traffic pattern of data center networks [35]. For achieving our goal, we evaluate the performance of NewDCTCP in a typical network topology for partition/aggregate cloud applications as shown in Fig. 2. In this scenario, the client requests a chunk of data from 'N' servers. When the client receives the full amount of requested data from the servers, it sends another request. However, if the response data from the servers overflow the buffer space, it causes packet loss. As specified in [11,20], the workload we used in our simulation is fixed volume per server to compare the performance of NewDCTCP with DCTCP and NewReno.

5.1.2 General parameter settings

To simulate the incast scenario, we used 50 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 and 256 KB. We vary the SRU size from 10 to 128 KB. The marking threshold value 'K' is set according to [7,11] 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.

5.1.3 Performance metrics

To evaluate the performance of NewDCTCP with DCTCP and TCP, we use four important performance metrics as mentioned in [10]. First, we calculated the goodput as the ratio of



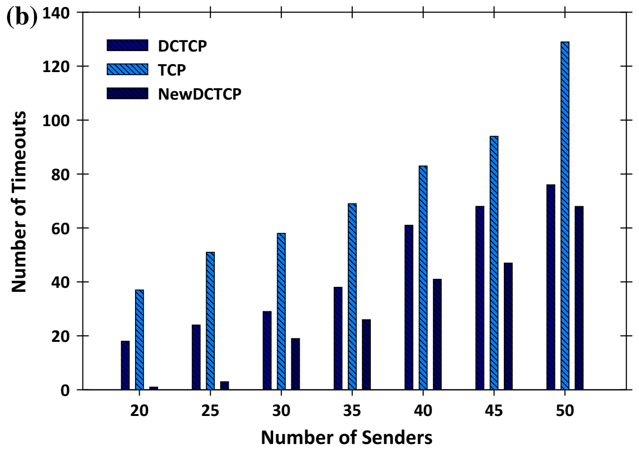


Fig. 7 Comparison of **a** goodput and **b** timeouts of NewDCTCP, DCTCP and TCP with buffer size 64 KB

the total data transferred by all the servers to the client and the time required to complete the data transfer. Second, we evaluated the performance of the above variants in terms of link efficiency and it is calculated from the number of acknowledged packets (not counting the retransmissions) divided by the possible number of packets that can be transmitted during the simulation. Third, we evaluate the flow completion time as specified in [36] and finally we evaluated the fairness of NewDCTCP and DCTCP using Jain's fairness index (JFI) [37]. The Jain fairness index (JFI) function is expressed as below,

$$F(x_1, \ldots, x_N) = \left(\sum_{i=1}^{N} x_i\right)^2 \Big/ N \times \sum_{i=1}^{N} (x_i)^2$$

where x_i is the goodput of the ith connection, and N is the number of connections.

5.2 Results

In this section, we present the results of our evaluation of NewDCTCP by comparing it with DCTCP and TCP in terms of goodput, flow completion time, link efficiency and fairness using a single bottleneck TCP Incast scenario with two different switch buffer sizes and various SRU sizes as stated in the previous subsection. In addition, we present the performance of NewDCTCP in terms of timeouts as it is the main cause of TCP Incast in data center networks.

Figure 7a shows the performance of NewDCTCP compared to DCTCP and TCP in terms of goodput. The buffer size we set for this simulation is 64 KB with SRU sizes 64 and 128 KB. From the result, we observe that even we used a smaller buffer size, the performance of NewDCTCP is identical to 64 and 128 KB SRU till about 45 senders. That means, the goodput of NewDCTCP does not vary much with different SRU sizes. However, we see a sudden drop in the goodput of TCP and DCTCP as the number of senders increases from 10 for TCP and around 25 for DCTCP. This is due to packet losses caused by the overflow of small switch buffer size. In our experiment for SRU size 64 KB, the maximum goodput of TCP is around 850 Mbps, while that of DCTCP is around 945 Mbps. However, NewDCTCP achieves a goodput of 981 Mbps. On the other hand, when the SRU size is 128 KB, the goodput of DCTCP is reduced to 300 Mbps around 20 senders, while NewDCTCP maintains higher performance. One of the main reasons for this achievement of NewDCTCP is its earlier congestion feedback and rate adjustment schemes.

