Photon Netw Commun
DOI 10.1007/s11107-015-0550-y

Transport protocols for data center networks: a survey of issues, solutions and challenges

Prasanthi Sreekumari[1] · **Jae-il Jung**[1]

Received: 12 February 2015 / Accepted: 6 August 2015

Abstract In recent years, data centers play an important role in academia and industry for supporting various services and applications. Compared with other IP networks, data center networks have some special features such as many-to-one communication pattern with high bandwidth, low latency, auto-scaling, shallow buffered switches and multi-rooted tree topology. Owing to these special features of data center networks, traditional TCP suffers from severe performance degradation. For improving the performance of TCP in data center networks, various solutions have been proposed in recent years. This article presents a comprehensive survey of existing transport layer solutions proposed for mitigating the problems of TCP in data center networks. The objective of this article is threefold: to discuss about the issues of TCP in data center networks; to introduce various transport layer solutions and finally to compare and discuss the challenges of existing solutions proposed for improving the performance of TCP in data center networks.

Keywords Data center networks · TCP · Performance degradation · Issues · Survey

1 Introduction

Data center networks are the cost-effective communication infrastructure used in a data center for storing large volumes of data and providing efficient service applications [1].

✉ Jae-il Jung
jijung@hanyang.ac.kr

Prasanthi Sreekumari
s.prasanthy@gmail.com

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

Data centers are becoming essential computing platforms for IT enterprises to provide real-time response when carrying out various operations such as system configuration and query services [2,3]. Compared with other IP networks, data center networks have some special features such as many-to-one communication pattern with high bandwidth, freedom of deciding the endpoints of traffic flows, low round trip time (RTT) and the regularity of the topology [4]. Recently, many large data centers have been built to support hundreds of thousands of users, to host online services such as web query, distributed file system, distributed execution engine and structured storage system and also to provide the infrastructure services such as MapReduce, Bigtable and Dryad [5–7]. The current data centers are based on commodity switches for the interconnection network and their supporting storage based on high- speed links to a very connected world [8].

Figure 1 shows the architecture of a current data center network [9]. This three-tier architecture consists of three levels of switches such as core at the root, aggregation at the middle and edge switches connecting to the hosts for receiving and sending back the data from the client and servers. The main advantage of this architecture is that they are fault tolerant and can be scaled easily. Typically, the edge switches provides connectivity to thousands of servers which are accommodated into racks using 1 Gbps links capacity. These switches are further interconnected to multiple aggregate switches using 10 Gbps links capacity for redundancy. The aggregator switches are connected to core switches which provide security to connections and forward the requested data from the servers to the clients. The performances of data center networks are significantly affected by the communication network, since all the applications are distributed in nature. Generally, the applications of data centers create large data flows and small control flows that require high throughput

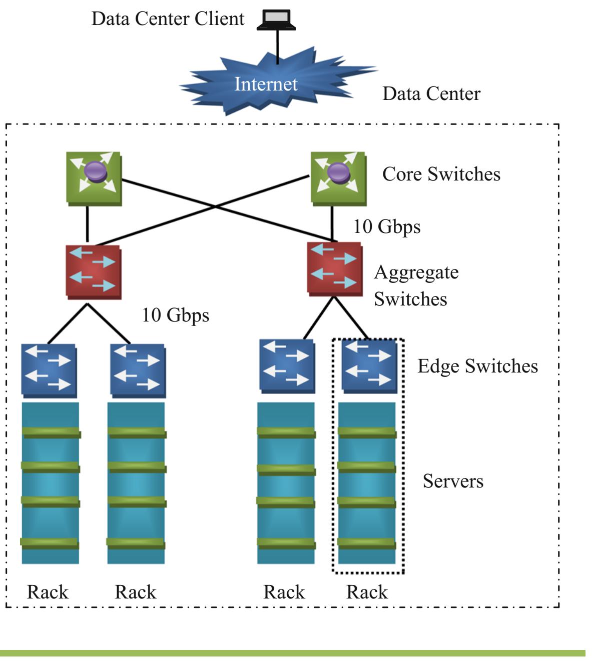

Fig. 1 Current data center network architecture

and low delay. The traffic in data center networks is quite bursty and unpredictable [10].

When the size of servers is large, traffic congestion can occur which results in the sudden loss of packets. In order to design efficient networking mechanisms for data centers, the analysis of the characteristics of data center traffic is important. Typically, there are three classifications for data centers: cloud-computing, private enterprise and university campus. In all data centers, packet size is the common traffic characteristic, while an application and traffic flow vary according to the classification of data centers. In [10], the studies show that the main characteristics of data center traffic are: applications, locality of the traffic flow, the duration and size of traffic flow, concurrent traffic flows and link utilization. The details about the characteristics of data center traffic can be found in [10] for interested readers.

Recent research studies [5,11,12] have shown that majority of the network traffic in data centers are contributed by the transport protocols. Although there are two transport protocols, TCP and UDP, in data center networks, most of the traffic flows over the data center are TCP-based. According to the special features of data center networks, the transport protocol needs some desirable properties such as fast convergence and rare packet losses [13]. However, the conventional TCP does not meet these properties of data center networks. This is because when TCP deployed in data center networks, TCP faces some important issues such as TCP Incast, Outcast, latency and packet reordering. To resolve these issues, recently, various solutions have been proposed for data center networks.

This article surveys the new transport protocols proposed for mitigating the TCP issues in data center networks. The objective of this article is threefold: to discuss about the issues of TCP in data center networks; to introduce various transport layer solutions and finally to compare and discuss the challenges of existing solutions proposed for improving the performance of TCP in data center networks.

The remainder of the survey is organized as follows. In Sect. 2, we present a detailed description about the issues of TCP in data center networks. In Sect. 3, we discuss the recent TCP variants proposed for data center networks for mitigating the issues of TCP and present the comparison of the surveyed transport protocols in terms of some important metrics. Finally, Sect. 4 concludes our paper.

2 Issues of TCP in data center networks

As we mentioned in the above section, the performance of TCP is not satisfactory in data center networks due to its special features when compared to other IP networks. In this section, we discuss about the important issues as shown in Fig. 2 that TCP faces when deployed in data center networks.

2.1 TCP Incast

TCP Incast is one of the crucial performance issues in data center networks [14,15]. It is a catastrophic throughput collapse that occurs when a large number of servers send data simultaneously to a single receiver with high bandwidth and low round trip time. It 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 [16]. TCP Incast was first found in the distributed storage system Panasas [14]. Figure 3 shows a typical TCP Incast scenario of data center networks. In this many-to-one communication pattern, the client sends barrier-synchronized data requests (i.e., the client will not send data requests until all of the senders completed the current request) using a large logical block size to multiple servers via a switch for increasing the performance and reliability. Each server stores a fragment of data block, which is referred to as Server Request Unit (SRU).

All the servers take approximately the same amount of time for sending the requested data to the client, which results in the overflow of buffers at the bottleneck link, and this leads to large amount of packet losses and timeouts, termed as TCP Incast throughput collapse problem. A nice summary of the preconditions for TCP Incast is stated in [17], where the preconditions are listed as follows:

- High-bandwidth, low-latency networks with small switch buffers

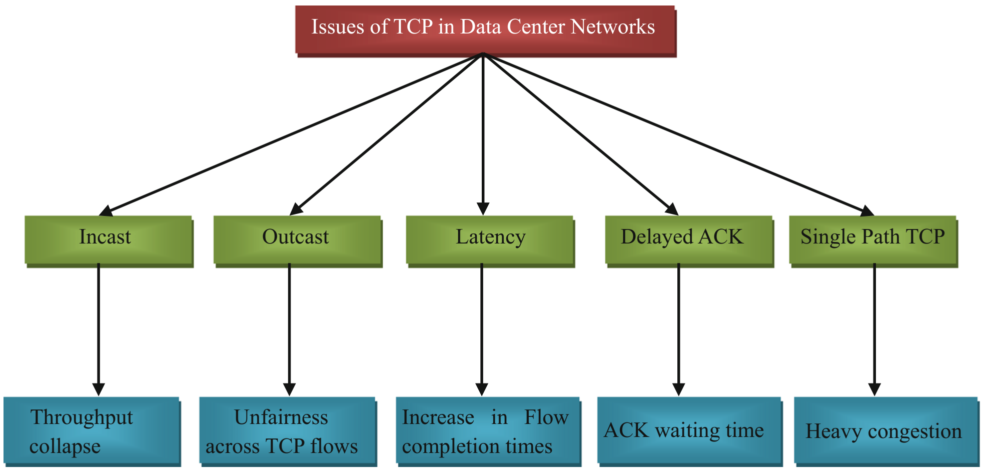
Issues of TCP in Data Center Networks
Incast
Outcast
Latency
Delayed ACK
Single Path TCP
Throughput collapse
Unfairness across TCP flows
Increase in Flow completion times
ACK waiting time
Heavy congestion

