IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INTELLIGENT INFORMATION SERVICES

Received December 6, 2019, accepted December 13, 2019, date of publication December 16, 2019,
date of current version December 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2960180

Transmission Control of MPTCP Incast Based on Buffer Balance Factor Allocation in Data Center Networks

SHANCHEN PANG, JIAMIN YAO, XUN WANG, TONG DING, AND LI ZHANG
College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, China

Corresponding author: Xun Wang (wangxun0830@hotmail.com)

This work was supported by the National Natural Science Foundation of China under Grant 61572523, Grant 61502535, Grant 61672248, Grant 61873281, Grant 61572522, Grant 61672033, and Grant 61502535.

ABSTRACT The multi-home host will become increasingly popular within IPv6, especially in data center Fat-tree networks. MPTCP protocol is designed to transmit over multipath load, providing users with transparent multi-path utilization and high data transmission rate. As the characteristics of Many-to-one pattern and a large number of short bursts, all of flows collide on the shared link and same output port, creating an inevitable bottleneck collapse during transmission process. In order to avoid the collapse of throughput and improve the throughput of the receiving end, the MPTCP protocol in the data center network should have the characteristics of minimal packet loss and rapid convergence. In this paper, we focus on throughput collapse avoidance and design a MPTCP Transmission Control method based Queuing Cache Balance Factor called QCBF in Multi-homed FatTree topology. This method can realize the sub-flows are allocated by the size of ToRs cluster cache. We first propose a buffer pool balance factor and build a ToRs cluster cache balance queue. Then according to ToRs cluster cache allocation to calculate a sub-flow congestion window value. QCBF not only avoids the MPTCP Incast problem effectively, but also makes full use of the aggregate bandwidth. Based the software NS3, the experiment results of experiment show the QCBF transmission control method could avoid loss of packet, and achieve the load balance of MPTCP data transmission.

INDEX TERMS
Data center, transmission control, balance factor, buffer cache.

I. INTRODUCTION

The multi-home hosts will become increasingly popular within IPv6 [1], especially in the architecture of data center Fat-tree network and wireless environment, which rely heavily on the multi-path function of seamless switch. TCP can not use these connection resources at the same time, and MPTCP supports the reverse multiplexing of redundant channel resources [2], providing users with transparent multi-path utilization and high data transmission rate, which is gaining momentum.

Data center network traffic characteristics, as the basic basis of transmission control, are characterized by locality, dynamics, and imbalance [3]. Among them, the characteristics of Many-to-one pattern [4] and a large number of short bursts are most prominent. Particularly in the Incast communication mode such as Cluster-based storage system [5] and MapReduce-based application [6], multiple senders transmit data to the same receiver in parallel, and only when all senders complete the current round of data transmission, the next round of data block transmission begins. By the fact that all of flows adopt the window evolution mechanism of Additive Increase Multiplicative Decrease (AIMD) [7] and the buffer at the receiving end needs a certain time to wait for data recombination, all of flows collide on the shared link and same output port, creating an inevitable bottleneck collapse during transmission process, even though MPTCP mitigates the load of bottleneck to some extent through multipath load transmission [8]. So, in Incast communication mode, in order to avoid the collapse of throughput and improve the throughput of the receiving end, the MPTCP protocol should have the following characteristics. First, there is very little packet loss. Ideally, the transmission protocol should never lose packets. As long as there is no packet loss, the buffer on the

The associate editor coordinating the review of this manuscript and approving it for publication was Shouguang Wang.

receiving end completes the data reorganization in a certain time, and the probability of collision between the subflows on the shared link is almost zero. Hence, the bottleneck collapse during transmission is avoided.

Second, fast convergence. Because MPTCP data transmission is asynchronous, it is likely that there will still be substreams that cannot preempt fair bandwidth at the end of the transmission in the data center. Moreover, both TCP and MPTCP have more stable throughput performance when there is enough cache space. The buffer required to achieve stable maximum network throughput in MPTCP is close to 3 times that of TCP cache [9], which consumes a lot of cache space. Actually the buffer pool of the commercial Ethernet switch is usually small. In this communication mode, when multiple flows, concurrently injecting traffic to the bottleneck link is more likely to cause a large number of packets to overflow from the bottleneck buffer pool. Some lost packets cannot be read and retransmitted for recovery in this mode, causing a large number of timeout retransmissions [10]. By default, the timeout retransmission time is a minimum of 200 milliseconds [11], which is much larger than the round-trip propagation delay in the data center network. A large number of timeout retransmissions can cause the link to be idle, which greatly reduces the throughput of the receiver.