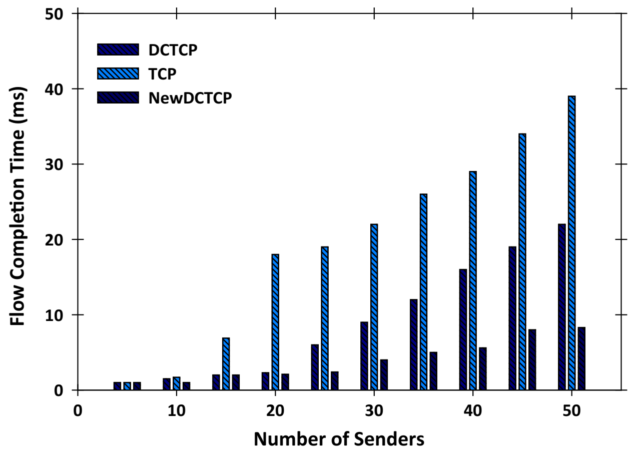


Fig. 8 Comparison of flow completion time

In Fig. 7b, we present the timeouts comparison of NewDCTCP using buffer size 64 KB and SRU size 128 KB. As we expected, NewDCTCP suffers only less number of timeouts compared to DCTCP and TCP. This is because NewDCTCP can efficiently control the queue length and thereby allocates more packets into the queue than DCTCP and TCP and saves the packets from queuing delay. Figure 8 presents the flow

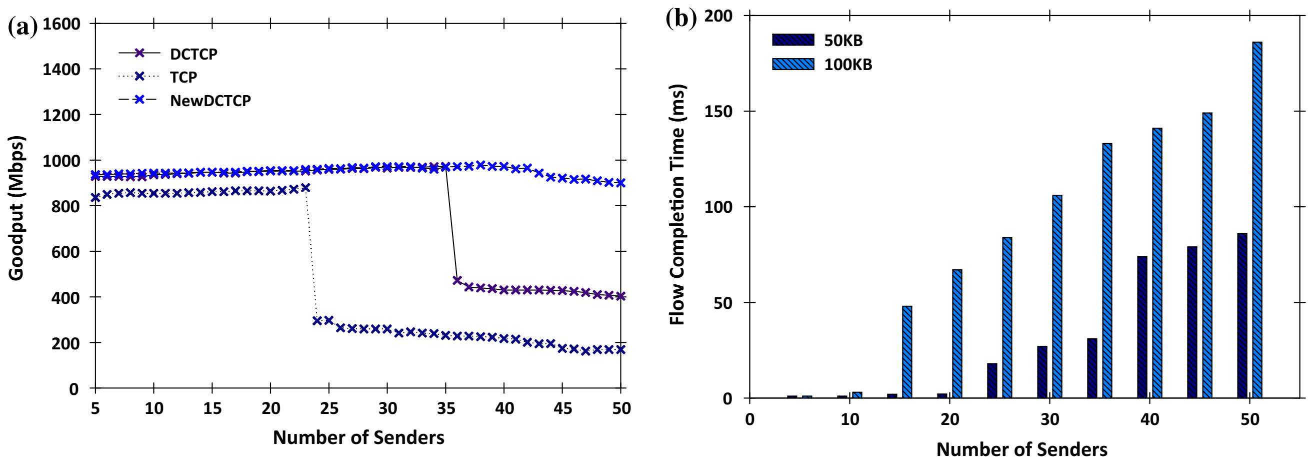


Fig. 9 Comparison of **a** goodput and **b** the flow completion time with buffer size 256 KB

completion time of NewDCTCP compared to DCTCP and TCP with SRU size 10 KB and buffer size 256 KB. The flow completion time is influenced by the amount of queuing delay and retransmission time [36]. From the result, we can see that the flow completion time increases as the number of senders increases. Compared to DCTCP and TCP, the flow completion time of NewDCTCP is less than 10 ms when the number of sender increases to 50. It means that the marking scheme of NewDCTCP saves the senders from very large number of packet drops and frequent timeouts. In the case of DCTCP, the flow completion time is around 25 ms as the number of senders increases to 50, while that of TCP is 40 ms.

This is because TCP suffers more packet drops and timeouts due to its inability to detect the congestion of the network before packet drop. As a result, in the presence of network congestion either it is light or heavy, TCP can detect upon the arrival of three duplicate ACKs or timeouts. On the other hand, DCTCP is able to detect the congestion very efficiently. However, the fine-grained reduction function of DCTCP sometimes estimates the size of congestion window size inaccurately and thereby increases the queuing delay which results in packets drop.

Figure 9a presents the goodput of NewDCTCP compared to DCTCP and TCP with SRU 128 KB and buffer size 256 KB. Compared to the result in Fig. 7a, in Fig. 9, we observe that the performance of NewDCTCP, DCTCP and TCP improves with increase in buffer size. However, the goodput of DCTCP degrades around 475 Mbps after 35 senders. The reason is, the senders of DCTCP suffer from lot of packet drops due to its overestimation of congestion window size. On the other hand, NewDCTCP still outperforms the performance of DCTCP and TCP due to its early congestion feedback using mark-front strategy and rate adjustment mechanisms according to the level of network congestion.