Fig. 2 TCP issues in data center networks

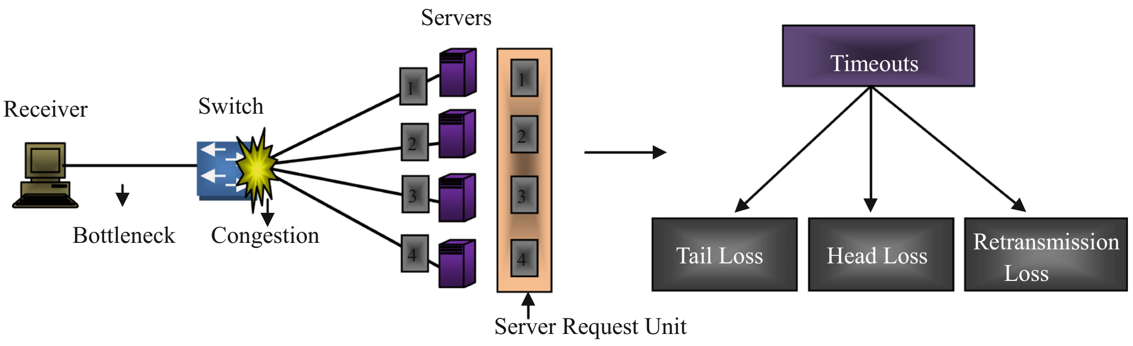

Fig. 3 TCP Incast and causes of timeouts

- Clients that issue barrier-synchronized request in parallel
- Servers that return a relatively small amount of data per request

2.1.1 TCP timeouts

By experimental studies, researchers have found that TCP timeouts are the root cause of Incast problem in data center networks [18–20]. TCP timeouts occur due to fast data transmissions from multiple servers which easily overfills the switch buffers resulting in heavy packet losses. These timeouts impose delays of hundreds of milliseconds in networks with round trip times in the 10 or 100 s of microseconds can reduce the throughput by 90 %. In addition, the frequent timeouts can harm the performance of latency-sensitive data center applications [17]. As shown in Fig. 3, in data center networks, the timeouts are mainly caused by the loss of packets from the tail of data blocks (LTTO), from the head of data blocks (LHTO) and the loss of retransmitted packets (LRTO).

LTTO These timeouts are caused due to the lack of insufficient duplicate acknowledgments. Figure 4a presents an example of timeouts that occur due to packet losses from the tail of data block. Consider that the sender sends 5 packets using the congestion window size of 5. Among those, the packets 3 and 4 are lost. The sender will not receive three duplicate acknowledgments (DACK) for triggering fast retransmission for recovering the lost packets 3 and 4 since only one packet left from the end of the life of TCP flow. As a result, the sender needs to wait for the expiration of timeouts. Such situation is not unavoidable in data center networks.

LHTO If all the packets from a data block are dropped, the receiver cannot send any acknowledgments for the transmitted data as shown in Fig. 4b. As a result, the TCP sender can detect a packet loss after the long idle period of the expiration of retransmission timer. Usually, in data center networks these types of timeouts happen due to the large summation and big variance of the initial size of congestion window when the number of senders becomes large [20]. That is, some senders finish their transmissions earlier than others and needs to wait for other senders to finish their transmissions. As a result, the senders without finishing data transmission have large send window size. On the next data block transmissions, all the senders inject packets using their whole

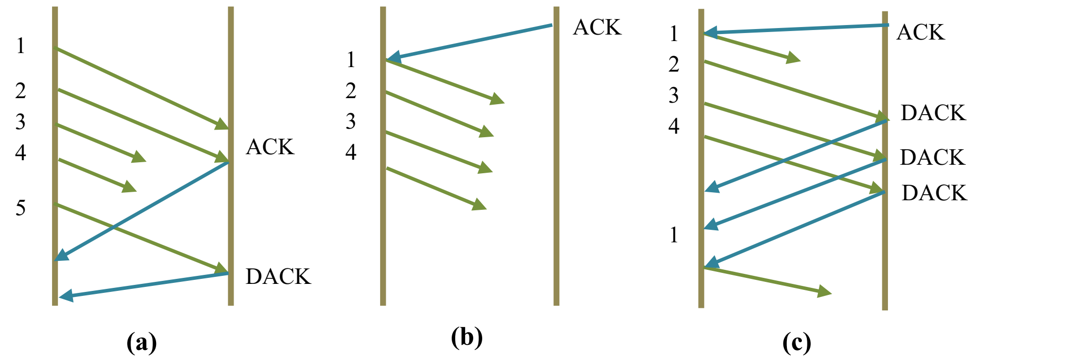

Fig. 4 Different TCP timeouts **a** LTTO, **b** LHTO, **c** LRTO

windows to the small switch buffer, which usually causes lots of packet loss. If a flow losses its whole window, then it will enter a timeout period [18].

LRTO Another important type of timeout is timeouts due to the loss of retransmitted packets. When the TCP sender detects a packet loss via TCP loss detection mechanism, it retransmits the lost packet immediately. However, if the retransmitted packet is lost again as shown in Fig. 4c, the sender needs to wait until the expiration of retransmission timer due to insufficient duplicate acknowledgments. The loss of retransmissions is not rare in data center networks [19]. As a result for improving the performance of TCP, the senders need a mechanism for detecting the loss retransmitted packets.

2.2 TCP Outcast

TCP Outcast is another important issue for degrading the performance of TCP in data center networks. When a large and small set of flows share the same bottleneck link with multi-rooted tree topology, the throughput of TCPs with small set of flows obtain lower throughput than TCPs with large set of flows, which will lead to severe unfairness. This phenomenon has been termed as TCP Outcast and is first described in [21]. One of the main reasons of the TCP Outcast problem is ‘Port blackout.’ That is, a series of packets enter into the same switch from different input ports and compete for the only output port. In this case, some of the packets get dropped when the queue in the output port becomes full. Figure 5 presents the problem of TCP Outcast due to port black out problem.

Consider three packets say PA1, PA2 and PA3 arriving at port A and PB1, PB2 and PB3 arriving at port B. These asynchronously arrived packets from the two input ports A and B are competing for the output port C which is full due to packets PC1, PC2 and PC3. Due to the asynchronous nature, one port may have packets slightly ahead of others. As a result, when a packet dequeued from the output port C, the first arrived packet enters into the occupied space of port C and the next incoming packet gets dropped. In this example, the packet PB1 is enqueued, and the packet PA1 is dropped. This result in a sequence of packets gets dropped from one input port, and thus, that input port suffers a blackout. As a result, the input port suffers from blackout and degrades the throughput. In addition to port blackout problem, in [22] the authors found another reason for the TCP Outcast problem which is mainly caused by the variations in RTT of different TCP flows. That is, the flows with large RTT are less affected than the flows with small RTT.

2.3 Latency

Here, we explain about the latency issue inside a data center network. The main culprit of latency in data center networks is the long queuing delay in switches which is caused by the TCP flow characteristics of data centers [23]. Specifically, data center networks carry two types of TCP flows: short lived and long lived, with sizes that typically ranging from 2 KB to 100 MB [24]. Among this, short-lived flows are latency sensitive, while long-lived flows are latency insensitive that can transfer bulky traffic which causes to grow the bottleneck queue until the packets get dropped. As a result, when long and short flows share the same bottleneck queue, the short flows experience increased latency due to queue build up by long flows [25]. The result is that large number of packet drops and frequent retransmissions.

As a consequence of frequent retransmissions due to packet drops, TCP sender needs to reduce the size of congestion window and needs more RTTs to complete the flow. In addition to confirm packet drops, the retransmission timer needs to be larger than the largest possible RTT, which is too long for a TCP flow to meet its deadline [23]. Furthermore, in data center networks the majority of the traffic is bursty, and hence, packets of short-lived TCP flows get dropped frequently.