Based on the MPTCP Incast problem [12], the easiest solution is to increase the number of server interfaces and distribute the traffic load as much as possible across multiple links and edge switch buffer pools. This physical solution requires higher hardware overhead [13]. As servers are mostly dual-homed hosts today and it is impractical to switch to large data infrastructures for general commercial use. The problem of throughput collapse does not represent the shortage of resources available in the data center. On the contrary, it shows that the average load of the data center is too low and the link resources are not maximized [14]. Therefore, With the continuous application of MPTCP, how to avoid the MPTCP Incast collapse problem and improve the throughput have become a urgent problem faced by researchers. In this paper, to avoid the collapse and improve the throughput of MPTCP Incast, a MPTCP Transmission Control method based Queuing Cache Balance Factor (QCBF) is proposed, which can realize the sub-flows are allocated by the size of ToRs cluster cache in Multi-homed FatTree topology. QCBF adopts a window-based congestion control method to design sender S, the ToR cluster and receiver D respectively. We build a ToRs cluster cache balance queue based on the a buffer pool balance factor and build a ToRs cluster cache balance queue. Then according to ToRs cluster cache allocation to calculate a sub-flow congestion window value. QCBF not only avoids the MPTCP Incast problem effectively, but also makes full use of the aggregate bandwidth.

II. RELATED WORK

In the past a few years, a number of efficient heuristics have been proposed to solve the problem of MPTCP Incast callapse. In particular, ECMP routing is commonly used in DCNs [15]. Static hash allocation may distribute traffic to the same path, resulting in unbalanced traffic in the network and switch resource pools. Further, Zhu *et al.* [16] proposed a link increase algorithm based on bandwidth estimation,which determined the window increase and decrease according to the link bandwidth estimated by the batch filter. Ozcan *et al.* [17] suggested closing the heavily congested TCP connections at the receiving end to avoid redundant retransmission and reduce buffer blocking. Li *et al.* [18] proposed a equal-weighted congestion control mechanism by allowing multiple sub-flows to compete with a single TCP flow fairly stream. Wang *et al.* [19] described the mechanism of the Balanced linked adaptation, which was a congestion control algorithm for MPTCP. This method judiciously balanced this trade off based on a new design framework that allowed one to systematically explore the design space.

Furthermore, Morteza analyzed the obstruction of MPTCP and XMP deployment in DCNs in [20]: TCP Incast and Minimum Window Syndrome. And he proposed an adaptive multipath congestion control mechanism by switching multi-path flows and single-path flows adaptively. J. Ye *et al.* [21] proposed an enhanced MPTCP protocol, which adjusted the time granularity of the congestion detection and control under a different number of subflows and achieved lower latency for small flows and higher throughput for large flows. Kimura and Loureiro [22] proposed a multipath congestion control algorithm adjusts the sending operation of all subflows in a coupled fashion in order to achieve various objectives, such as friendliness, responsiveness, throughput improvement and congestion balance.

The above methods won't reduce the number of packets of lost, even though those have relieved MPTCP Incast collapse. With the development of multi-homed technology, Tariq and Bassiouni [23], [24] applied MPTCP in multi-homed networks for wireless communication to estimate performance. Jereczek proposed replacing the network interface with a multi-homed host Ethernet host in [25], built a software switch network on a commercial off-the-shelf server, and estimated the network cost. More and more researches show that multi-homed technology will become the key technology of DCNs currently. To avoid the MPTCP Incast collapse problem and improve the throughput. This paper proposes a MPTCP Transmission Control method QCBF based on ToRs cluster cache allocation in Multi-homed FatTree topology. A buffer pool balance factor and a ToRs cluster cache balance queue are designed to calculate a sub-flow congestion window value. QCBF not only avoids the MPTCP Incast problem effectively, but also makes full use of the aggregate bandwidth.