Figure 9b depicts the performance of NewDCTCP in terms of flow completion time with SRU size 50 and 100 KB. For

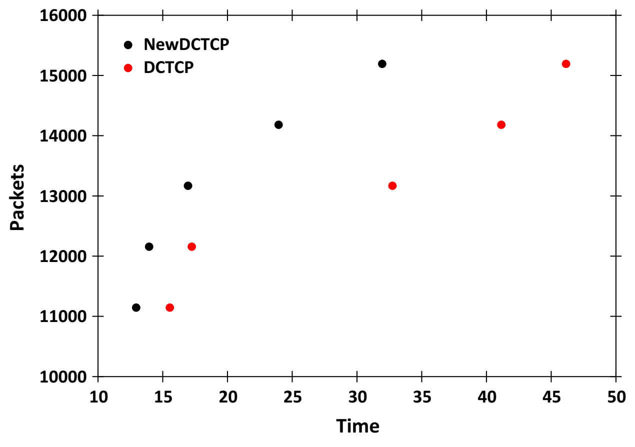


Fig. 10 Comparison of the marking schemes of NewDCTCP and DCTCP

this experiment, we set the buffer size to 256 KB. From the result, we observed that the flow completion time of NewDCTCP with SRU size 50 KB is lesser than with SRU size 100 KB. As we mentioned above, the flow completion time increases as the number of senders increases. With SRU size 50 KB, the flow completion time is in the order of less than 100 ms upto 50 senders and less than 50 ms upto 35 senders. In the case of SRU size 100 KB, the flow completion time still maintains less values. It means that NewDCTCP can achieve better goodput by reducing frequent retransmission timeouts. Figure 10 shows the packet marking mechanism of NewDCTCP and DCTCP. That means how early the NewDCTCP marked the packet compared to DCTCP. For this experiment, we trace the same packets marked by both protocols. In the traced result, we can clearly find that the marking scheme of NewDCTCP is much faster than DCTCP.

Next, we evaluate the protocols in terms of link efficiency and fairness. Figure 11a shows the link efficiency of NewDCTCP with DCTCP according to varying number of

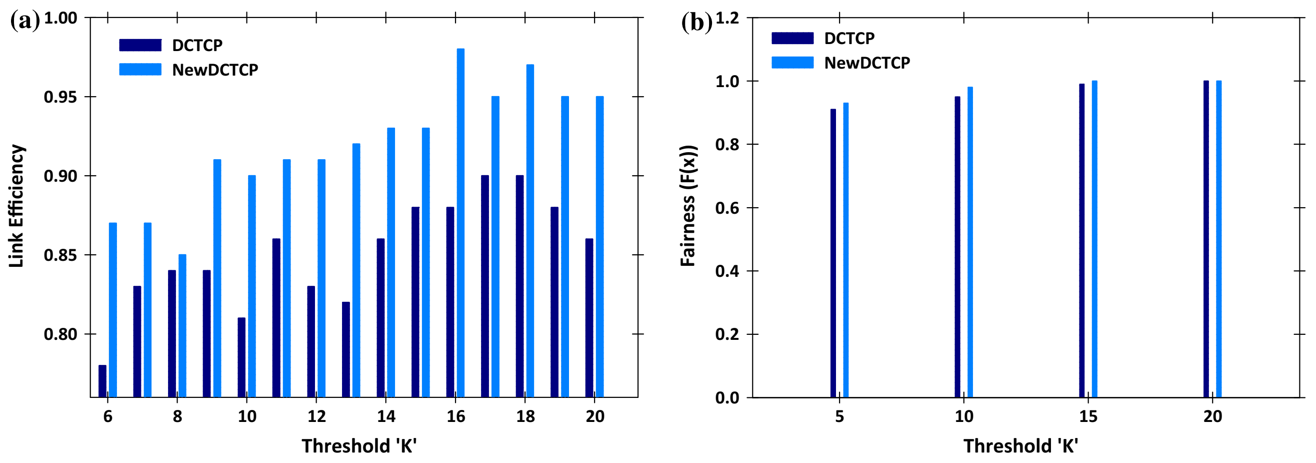


Fig. 11 Comparison of **a** link efficiency and **b** fairness of NewDCTCP with DCTCP