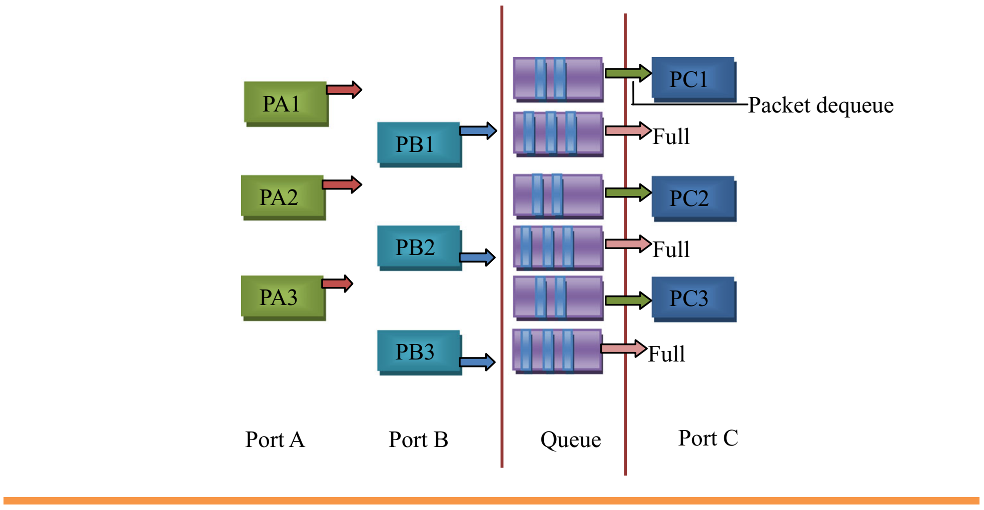

Fig. 5 TCP Outcast due to port blackout

2.4 Delayed ACK

The delayed ACK mechanism of TCP attempts to reduce the amount of ACK traffic in the reverse path of the network. However, in data center networks the delayed ACK mechanism causes frequent timeouts due to RTTs which is in the order of 10 or 100 s of microseconds. This leads to reduction in the throughput. For example, Fig. 6a shows the communication of sender and receiver with delayed ACK disabled. In this case, the receiver sends ACK for each packet and can recover the lost packet by triggering fast retransmissions. In this example, the sender sends two packets P1 and P2. Among those, the packet P2 was lost. When the receiver receives the packet P1, it sends an ACK for packet P2 immediately. As a result, the sender can send two more packets P3 and P4. When the receiver receives those packets, it sends an ACK for getting the lost packet P2. When the sender receives three dupacks, it triggers fast retransmissions and retransmits the lost packet immediately. However, in the case of TCP with delayed ACK enabled, the sender needs to wait more than 40 ms for recovering the lost packet as shown in Fig. 6b and this will affect the performance of TCP in data center networks [17,26,27] in terms of spurious retransmissions and timeouts.

2.5 Single-path TCP

Single-path TCP is another important issue for achieving higher utilization of available bandwidth in data center networks. In data center networks, most of the applications use single-path TCP. As a result, the path gets congested easily and cannot utilize the available bandwidth fully, which results in the degradation of throughput and severe unfairness. For overcoming this problem, recently multi-path TCP (MPTCP) was proposed in [27]. The advantage of this approach is that the linked congestion controller dynamics in each MPTCP end system can act on very short time scales to move traffic away from the more congested paths and place it on the less congested paths. As a result, the packet loss rates can be minimized and thereby improve the throughput and fairness of TCP in data center networks.

3 Existing transport protocols for data center networks

This section surveys the existing transport protocols for data center networks. For our survey, we selected a total of 25 transport protocols proposed for solving the issues of TCP in data center networks. We classified these protocols with respect to different TCP issues such as TCP Incast, Outcast and latency as shown in Fig. 7. Among those, 16 transport protocols are able to mitigate the problem of TCP Incast, 3 transport protocols are able to solve the problem of TCP Outcast, and 10 transport protocols are able to solve the problem of TCP flow completion times. In the following subsections, we will present each of the transport protocols based on the classification of different TCP issues by highlighting their advantages and disadvantages. First, we discuss the protocols proposed for mitigating the TCP Incast problem in data center networks.

3.1 Protocols for solving TCP Incast issue

As we mentioned in Sect. 2, TCP Incast is the degradation in throughput due to the simultaneous transmission of TCP data streams from thousands of worker nodes to one aggregator. As shown in Fig. 7, so far out of 25 protocols, 16 transport

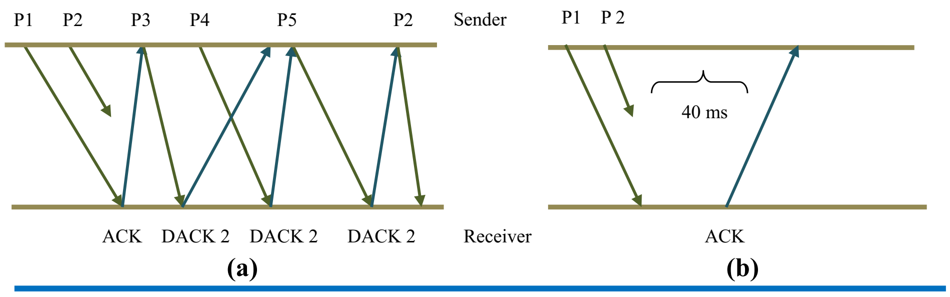

Fig. 6 **a** Delayed ACK disabled, **b** delayed ACK enabled

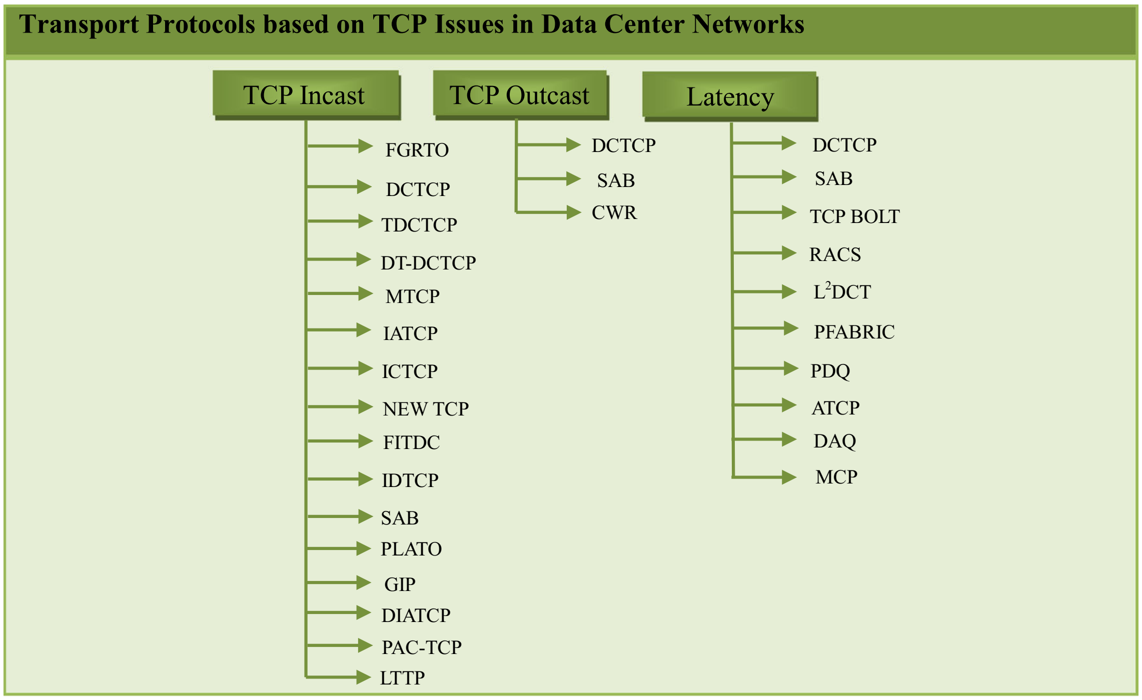

Fig. 7 Transport protocols for data center networks

protocols are proposed for solving TCP Incast issue in data center networks. We classified these protocols in terms of their modifications with respect to conventional TCP and mechanisms for handling different types of TCP timeouts as they are the root causes of TCP Incast.

3.1.1 FGRTO

Generally, in the Internet, the default value of TCP minimum retransmission timeout (RTO) is 200 ms, while the propagation round trip delay in a data center network is less than 1 ms. As a result, in data center networks, when the client requests data from multiple servers, any flow that experiences an RTO will be delayed by 200 ms, resulting in poor query latency. This imbalance between the minimum TCP RTO and the latencies in the data center networks results in the significant degradation of the performance of TCP. In [17], the authors found that eliminating minimum RTO and enabling microsecond-granularity TCP RTO can successfully avoid TCP Incast collapse in data center applications. In addition, the authors recommended that for achieving full performance delayed acknowledgment (ACK) should be disabled. This is because, in data center environment, for servers using fine-grained RTOs, the retransmission timer may expire long in the presence of delayed ACK mechanism. As a result, RTO occurs at the server and retransmit the unacknowledged packet unnecessarily without observing the coarse-grained 40 ms delayed ACK configuration. The major advantage of this approach for avoiding TCP Incast collapse is that the solution is practical, safe and effective for use in data center environment. Also, this approach can be easily deployed, since it requires only three changes to Linux TCP source code such as microsecond resolution time for tracking RTT, redefinition of TCP constants and the replacement of low-resolution timers with high-resolution timers to support microsecond RTOs. Although the solution requires only three changes to the Linux TCP source code, some operating systems not supported the required kernel changes for implementing the software timers. In addition, when the servers communicate with clients outside the data center network, the fine-grained RTO value can be harmful.