III. MATERIALS AND METHODS

A. MPTCP INCAST COLLAPSE PROBLEM

In the DCNs, With the introduction of dual-homed hosts, Dual-Fat-Tree is widely used to realize full connect between

different layers of switch devices [26]. The network is centered on the switch with three-layer of cascaded multi-tree topology. There are a core layer, an aggregation layer, and an edge layer. Dual-homed FatTree topology diagram is showed in Fig 1.

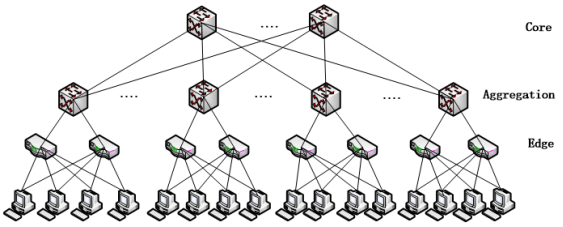

FIGURE 1. Dual-homing FatTree topology.

Based on the application of the MPTCP protocol in the multi-homed fat tree topology DCN, we take a PoD [27] as an example to analyze the MPTCP Incast problem. The topology is shown in Fig 2. Define the data center network topology as an N-host fat tree topology. The number of servers is k, and the number of edge switches ToR is N. We assume that each server is an N-homed host and is connected to the same edge switch ToR at the same time. When K senders transmitted, data to the same receiver D through the ToR switch and $k * n$ paths. K MPTCP connections are established.

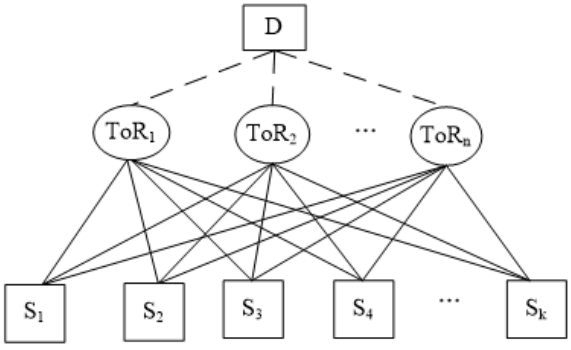

FIGURE 2. PoD of multiple-homing FatTree topology.

According to the attribution of sub-flows, it can be divided into the following two cases: MPTCP sub-flows $F_{1j}\{f_{11}, f_{12}, \ldots, f_{1j}\}$ belonging to the same MPTCP connection transmit data to $ToR\{1, 2, \ldots, n\}$ through links $P1j\{p_{11}, p_{12}, \ldots, p_{1n}\}$ where $j\{1, 2, \ldots, n\}$; 2) MPTCP sub-flows $F_{ij}\{f_{11}, f_{12}, \ldots, f_{ij}\}$ belonging to different MPTCP connections transmit data to $ToR\{1, 2, \ldots, n\}$ through links $P_{ij}\{p_{11}, p_{12}, \ldots, p_{ij}\}$ where $i\{1, 2, \ldots, k\}, j\{1, 2, \ldots, n\}$,which could competing ToR buffer pool resources between the sub-flows and perform a special many-to-one transmission mode, even though sub-flows traffic from the same MPTCP connection can be load balanced across multiple links,

When the $S_i\{s_1, s_2, \ldots, s_k\}$ transmit data to the same receiving end, that is, simultaneously injects the sub-streams of different MPTCP connections into the bottleneck link $P_i\{p_{i1}, p_{i2}, \ldots, p_{in}\}$, the traffic competing ToRs resource. The links Pij become the bottleneck links, and the ToRs become the transmission hotspot nodes.

We set the congestion window of the sub-flows Fij to be CWij, and the sum of the congestion window sizes of all sub-flows is CWtotal. When the traffic load is too large, that is, $\sum_0^n(ToR) < CW_{total}$, ToRs will not be able to handle burst traffic because the buffer pool is too small. A large number of packets overflow from the ToRs buffer pool and Head of Line blocking (HoL) [28]. Especially when there is a large difference of delay between sub flflows, especially one of the sub flflow has a retransmission timeout problem, which can cause timeout retransmission and the drop in throughput of receivers.

B. DESCRIPTION OF QCBF

In Incast communication pattern, in order to avoid throughput collapse and improve receiver throughput, this paper proposes a buffer pool balance factor, builds a ToRs cluster cache balance queue, calculates a sub-flow congestion window value based on ToRs cluster cache allocation, and designs a queue cache balance factor based on the queue cache balance factor called QCBF. The ToRs cluster resource buffer pool architecture is shown in Fig 3.