threshold values ranging from 6 to 20. From the results, we observed that the link efficiency increases when the value of threshold increases. Compared to DCTCP, NewDCTCP can utilize the link more efficiently according to different thresholds. In small value of thresholds, the link utilization of DCTCP is slightly low due to the reduction in congestion window size unnecessarily. That is, the sender receives the congestion signal falsely. However, the link efficiency of NewDCTCP is better in low as well as high values of thresholds. When congestion is detected, the congestion feedback scheme of NewDCTCP marks the packets from the front of the queue which helps to prevent the servers from sending more packets to the congested switch and can avoid the packet drops and timeouts. In addition, the feedback time of the mark-front strategy is considerably shorter than that of the mark-tail method. As a result, the servers can maintain the lower queue length and can utilize the link efficiently.

Figure 11b presents the evaluation of NewDCTCP compared to DCTCP in terms of fairness. For this experiment, we use 10 connections and calculate the fairness using the Jain fairness index as we mentioned in the above section. The result shows that the fairness of NewDCTCP is satisfactorily compared to DCTCP.

6 Conclusion

In this paper, we have developed a modified DCTCP protocol called NewDCTCP for improving the performance of TCP by mitigating the TCP Incast problem in data center networks. NewDCTCP modified the packet marking and rate adjustment mechanisms of DCTCP for maintaining low queue length and high throughput in data center networks. Instead of using mark-tail strategy for marking packets, NewDCTCP uses mark-front strategy for sending early congestion feedback to the servers and thus avoids frequent timeouts. We conducted extensive simulation using QualNet to validate the performance and effectiveness of NewDCTCP compared to DCTCP and TCP in terms of goodput, flow completion time, link efficiency, timeouts and fairness. Our experimental results using the typical TCP Incast scenario show that NewDCTCP achieves significant improvement in goodput even when the number of servers is too large. Also, NewDCTCP achieves fairly satisfactory fairness index compared to DCTCP.

Acknowledgments The authors gratefully acknowledge the anonymous reviewers for their valuable comments and suggestions for improving the manuscript. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2011-0023177) and the ICT R&D program of MSIP/IITP (B0126-15-1051, A study on Hyper Connected Self-Organizing Network Infrastructure Technologies for IoT Service) and MSIP/IITP (12221-14-1005, Software Platform for ICT Equipments).

References

[1] Li, D., Xu, M., Liu, Y., Xie, X., Cui, Y., Wang, J., Chen, G.: Reliable multicast in data center networks. IEEE Trans. Comput. **99**, 1 (2013). doi:10.1109/TC.2013.91
[2] Zhang. Y., Ansari, N.: HERO: hierarchical energy optimization for data center networks. In: 2012 IEEE international conference on communications (ICC), pp. 2924–2928, 10–15 (2012). doi:10.1109/ICC.2012.6363830
[3] Shang, Y., Li, D., Xu, M.: A comparison study of energy proportionality of data center network architectures. In: 32nd International conference on distributed computing systems workshops (ICDCSW), 2012, pp. 1–7, 18–21 (2012). doi:10.1109/ICDCSW.2012.17
[4] Kant, Krishna: Data center evolution: a tutorial on state of the art, issues, and challenges. Comput. Netw. **53**(17), 2939–2965 (2009)
[5] 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, IEEE Computer Society, Washington, DC, USA, pp. 57–62 (2012). doi:10.1109/ISCOS.2012.38