3.1.2 DCTCP

Data center TCP (DCTCP) [28] is a TCP-like protocol designed to operate with very low buffer occupancies, without the loss of throughput for data center networks [29]. 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 algorithm consists of three components: First, DCTCP employs a very simple active queue management scheme which ensures that sources quickly notified the overshoot of buffer queue. The simple marking scheme of DCTCP marks the arriving packets with the Congestion Experienced (CE) code point as soon as the queue occupancy exceeds a fixed small threshold 'K.' Second, DCTCP conveys the exact sequence of marked packets back to the sender by sending ACK for every packet, setting the ECN-echo flag if and only if the packet has a marked CE code point. Third, DCTCP controls the sender for reducing the size of congestion window according to the fraction of marked packets based on the congestion status in the network. The minor modifications of the original design of ECN help the DCTCP senders to react early to congestion, which leads to improve the throughput performance than conventional TCP, while using 90 % less buffer space. In DCTCP, the usage of very small buffer space affects the size of congestion window. As a result, the congestion window reduces to one maximum segment size frequently due to the expiration of timeouts. The experimental result of DCTCP shows that when the number of flows is relatively large, DCTCP cannot deal with the TCP Incast problem and the throughput falls back to that of conventional TCP. Moreover, DCTCP requires the modification of ECN settings at the switch as well as changes in the end servers. It is not clear that how DCTCP tunes the ECN threshold for estimating the instant queue length to achieve the high performance of TCP using very low buffer occupancies.

3.1.3 TDCTCP

An improved version of DCTCP called TDCTCP [30] was developed, 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 the dynamic delayed ACK calculation of TCP retransmission timer. 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. This modification helps the sender to react better to the current congestion state and to provide better throughput. Moreover, TDCTCP resets the congestion indicator to the value of 0 after every delayed ACK timeout. This ensures that TDCTCP does not use a stale value of congestion indicator. If the stale value of congestion indicator remains very high, it hinders the increment of congestion window and hence causes successive delayed ACK timeout. In original DCTCP, the old value of congestion indicator gives an incorrect estimation of the current network status. As a result, the congestion window of DCTCP is affected by the old value which results in a high variation in the size of DCTCP's congestion window. This high variation can be prevented with the help of TDCTCP modifications to the congestion indicator. Furthermore, TDCTCP calculates the delayed ACK timeout dynamically to better adapt to the network conditions for achieving good fairness. These three modifications help the TDCTCP to achieve good fairness in data center networks. Although TDCTCP can improve the performance of data centers by mitigating the TCP Incast problem, it needs to modify the sender, receiver and switches in the network. The experiment results show that in data center network with 10 Gbps links, the queue length of TDCTCP is higher than that of DCTCP. This affects the increment in the end-to-end delay of packets. In addition to the network simulation, a testbed experiment also need to be considered for confirming the efficiency of TDCTCP over DCTCP.

3.1.4 DT-DCTCP

Double-threshold DCTCP (DT-DCTCP) [29] improves the DCTCP algorithm by introducing a new marking mechanism for detecting the network congestion. As shown in Fig. 8, DT-DCTCP uses two thresholds 'K1' and 'K2' to share the load of a single threshold 'K' which is used in DCTCP. Among the two thresholds, 'K1' is used to start the ECN marking in advance, and the threshold 'K2' is used to stop the ECN marking. The original intention of double threshold instead of a single threshold is to control the oscillation of queue length which is caused by the nonlinear control scheme at the switches. When the queue length increases beyond the lower threshold 'K1', the network is having a potential congestion, and should set CE to inform the senders to decrease their window size.

That is, in DT-DCTCP the packet will be marked when the queue length increases to 'K1'. On the other hand, when the queue length decreases under the high threshold 'K2', the switch should release the message of congestion. By using the describing function (DF) approach, DT-DCTCP analyzed

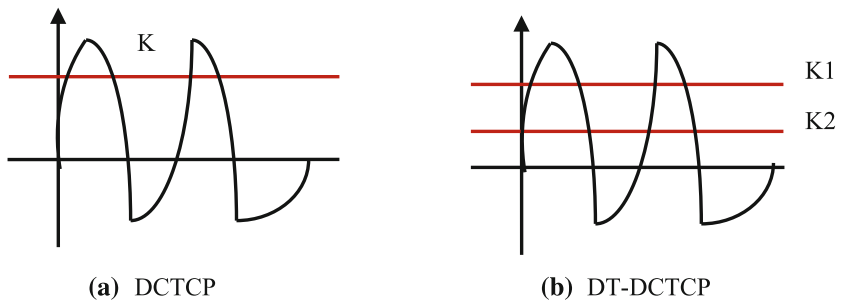

Fig. 8 Thresholds of DCTCP and DT-DCTCP

that it has more stability in queue than DCTCP and thereby solved the issue of TCP Incast in data center networks.

3.1.5 TCP-FITDC

Zhang et al. [31] 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 utilizes the modified packet marking mechanism defined in [28] 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.

3.1.6 DIATCP

A new deadline- and Incast-aware aggregator-based transport protocol (DIATCP) [32], is designed for controlling the peer's sending rate directly to avoid the TCP Incast congestion and to meet the cloud application deadline. The key idea of DIATCP is that the aggregator node may monitor all the incoming traffic to itself. DIATCP works only for the incoming traffic from peers to the aggregator, and the outgoing traffic from the aggregator is managed by the destination nodes of the traffic. In DIATCP, each application connection is represented by an abstract node, which includes the information such as the data size, application deadline and allocated DIATCP window size. Whenever a connection is created, DIATCP inserts a node to the connection list and the node is deleted from the connection list when the connection is closed. After that, the allocated DIATCP window size is updated whenever a change occurs in the connection list. For doing this, DIATCP developed a new global window allocation scheme that allocates the global window based on the deadline and data size. By accessing the information of each node, the advertisement window in the TCP ACK header is set to the allocated window size and can control the total amount of traffic in order not to overflow the bottleneck link. The implementation and the deployment of DIATCP are easy since it does not require any support from the network switches.

3.1.7 IDTCP

Incast decrease TCP (IDTCP) [33] is proposed to mitigate the TCP Incast problem by using the following strategies: (1) constantly monitoring the congestion level of the link, (2) slowing down and dynamically adjusting the congestion window growth rate and (3) setting the congestion window to 1 if the link is totally congested. For monitoring the congestion level of the link, instead of using the queuing packet, IDTCP propose a delicate and effective mechanism for continuously estimating the bottleneck status of the network. For getting the congestion level of the link, IDTCP measures the minimum and average RTT values. According to the congestion level of the link, IDTCP slows down and dynamically adjusts the congestion window growth rate. The congestion window growth rate of IDTCP is a discrete exponential increase with RTT and the base. With the increase in the congestion level, the congestion window growth rate of IDTCP decreases. If the propagation delay is equal to the queuing delay, IDTCP assumes that the link is totally congested and immediately sets the value of congestion window to 1. In this way, IDTCP can allow as many as possible concurrent servers to join into the network.

3.1.8 New TCP

Zheng et al. [37] found that the root cause of TCP Incast is the drop tails induced by TCP congestion control algorithm based on packet losses. By considering the root cause, the authors designed a new delay-based TCP congestion control algorithm based on FAST TCP, for improving the performance of TCP in data center networks. The design of new TCP contains three parts: slow start, congestion avoidance and retransmission mechanism. In the slow-start phase of new TCP, the sender increases its congestion window (W) based on queueing delay and enters the congestion avoidance stage if the queueing delay detected exceeds the slow-start threshold (ssthreshold). Further, the authors suggested a suitable ssthreshold value which is equal to 20 μs. In the congestion avoidance stage of new TCP, the sender periodically updates its congestion window based on queueing delay. Further, for each flow, new TCP maintains a certain queue length at the switch buffer and controls the summation of queue length does not exceed the buffer size. Finally, new TCP adopts the retransmission mechanism of TCP NewReno as it is effective and robust in data centers. By using these three parts, new TCP avoids the drop tails and hence avoids the problem of TCP Incast.

3.1.9 ICTCP

Wu et al. [35] studied the problem of TCP Incast in detail by focusing on the relationship between TCP throughputs, RTT and receive window. The authors observed that the receiver side of TCP is able to know the throughput of all TCP connections and the available bandwidth.

Further, the receiver can control the burstiness of all the synchronized senders by adjusting the receive window size of each TCP connection. In addition, the frequency of receive-window-based congestion control should be made according to the per-flow feedback-loop delay independently. Based upon these observations, the authors proposed an Incast congestion control for TCP (ICTCP) on the receiver side for preventing TCP Incast congestion. The main aim of ICTCP is to reduce the packet loss before Incast congestion instead of recovery after loss. In this regard, to perform congestion control on the receiver side, ICTCP measures the available bandwidth on the network interface and provide a quota for all incoming connections to increase the receiver window for higher throughput. The authors state that for the estimation of throughput live RTTs are necessary as they found that even if the link capacity is not reached, RTT in a high-bandwidth, low-latency network improves with throughput. Similar to the TCP congestion window adjustment at the sender, ICTCP adjusts the receive window based on the ratio of the difference between the measured and expected per-connection throughputs over the expected throughput, as well as the last hop available bandwidth to the receiver. When the ratio is small, ICTCP increases its receive window, but decreases otherwise.