For convenience, we make the following assumptions about the system:

(1) Supposing the number of K MPTCP connections share the bandwidth of the number of n bottleneck links, and the bandwidth is C, and RTT is T_{ij}.

(2) Defining the ToRs buffer pool as the balance queue Q, which receives and stores each link data as process P temporarily, the total number of processes is $N_p = n * k$, and there are the number of k processes under each balance queue.

The method of QCBF needs to distribute data packet traffic to processes of N_p to load balance the balanced queues of n without causing throughput collapse. Define the expectation of balance queue i is $Q_{i,ave}$, and set the data packet traffic distribution mode is M, the data traffic receiving load balancing factor $B_i(M)$ on the balanced queue is defined as shown in (1), which depicts the relative deviation of actual data traffic receiving load and expectation on the balanced queue. The amount of data actually received in the queue is set $Q_i(M)$.

$$B_i(M) = \frac{Q_i - Q_{i,\,ave}}{Q_{i,\,ave}}, \tag{1}$$

when, the ToRs cluster resource pool were fully utilized and didn't occur collapse. The data packet that can be accommodated on the link L_i is $D_{iL} = CT_{ij}$. QCBF dynamically adjusts the sub-flow congestion window value according to the balance queue factor, and sets the congestion window value of any sub-flow is W_ij. When the total window value satisfies the (2), the collapse does not occur and the bottleneck link is fully utilized.

$$\sum_{i=0}^{N^P} W_i \in \left[n \times D_i^L,\ n \times D_i^L + n \times Q_{i,\,ave}\right] \tag{2}$$

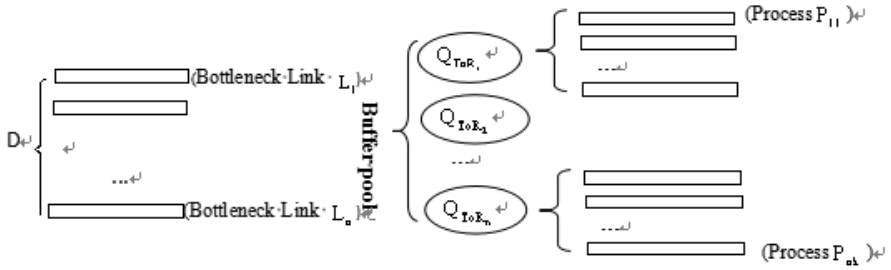

FIGURE 3. Architecture of ToRs cluster resource pool.

Then the amount of data that can be received on the balance queue i is as shown in (3).

$$Q_i(M) = min\{(\sum_{j=1}^{N^P} W_j - n \times D_i^L/n),\ Q_{i,\,ave}\} \tag{3}$$

The QCBF data packet traffic load balancing model is designed to find an allocation scheme so that the following constraint optimization problem is established. The amount of data received and stored by the the number of N processes of the ToRs cluster is equal to the total amount of data of the sender S cluster switch. The value is $Q_{i,ave}$, assigned to the number of N_F passed sub-flows fairly.

The QCBF adjusts the size allocation of buffer pool through β. Among this,the congestion window value obtained by any MPTCP sub-flow is as shown in (4), and the constraint condition is as shown in (5):

$$W_{ij} = \frac{\beta \times Q_{i,\,ave}}{N^F} \tag{4}$$

$$B_i(M) \le 0, \quad i \in (0,\ 1,\ 2, \ldots,\ N_P - 1) \tag{5}$$

C. METHOD OF QCBF DISPOSITION

QCBF adopts a window-based congestion control method to design sender S, the ToR cluster and receiver D respectively.

(1) Sender S. The sender initializes the window field which carries the window value in the header of the packet when it sends the packet, and initializes it to 0xffff. After sender receiving the ACK, the value of the window field is updated according to the window value carried in the ACK header, that is, the value of the congestion window of receiver.