[6] Das, T., Sivalingam, K.M.: TCP improvements for data center networks. In: Fifth international conference on communication systems and networks (COMSNETS), 2013, pp. 1–10, 7–10 (2013)
[7] 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)
[8] Zhang, Y., Ansari, N.: On architecture design, congestion notification, TCP incast and power consumption in data centers. IEEE Commun. Surv. Tutor. **15**(1), 39–64 (2013). (First Quarter 2013)
[9] Jiao, Zhang, Fengyuan, Ren, Xin, Yue, Ran, Shu, Chuang, Lin: Sharing bandwidth by allocating switch buffer in data center networks. IEEE J. Sel. Areas Commun. **32**(1), 39,51 (2014)
[10] Chen, W., Cheng, P., Ren, F., Shu, R., Lin, C.: Ease the queue oscillation: analysis and enhancement of DCTCP. In: IEEE 33rd international conference on distributed computing systems (ICDCS), 2013, pp. 450–459, 8–11 (2013). doi:10.1109/ICDCS.2013.22
[11] Alizadeh, M., Greenberg, G., Maltz, D.A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., Sridharan, M.: Data center TCP (DCTCP). In: Proceedings of SIGCOMM'10, pp. 63–74, New York, NY, USA, 2010. ACM (2010)
[12] Hwang, J., Yoo, J., Choi, N.: Deadline and incast aware TCP for cloud data center networks. Comput. Netw. (2014). doi:10.1016/j.comnet.2013.12.002
[13] Devkota, P., Reddy, A.L.N.: Performance of quantized congestion notification in TCP incast scenarios of data centers. In: IEEE International symposium on modeling, analysis and simulation of computer and telecommunication systems (MASCOTS), 2010, pp. 235–243, 17–19 (2010)
[14] 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, Germany (2013)
[15] http://technet.microsoft.com/en-us/library/hh997028.aspx
[16] Ramakrishnan, K., Floyd, S., Black, D.: RFC 3168: the addition of explicit congestion notification (ECN) to IP
[17] http://www.cisco.com/go/dce
[18] Abts, D., Felderman, B.: A guided tour through data-center networking. Queue. **10**(5), 10–14 (2012). doi:10.1145/2208917.2208919. http://doi.acm.org/10.1145/2208917.2208919
[19] Bari, M.F., Boutaba, R., Esteves, R., Granville, L.Z., Podlesny, M., Rabbani, M.G.: Data center network virtualization: a survey. IEEE Commun. Surv. Tutor. **15**(2), 909–928 (2013)
[20] 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 SIGCOMM09, Barcelona, Spain, 2009, pp. 303–314 (2009)
[21] Quet, P-F., Chellappan, S., Durresi, A., Sridharan, M., Ozbay, H., Jain, R.: Guidelines for optimizing multi-level ECN using fluid flow based TCP model. In: Proceedings of ITCOM2002 Quality of Service over Next Generation Internet (2002)
[22] Wu, H., Ju, J., Lu, G., Guo, C., Xiong, Y., Zhang, Y.: Tuning ECN for data center networks. In: Proceedings of the 8th international conference on emerging networking experiments and technologies (CoNEXT '12). ACM, New York, NY, USA, 25–36 (2012)
[23] Xu, K., Tian, Y., Ansari, N.: TCP-Jersey for wireless IP communications. IEEE J. Sel. A. Commun. **22**(4), 747–756 (2006). doi:10.1109/JSAC.2004.825989
[24] Zhang, J., Wen, J., Wang, J., Zhao, W.: TCP-FITDC: an adaptive approach to TCP incast avoidance for data center applications. In: International conference on computing, networking and communications (ICNC), 2013, pp. 1048–1052, 28–31 (2013)
[25] Hwang, J., Yoo, J.: FaST: fine-grained and scalable TCP for cloud data center networks. KSII Trans. Internet Inf. Syst. (TIIS) **8**(3), 762–777 (2014)
[26] Arora, RM.: TCP/IP networks with ECN over AQM. https://curve.carleton.ca/system/files/theses/26732.pdf
[27] Liu, C., Jain, R.: Improving explicit congestion notification with the mark-front strategy. Comput. Netw. **35**(2–3), 185–201 (2001). doi:10.1016/S1389-1286(00)00167-5
[28] https://tools.ietf.org/html/rfc2309
[29] https://eggert.org/students/kato-thesis.pdf
[30] Ramakrishnan, K., Floyd, S., Black, D.: The addition of explicit congestion notification (ECN) to IP, RFC 3168, Sept. 2001 (2001)
[31] http://web.scalable-networks.com/content/qualnet
[32] http://www.ietf.org/rfc/rfc3782.txt
[33] Anghel, A.S., Birke, R., Crisan, D., Gusat, M.: Cross-layer flow and congestion control for datacenter networks. In: Proceedings of the 3rd workshop on data center-converged and virtual ethernet switching (DC-CaVES '11), Yi Qian and Kurt Tutschku (Eds.). ITCP 44–62 (2011)
[34] http://dev.pyra-handheld.com/index.php
[35] Hwang, J., Yoo, J., Choi, N.: IA-TCP: a rate based incast-avoidance algorithm for TCP in data center networks. IEEE International conference on communications (ICC) **2012**, 1292–1296 (2012)
[36] Xu, H., Li, B.: RepFlow: minimizing flow completion times with replicated flows in data centers. Proceedings IEEE INFOCOM **2014**, 1581–1589 (2014)
[37] "Fairness measure," Dec. 2011, http://en.wikipedia.org/wiki/Fairnessmeasure. (2011)

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 2012 to 2014. She is currently a Research Professor at the Department of Electronics and Computer Engineering, Hanyang University, Seoul, South Korea. 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.