3.1.10 IA-TCP

Hwang et al. [36] proposed an end-to-end congestion control algorithm, named Incast-avoidance TCP (IA-TCP), which is able to control the total number of packets injected into the network pipe to meet the bandwidth-delay product. IA-TCP designed to operate only at the receiver side, which control both the window size of senders and ACK delay. For avoiding the TCP Incast congestion, IA-TCP regulates the sending rate of the senders by using the link capacity measurements obtained from the transport layer instead of measuring live RTT and adjust the window size by using the acknowledgment packet control method. IA-TCP algorithm is performed as follows: First, the receiver counts the total number of TCP connections that share the same bottleneck link. Whenever the receiver sends an ACK, it calculates the advertise window for controlling the total number of packets injected by senders. Then, the receiver induces ACK delay for each packet to prevent the aggregate data packet rate from exceeding the link capacity, and gives uniform random delay for the first ACK. Finally, if the ACK delay is larger than zero, the ACK with advertise would be sent after the expiration of delay timer.

3.1.11 MTCP

Fang et al. [34] proposed a prompt congestion reaction scheme for data center networks called MTCP with consideration of multiple congestion points along data paths. The key goal of MTCP is to achieve congestion control in a transient period of time. For achieving this goal, MTCP reacts to congestion in proportion to the extent of network congestion and removing RTT factor in rate adjustment cycles. For signaling congestion, MTCP utilized ECN field to monitor each congestion point that a packet has passed. When the sender receives a marked ACK, it calculates the congestion status (CS) value according to the number it carries along, accumulates CS value within a certain time interval and adjusts its congestion window with regards to an estimator called α. The calculation of α is as follows:

$$\alpha \leftarrow (1 - g) \times \alpha + g \times R \qquad (1)$$

where g is a constant and R indicates the feedback information of network congestion. Using α, MTCP sender reduces its congestion window according to Eq. (2),

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

where the higher value of α indicates heavy congestion and reduces the size of congestion window significantly, otherwise reduces slightly. In addition to above mechanisms, MTCP designs two more optional mechanism, namely saturation regulation and congestion prediction. In saturation regulation, if all the switches on a path are congested, the senders of such path reduce its congestion window significantly. On the other hand, the mechanism congestion prediction is used to avoid lags between congestion detection and rate reduction.

3.1.12 GIP

Zhang et al. [20] proposed a simple and effective solution called Guarantee Important Packets (GIP) for the applications with the problem of TCP Incast in data center networks. The key idea behind GIP is making TCP aware of the boundaries of the stripe units, and reducing the size of congestion window of each flow at the start of each stripe as well as redundantly transmitting the last packet of each stripe unit. For achieving the goal, GIP modified TCP at the sender side. Since timeout is the root cause of TCP Incast issue, the authors observed that two types of timeouts should be avoided for improving the throughput. First, the timeouts caused by full window losses and second, the timeouts caused by lack of duplicate ACKs. In this regard, GIP proposed two mechanisms for eliminating the two kinds of timeouts. One is reducing the initial congestion window of each sender at the head of the strip unit for avoiding full window loss, and other is redundantly transmitting the last packet of a stripe unit for avoiding the timeouts due to insufficient duplicate ACKs. TCP-GIP modifies only the sender side of the standard TCP; thus, it can be easily implemented in real-time data center environments. The redundant transmission of some of the last three packets for avoiding the lack of ACKs can not only generate more ACKs but also improve the probability that the last three packets successfully reach the receiver before the timeouts occur due to lack of ACKs.

3.1.13 LTTP

Jiang et al. [39] proposed a UDP-based transport protocol, namely LTTP, to tackle the problem of TCP performance degradation due to many-to-one communication pattern of data center networks. LTTP protocol improves Luby transform code to guarantee reliable UDP transmission from servers to the client by using forward error correction method and to restore the original data without requesting for retransmissions. In addition, for congestion control LTTP employs the TCP friendly rate control (TFRC) protocol to adjust the traffic sending rate at servers and to maintain reasonable bandwidth utilization. This congestion control technique ensures that the sender can send data continuously even if the network is congested rather than stopping sending data for a long time.

3.1.14 SAB

Zhang et al. [13] observed that switch buffer sizes are much larger than the bandwidth-delay product. Based on this observation, the authors proposed a new transport protocol called SAB which allocates switch buffer space to the senders for explicitly determining the congestion window of each flow. The authors stated that SAB has two main advantages: First, the SAB flows can converge to their fair bandwidth in one RTT. As a result, the flow completion time of short flows can be reduced significantly. Second, it rarely loses packets. As shown in Fig. 9(a), for each flow, SAB allocates only one packet by reducing the size of maximum segment at the sender. Hence, SAB loses packet rarely. In SAB, whenever the sender sends data packets to the receiver, the switches compute the congestion window for each passing flows. The measured value of congestion window conveyed by the headers of data packets and then returned to the senders by the headers of ACKs. After receiving an ACK, the senders send packets according to the received value of congestion window as shown in Fig. 9(b). Since the switch can accommodate most of the packets, the bottleneck link will not be idle and hence prevents packet losses.

3.1.15 PLATO

Shukla et al. [19] developed a packet labeling scheme called PLATO, as a solution to TCP Incast throughput collapse. PLATO improved the loss detection capabilities of TCP NewReno by leveraging the existing three duplicate ACK mechanisms. This helps the sender to avoid frequent retransmission timeouts and thus tackles the problem of TCP Incast. The main objective of PLATO is to prevent the loss of ACK clock and to avoid the retransmission timeouts resulting from retransmission losses without modifying the TCP congestion control algorithm. To achieve this goal, PLATO places a special label on certain TCP segments. The switch preferentially enqueue these labeled segments. The TCP receiver responds to labeled segments with a corresponding labeled ACK. Whenever the sender received a labeled ACK, it again send a labeled segment and the process repeats once every

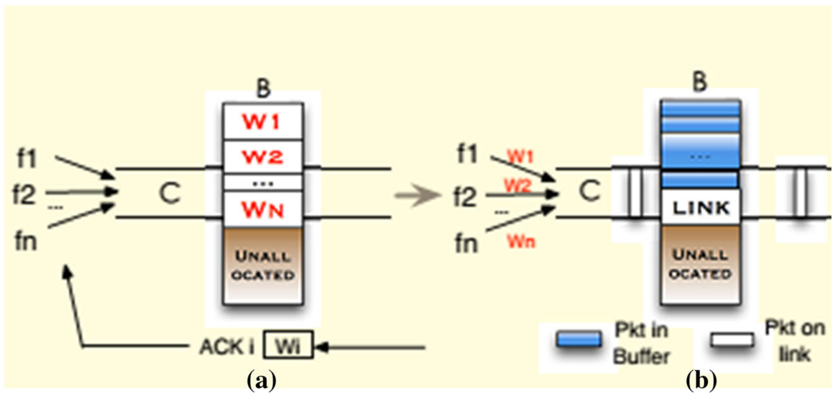

Fig. 9 The key idea of SAB [13]

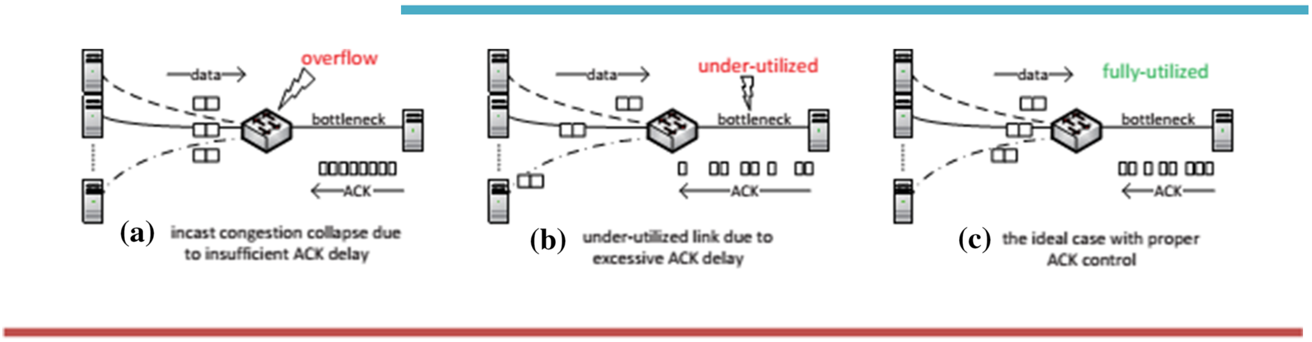
data
overflow
bottleneck
ACK
(a) incast congestion collapse due to insufficient ACK delay
data
under-utilized
bottleneck
ACK
(b) under-utilized link due to excessive ACK delay
data
fully-utilized
bottleneck
ACK
(c) the ideal case with proper ACK control

Fig. 10 The key idea of PAC-TCP [41]

RTT. If some or all unlabeled segments or ACKs are dropped at the switch, the labeled segment and ACKs keep the ACK clock alive. In this way, PLATO save the sender to avoid retransmission timeouts and helps the sender to recover the lost packets using fast retransmissions.

3.1.16 PAC-TCP

Bai et al. [41] proposed a proactive ACK control transport protocol, namely PAC-TCP for taming the problem of TCP Incast by leveraging the characteristics of data center networks. In [41], the authors state that they treat ACK not only as the acknowledgment of the received packets, but also as the trigger for new packets. Based on this, PAC proactively intercepts and releases ACKs in such a way that ACK-triggered in-flight traffic can fully utilize the bottleneck link without causing Incast congestion collapse. The general idea of PAC is presented in Fig. 10. PAC can eliminate the buffer flow as shown in Fig. 10 by imposing higher delay to ACKs. However, it results in the degradation of throughput.

3.2 Comparison of TCP Incast transport protocols

We presented 16 transport protocols proposed recently for mitigating the TCP Incast issue in data center networks. Table 1 summarizes the comparative study of each protocol. Apart from the novelty of the proposed TCP variants, we evaluate each solution using the following four main criteria: modifications to the TCP stack, support from the switch, congestion control algorithm and the retransmission timeouts specifically the mechanism for avoiding three main timeouts, namely LTTO, LHTO and LRTO, as these timeouts are inevitable in data center networks.

Among these protocols, IDTCP, NewTCP and GIP modified only TCP sender side, while DIATCP, ICTCP, IA-TCP and PAC-TCP modified only TCP receiver side. As a result, these protocols are easy to deploy. On the other hand, FGRTO, DCTCP, TDCTCP, DT-DCTCP, TCP-FITDC, MTCP, LTTP, SAB and PLATO needs modification on both the sender and receiver sides. In addition, except FGRTO, DIATCP, IDTCP, NewTCP, ICTCP, IA-TCP and GIP all other protocols needs support from switches for controlling buffer overflow.

However, in data center networks modifications in switches as well as operating systems are little hard in practice, especially in a large scale data center with huge number of servers and switches, which is not truly practical [11,42]. In terms of congestion control algorithms, except TCP-FITDC, NewTCP and IA-TCP, all other transport protocols are window-based congestion control. Being a window-based algorithm, TCP controls the sending rate by adjusting the congestion or receive window when detecting congestion for avoiding the overflow of switch buffer. When severe congestion occurs in the network, these window-based algorithms

Table 1 Comparison of TCP Incast transport protocols

Protocols	TCP modifications	Switch support	Congestion control algorithm	Mechanism for avoiding timeouts		
				LHTO	LTTO	LRTO
FGRTO	Sender and receiver	No	Window-based	No	No	No
DCTCP	Sender and receiver	Yes	Window-based	No	No	No
TDCTCP	Sender and receiver	Yes	Window-based	No	No	No
DT-DCTCP	Sender and receiver	Yes	Window-based	No	No	No
TCP-FITDC	Sender and receiver	Yes	Delay-based	No	No	No
DIATCP	Receiver	No	Window-based	No	No	No
IDTCP	Sender	No	Window-based	No	No	No
NewTCP	Sender	No	Delay-based	Yes	Yes	No
ICTCP	Receiver	No	Window-based	No	No	No
IA-TCP	Receiver	No	Rate-based	No	No	No
MTCP	Sender and receiver	Yes	Window-based	No	No	No
GIP	Sender	No	Window-based	Yes	Yes	No
LTTP	Sender and receiver	Yes	Window-based	No	No	No
SAB	Sender and receiver	Yes	Window-based	No	No	No
PAC-TCP	Receiver	Yes	Window/recovery-based	No	No	No
PLATO	Sender and receiver	Yes	Window-based	Yes	Yes	Yes

can reduce the value of the congestion or receive window size to one MSS. Recent studies show that window-based algorithms can support only limited number of servers around 65 [43–45]. As a result, in the case of large scale data centers, window-based algorithms cannot control the sending rate due to the high level of burstiness of traffic.

Compared with window-based congestion control algorithms, delay-based algorithms can improve throughput as it is a good source to determine the level of network congestion [44,46]. In our survey, the transport protocols TCP-FITDC and NewTCP are delay-based congestion control algorithms. These algorithms estimates queuing delay variation using RTT to signal the network congestion [46]. Based on the queuing delay, the protocols update the size of congestion window which helps to control the sending rate. In addition to window-based and delay-based congestion control algorithms, the transport protocol IA-TCP proposed a rate-based solution for mitigating the TCP Incast problem. To meet the bandwidth-delay product, this protocol controls the total number of packets injected into the network. For controlling the sending rate, IA-TCP measured the link capacity instead of RTT-like delay-based algorithms. However, the recent studies show that measuring bandwidth and RTT is difficult in real data center networks [13,38]. As we mentioned earlier, timeouts are the root cause of TCP Incast problem in data center networks, especially unavoidable timeouts due to head packet loss, tail packet loss and retransmission losses. Therefore, we classified the Incast transport protocols in terms of mechanism for avoiding the three types of timeouts. Out of 16 protocols, all protocols are able to reduce timeouts compared with conventional TCP. However, only three protocols are specifically designed for avoiding three important timeouts for tackling the problem of TCP Incast. They are NewTCP, GIP and PLATO. In case of implementation, compared with PLATO, NewTCP and GIP are easy to deploy. This is because these two protocols need only sender side modifications without any support from switches. However, they do not have mechanisms for avoiding the timeouts due to retransmission losses. In summary, it is essential to develop an easy deployable solution for tackling the problem of TCP Incast by considering the three types of timeouts.

3.3 Protocols for solving TCP Outcast issue

In this section, we present the transport protocols proposed for mitigating the problem of TCP Outcast in data center networks. In [21], the authors stated that in data center networks, the root cause of the TCP Outcast problem is input port blackout at bottleneck switches which is occurring due to the tail drop policy of the output queue. This reduces the throughput of few flows that share the blackout input port drastically. In our up-to-date literature review, only one transport protocol specifically designed to solve the TCP Outcast problem.

3.3.1 TCP-CWR

In [22], Qin et al. stated that the uneven distribution of flows with different RTTs on the physical links and the characteristics of the applications which run on the data center networks are the substantial causes of TCP Outcast problem. Based on their observations, a window size notification-based algorithm, called TCP congestion window replacement (TCP-CWR) was proposed for solving TCP Outcast problem in data center networks. The key idea of TCP-CWR is to adjust the congestion window size based on the information received from the receiver. That is, upon finishing the transmission of the current data block, the applications running on the senders send their current congestion window size to the receiver. When the receiver receives all the data including the window size, it calculates the average window size and sends back to each sender. The senders adjust their congestion window size based on the value received from the receiver. One of the main advantages of this idea is all the senders have the same congestion windows when a new data block starts to end. In addition to TCP-CWR, two Incast protocols, namely DCTCP and SAB, are also able to solve the Outcast problem.

3.4 Protocols for solving latency issue

As we mentioned in Sect. 2, the data traffic of data center networks is a mix of long and short TCP flows. When long and short flows share the same queue, the short flows suffer from increased latency due to queue build up by long flows. In this subsection, we survey the proposed transport protocols specifically designed for solving the flow completion time (latency) issue in data center networks.

3.4.1 TCP BOLT

In [40], Brent Stephens et al. proposed a transport protocol called TCP BOLT, designed to achieve shorter flow completion time in data centers while avoiding head-of-line blocking and maintaining near TCP fairness. Furthermore, to prevent deadlock, TCP BOLT presented a novel routing algorithm that uses edge-disjoint spanning trees. In [40], for reducing the TCP flow completion times, TCP BOLT utilizes data center bridging (DCB) to eliminate TCP slow start, DCTCP to maintain low queue occupancy and bandwidth-delay product sized congestion windows. For ensuring the prioritization of latency-sensitive flows, TCP BOLT used the DCB feature of Priority Flow Control.

3.4.2 RACS

Munir et al. [43] proposed a Router Assisted Capacity Sharing (RACS) transport protocol for achieving low latency in data centers. Behind the design of RACS, there are three main goals: (1) minimize flow completion times, while maintaining high network throughput by approximating the shortest remaining processing time (SRPT) scheduling policy in a distributed manner, (2) achieve high utilization and (3) achieve high burst tolerance. RACS achieved these goals by using a weighted capacity sharing mechanism, explicit rate feedback and rate-based transfer. With RACS, each flow maintains a weight, which determines their relative priority in rate allocation decisions and communicates it to the routers by rate request packets that are piggybacked onto one of the data packets every RTT. With SRPT, a flow with shortest remaining processing time gets the highest priority. Routers periodically compute the sum of weights of all flows traversing them and assign a rate to each flow that is proportional to the ratio of flow's weight to the sum. The minimum rate along the path of a flow is communicated to the receiver, which conveys the allocated rate to the senders by copying it onto their acknowledgment packets or ACKs.

3.4.3 L^2DCT

In [25], Munir et al. proposed another low-latency data center transport protocol called L^2DCT, for minimizing flow completion times in data centers. L^2DCT focused on reducing the completion time for short flows by approximating least attained service (LAS) scheduling discipline, in a distributed way. With L^2DCT, each flow assigned a weight based on the amount of data sent to the receiver for estimating the flow size. Moreover, for getting the congestion status of the network, the sender used explicit congestion notification marking scheme. Using this scheme, the sender measures the extent of network congestion by maintaining a weighted average of the fraction of marked packets α. Using the estimated flow size as well as the value of alpha, the sender modulates the size of congestion window. When the sender updates its congestion window size, short flows are given higher bandwidth than long flows. This size-aware congestion management scheme of L^2DCT can significantly reduce the flow completion time in data centers.

3.4.4 PFABRIC

Alizadeh et al. [44] observed that rate control is a poor and ineffective technique for flow scheduling and that the mechanisms for the two should be decoupled and designed independently. Based on this observation, the authors proposed a minimalistic data center transport protocol called PFABRIC, to provide near-optimal flow completion times even at the 99th percentile for short flows. In PFABRIC, for each packet, the end hosts attach a single number in the header that encodes its priority. Whenever the switch receives a new packet, it decides which packets to accept into the buffer and which ones to schedule strictly according to the packet's priority number. When a new packet arrives and the buffer is full, then the switch checks whether the priority of the incoming packet is lower than the packets stored in the buffer. If yes, then the switch drops that packet. On the other hand, if the packet in the buffer has lower priority than the incoming one, then switch dropped the stored lowest priority packet and replaced with the incoming packet. When transmitting, the switch sends the packet with the highest priority. In this way, the switches can operate independently in a greedy and local fashion using priority-based scheduling and dropping mechanism. Furthermore, the rate control design of PFABRIC used only timeouts mechanism for recovering the dropped packets. Upon a timeout, the flow enters into the slow-start algorithm and reduces the value of slow-start threshold into half of the congestion window size before the timeout occurred.

3.4.5 PDQ

In Hong et al. [45], proposed a Preemptive Distributed Quick (PDQ) flow scheduling transport protocol to complete flow quickly and meet flow deadlines. As a centralized scheduling mechanism, PDQ follows the scheduling principles of EDF to minimize mean flow completion time, and SJF to minimize the number of deadline-missing flows. In PDQ, whenever the sender departs a packet, it attaches a scheduling header to the packet including some state variables about its expected transmission time, the flow's deadline, its sending rate and round trip time. Upon receiving the packets at switches, PDQ checks the flows having the earliest deadline and its size. If more than one flows with the earliest deadline, PDQ transmit the flow with smaller size. On the other hand, if there is only one flow with the earliest deadline, then PDQ transmits that flow to the receiver. On the arrival of packets at the receiver side, the receiver copies the scheduling header from each packet to its corresponding ACK. Whenever an ACK packet arrives, the sender updates its flow sending rate based on the feedback. In this way, PDQ reduces the flow completion time and improves the missed deadline.

3.4.6 ATCP

Wu et al. [46] observed that the delay-sensitive applications typically use short flows and often coexist with large flows. This observation motivates the authors to propose an adaptive transmission control protocol (ATCP), to reduce the flow completion time of short flows. ATCP was designed to steal bandwidth from large flows over time and reallocate to small ones, and to compensate large flows by more transfer time. To achieve this, ATCP modified the additive increase behavior of TCP congestion control and perform adaptive weighted fairness sharing among flows. By setting higher weight to small flows, ATCP allocates more bandwidth to small flows than large flows, and thus, ATCP can reduce the completion time of short flows.

3.4.7 DAQ

Ding et al. [47] proposed a deadline-aware queue transport protocol to achieve high throughput for long background flows, while providing high application throughput of short flows. The objective of DAQ scheduling scheme is to maximize the number of flows completing transmission before their deadlines and to guarantee that long flows receive the required bandwidth. In [47], it is stated that there are four main reasons why TCP prolongs the flow completion time: (1) use of slow-start mechanism, (2) memory monopolizing by long TCP flows, (3) building up of queue at the switch and (4) use of timeouts and retransmissions for error control and congestion avoidance. For addressing the drawbacks of (2) and (3), DAQ used two different queues at supporting switches, one per flow type, for avoiding memory monopolizing by long flows and to enable reaching a high transmission rate. Furthermore, the queue for short flows is divided into urgent and not-urgent short flows to increase the number of short flows that finish service within their deadlines while keeping minimum state information. Urgent flows are served with higher priority than not-urgent flows. Weighted round-robin scheduling is used for determination of service to the different flows. By using a deadline threshold, DAQ determined the urgency of a flow. In addition, DAQ used a simple but effective link-by-link credit-based flow control for avoiding packet loses and retransmissions and to overcome the drawbacks of (1) and (4).

3.4.8 MCP

Chen et al. [49] proposed a novel distributed and reactive transport protocol called MCP, for reducing the per-packet delay while providing right transmission rates to meet deadlines by fully utilizing the ECN feedback mechanism. To minimize the overall per-packet delay, MCP aims to devise an optimal source rate control mechanism in data center network with throughput guarantee. At first, the authors formulate a stochastic packet delay minimization problem with constraints on deadline completion and network stability and then derive an optimal congestion window update function to determine the right transmission rate for every flow.

3.5 Comparison of transport protocols proposed for reducing TCP flow completion time

We presented ten transport protocols including DCTCP and SAB proposed recently for mitigating the TCP flow completion time issue in data center networks. Table 2 summarizes the comparative study of each protocol. Apart from the novelty of the proposed TCP variants, we evaluate each solution using the following four main criteria: modifications to the TCP stack, modification to the switch

Table 2 Comparison of transport protocols which is able to reduce the TCP flow completion time

Protocol	Deadline aware	Modification in switch hardware	ECN feedback	Require flow size information	TCP modifications
DAQ	Yes	Yes	No	No	Sender and receiver
MCP	Yes	No	Yes	Yes	Sender and receiver
TCP BOLT	No	No	Yes	No	Sender
PDQ	Yes	Yes	No	Yes	Sender and receiver
PFABRIC	Yes	No	No	Yes	Sender and receiver
RACS	No	No	No	No	Sender and receiver
L^2DCT	No	No	Yes	No	Sender
ATCP	Yes	No	No	Yes	Sender and receiver
DCTCP	No	No	Yes	No	Sender and receiver
SAB	No	No	No	Yes	Sender and receiver

hardware, ECN feedback, require flow size information, and to check whether the proposed protocols is deadline aware or not for improving the performance in data center networks. Among this transport protocols, DAQ, MCP, PDQ, PFABRIC and ATCP are deadline-aware protocols proposed for solving the flow completion time. The main objective of these protocols is to assign deadlines to flows and try to meet those deadlines. Compared with deadline-unaware protocols such as TCP BOLT, RACS, L^2DCT, DCTCP and SAB, reducing the flow completion time while determining deadlines for providing guaranteed transmission rate is a challenging tasks as it varies with load, network congestion, etc. However, in data center networks, the data traffic is a mix of long and short flows [48,50,51]. Short deadlines can be associated with long flows. As a result, deadline-aware protocols do not necessarily minimize the flow completion time. Currently, the deadlines are set based on the user experience surveys, since there is no established basis for accurately setting the deadlines.

In addition, the deadline-aware protocols such as ADQ and PDQ require nontrivial switch hardware modification and are quite challenging in the practical difficulties for implementing these protocols, not only that the deadline-aware protocols except DAQ required flow size information for prioritizing the short and long flows. Passing this information to the centralized schedulers causes overhead in the network, which may degrade the performance. Other protocol assigns weights to flows instead of getting the flow size information. Furthermore, the protocols MCP, TCP BOLT, L^2DCT and DCTCP require ECN mechanism for sending congestion information to the senders. As a result, in addition to the modifications of TCP stack, these protocols need switch support for reducing the flow completion time. By considering the limitations of these protocols, it is necessary to propose a protocol for reducing the flow completion time.

4 Conclusion

Data centers have become a cost-effective infrastructure for hosting a diverse range of cloud applications and for storing large volumes of data. The network applications of the data center networks are diverse in nature and have various performance requirements. Data center flows typically operate using the TCP as it is a mature technology that provides reliable and ordered bidirectional delivery of stream of bytes from one application to the other application residing on the same or two different machines. However, when TCP is deployed in data center networks, it is unable to provide high throughput and fairness mainly due to the problem of TCP Incast, Outcast and latency. Hence, there is a need to redesign TCP specifically to handle the traffic in data center networks. In this paper, we presented an in-depth survey of recently proposed transport protocols specifically to mitigate the TCP issues in data center networks by highlighting the advantages and limitations of the prominent transport protocols. In addition, we have provided a comparative study of the currently existing transport protocols in detail to further assist readers to understand the specific modifications required for the implementation of each protocols.

Acknowledgments The authors gratefully acknowledge the anonymous reviewers for their valuable comments and suggestions for improving the manuscript. This research was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. 2011-0023177). This research was also supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the "Establishing IT Research Infrastructure Projects" supervised by the NIPA (National IT Industry Promotion Agency) (I2221-14-1005).

References

[1] Chen, K., Chengchen, H., Zhang, X., Zheng, K., Chen, Y., Vasilakos, A.V.: Survey on routing in data centers: insights and future directions. IEEE Netw. **25**(4), 6–10 (2011)
[2] Hua, Y., Xue, L., Jiang, H.: ANTELOPE: a semantic-aware data cube scheme for cloud data center networks. IEEE Trans. Comput. **63**(9), 2146–2159 (2014)
[3] Kato, M.: Improving Transmission Performance with One-Sided Datacenter TCP. M.S. Thesis, Keio University (2014). http://eggert.org/students/kato-thesis.pdf
[4] Wang, L., Zhang, F., Vasilakos, A.V., Hou, C., Liu, Z.: Joint virtual machine assignment and traffic engineering for green data center networks. SIGMETRICS Perform. Eval. Rev. **41**(3), 107–112 (2014)
[5] 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)
[6] Yu, Y.-J., Chuang, C.-C., Lin, H.-P., Pang, A.-C.: Efficient multicast delivery for wireless data center networks. In: 2013 IEEE 38th Conference on Local Computer Networks (LCN), pp. 228–235, 21–24 Oct 2013
[7] Li, D., Wu, J.: On the design and analysis of data center network architectures for interconnecting dual-port servers. In: INFOCOM, 2014 Proceedings IEEE, pp. 1851–1859, 27 April–2 May 2014
[8] Ilyadis, N.: The evolution of next-generation data center networks for high capacity computing. In: 2012 Symposium on VLSI Circuits (VLSIC), pp. 1–5, 13–15 June 2012
[9] Kachris, C., Tomkos, I.: A survey on optical interconnects for data centers. IEEE Commun. Surv. Tutor. **14**(4), 1021–1036 (2012)
[10] Li, D., Mingwei, X., Liu, Y., Xie, X., Cui, Y., Wang, J., Chen, G.: Reliable multicast in data center networks. IEEE Trans. Comput. **63**(8), 2011–2024 (2014)
[11] Xu, H., Li, B.: RepFlow: minimizing flow completion times with replicated flows in data centers. In: INFOCOM, 2014 Proceedings IEEE, pp. 1581–1589, 27 April–2 May 2014
[12] Zhang, J., Ren, F., Lin, C.: Modeling and understanding TCP incast in data center networks. In: INFOCOM, 2011 Proceedings IEEE, pp. 1377–1385, 10–15 April 2011
[13] 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)
[14] Nagle, D., Serenyi, D., Matthews, A.: The panasas activescale storage cluster: delivering scalable high bandwidth storage. In: Proceedings of the ACM/IEEE Conference on Supercomputing, pp. 53–62 (2004)
[15] 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). USENIX Association, Berkeley, CA, USA, Article 12 (2008)
[16] 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, NY, USA (2009)
[17] 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, NY, USA (2009)
[18] Zhang, J., Ren, F., Tang, L., Lin, C.: Modeling and solving TCP Incast problem in data center networks. IEEE Trans. Parallel Distrib. Syst. **26**(2), 478–491 (2015)
[19] Shukla, S., Chan, S., Tam, A.S.-W., Gupta, A., Xu, Y., Chao, H.J.: TCP PLATO: packet labelling to alleviate time-out. IEEE J. Sel. Areas Commun. **32**(1), 65–76 (2014)
[20] Zhang, J., Ren, F., Tang, L., Lin, C.: Taming TCP incast throughput collapse in data center networks. In: 2013 21st IEEE International Conference on Network Protocols (ICNP), pp. 1–10, 7–10 Oct 2013
[21] Prakash, P., Dixit, A., Hu, Y.C., Kompella, R.: The TCP outcast problem: exposing unfairness in data center networks. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI'12), pp. 30–30. USENIX Association, Berkeley, CA, USA (2012)
[22] Qin, Y., Shi, Y., Sun, Q., Zhao, L.: Analysis for unfairness of TCP outcast problem in data center networks. In: 2013 25th International on Teletraffic Congress (ITC), pp. 1–4, 10–12 Sept 2013
[23] http://www.cs.cityu.edu.hk/~hxu/dcn.html
[24] Zats, D., Das, T., Mohan, P., Borthakur, D., Katz, R.: DeTail: reducing the flow completion time tail in datacenter networks. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '12), pp. 139–150. ACM, New York, NY, USA (2012)
[25] Munir, A., Qazi, I.A., Uzmi, Z.A., Mushtaq, A., Ismail, S.N., Iqbal, M.S., Khan, B.: Minimizing flow completion times in data centers. In: INFOCOM, 2013 Proceedings IEEE, pp. 2157–2165, 14–19 April 2013
[26] https://tools.ietf.org/html/rfc1122
[27] Ming, L., Lukyanenko, A., Tarkoma, S., Yla-Jaaski, A.: MPTCP Incast in data center networks. Commun. China **11**(4), 25–37 (2014)
[28] 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)
[29] Chen, W., Cheng, P., Ren, F., Shu, R., Lin, C.: Ease the queue oscillation: analysis and enhancement of DCTCP. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS), pp. 450–459, 8–11 July 2013
[30] Das, T., Sivalingam, K.M.: TCP improvements for data center networks In: 2013 Fifth International Conference on Communication Systems and Networks (COMSNETS), pp. 1–10, 7–10 Jan 2013
[31] Zhang, J., Wen, J., Wang, J., Zhao, W.: TCP-FITDC: an adaptive approach to TCP Incast avoidance for data center applications. In: 2013 International Conference on Computing, Networking and Communications (ICNC), pp. 1048–1052, 28–31 Jan 2013
[32] Hwang, J., Yoo, J., Choi, N.: Deadline and Incast aware TCP for cloud data center networks. Comput. Netw. **68**, 20–34 (2014)
[33] 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)
[34] Fang, S., Foh, C.H., Aung, K.M.M.: Prompt congestion reaction scheme for data center network using multiple congestion points. In: 2012 IEEE International Conference on Communications (ICC), pp. 2679–2683, 10–15 June 2012
[35] Haitao, W., Feng, Z., Guo, C., Zhang, Y.: ICTCP: Incast congestion control for TCP in data-center networks. IEEE/ACM Trans. Netw. **21**(2), 345–358 (2013)
[36] 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, 10–15 June 2012

[37] Zheng, F., Huang, Y., Sun, D.: Designing a new TCP based on FAST TCP for datacenter. In: 2014 IEEE International Conference on Communications (ICC), pp. 3209–3214, 10–14 June 2014
[38] http://onlinelibrary.wiley.com/doi/10.1002/ett.1485/abstract
[39] Jiang, C., Li, D., Mingwei, X.: 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)
[40] Stephens, B., Cox, A.L., Singla, A., Carter, J., Dixon, C., Felter, W.: Practical DCB for improved data center networks. In: INFOCOM, 2014 Proceedings IEEE, pp. 1824–1832, 27 April–2 May 2014
[41] Bai, W., Chen, K., Wu, H., Lan, W., Zhao, Y.: PAC: taming TCP Incast congestion using proactive ACK control. In: 2014 IEEE 22nd International Conference on Network Protocols (ICNP), pp. 385–396, 21–24 Oct 2014
[42] Lee, C., Jang, K., Moon, S.: Reviving delay-based TCP for data centers. SIGCOMM Comput. Commun. Rev. **42**(4), 111–112 (2012)
[43] Munir, A., Qazi, I.A., Bin Qaisar, S.: On achieving low latency in data centers. In: 2013 IEEE International Conference on Communications (ICC), pp. 3721–3725, 9–13 June 2013
[44] Alizadeh, M., Yang, S., Sharif, M., Katti, S., McKeown, N., Prabhakar, B., Shenker, S.: pFabric: minimal near-optimal datacenter transport. SIGCOMM Comput. Commun. Rev. **43**(4), 435–446 (2013)
[45] Hong, C.-Y., Caesar, M., Godfrey, P.B.: Finishing flows quickly with preemptive scheduling. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '12), pp. 127–138. ACM, New York, NY, USA (2012)
[46] Wu, W., Chen, Y., Durairajan, R., Kim, D., Anand, A., Akella, A.: Adaptive data transmission in the cloud. In: 2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS), pp. 1–10, 3–4 June 2013
[47] Ding, C., Rojas-Cessa, R.: DAQ: deadline-aware queue scheme for scheduling service flows in data centers. In: 2014 IEEE International Conference on Communications (ICC), pp. 2989–2994, 10–14 June 2014
[48] 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, NY, USA (2012)
[49] Chen, L., Hu, S., Chen, K., Wu, H., Tsang, D.H.K.: Towards minimal-delay deadline-driven data center TCP. In: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks (HotNets-XII). ACM, New York, NY, USA, Article 21 (2013)
[50] http://queue.acm.org/detail.cfm?id=2208919
[51] Haitao, W., Jiabo, J., Guohan, L., 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), pp. 25–36. ACM, New York, NY, USA (2012)

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.