(2) ToRs cluster. The ToRs cluster is mainly responsible for maintaining the number of flows NFbased on the balanced queue formula and calculating the window value, then assigning it to the passing packets. Each of ToRs maintains the number of flows which through the port. When a SYN packet is received by ToRs, the number of flow is increased by one. Oppositely when a FIN packet is received, the number of streams is decremented by one. Regarding the number of passed packets, when $B_i(M) > 0$, the sub-flows is closed to no longer receive the data packet. ToRs compares the window value carried in the packet header and the calculated congestion window value, and the smaller value will be assigned to the window domain.

(3) Receiver D. After receiving the data packet, the receiver compares the value carried in the window field in the packet header with the receiving window awnd, and assigns the smaller value of the two to the window in the ACK packet header, and then feeds back to the sender. The QCBF process is shown in Fig 4.

The algorithm pseudo code of QCBF congestion control is described as follows(where wnd is the sending window and awnd is the receiving party notification window):

Algorithm 1 PARTITION(A, p, r)

$Wnd = min(cwnd, awnd)$
$Cwnd = 1, N = 0, Ssthresh = 65535bytes$
$//WhennewlyACKarriving$
if $Cwnd < Ssthresh$ **then**
  $//SlowStart$
  $N = N + 1$
end if
if $B_i(M) > 0$ **then**
  $//CloseSubMPTCP$
else
  $//QCBF$
  $W = \frac{\beta \times Q_{i,\,ave}}{N_F}$
end if
if $wnd = 1 || wnd > w$ **then**
  $Wnd = w$
  $Awnd = min(awnd, wnd)$
else
  $//CongestionAvoidance$
  $Cwnd = Awnd$
  $//Timeout$
  $Ssthresh = max(2, min(cwnd/2, awnd))/n$
  $Cwnd = 1$
end if

IV. EXPERIMENT EVALUATION

To verify the MPTCP QCBF Method effectively, we uses the NS3 simulation tool [29] in the DCNs multi-homed fat tree topology. Here we take the dual-homed FatTree topology as an example,which is shown as Fig.2. We select TCP-newReno transmission mechanism and MPTCP transmission control mechanism based on LIA congestion control method

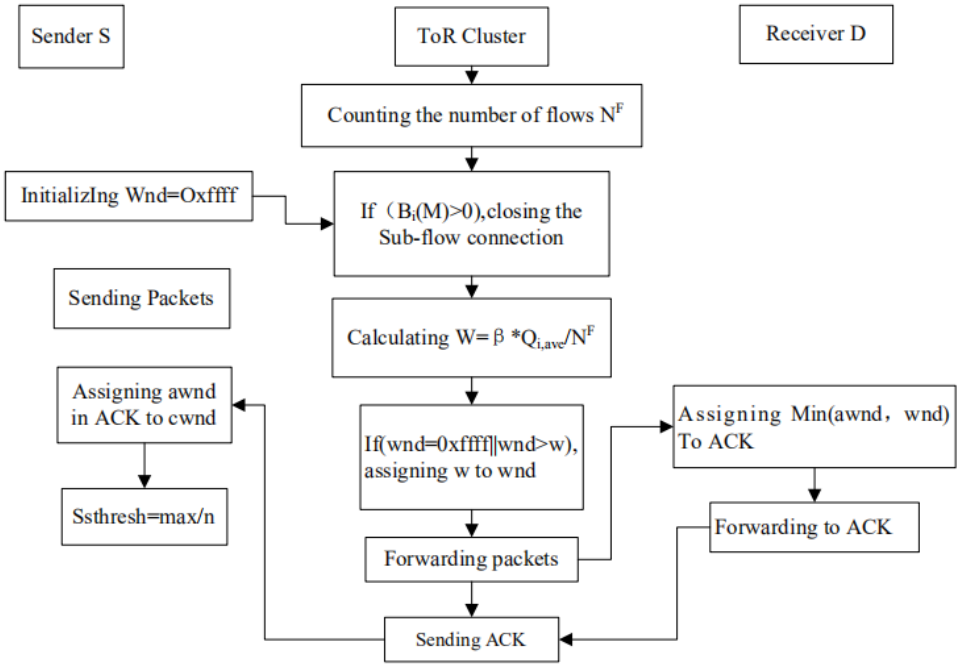

FIGURE 4. Flow diagram of QCBF method.

TABLE 1. The simulation of experimental topology.

The Number of Senders	The Number of Links I	The Number of ToRs	The Number of Links II	The Number of Receivers
4	8	2	2	1
8	16	2	2	1
16	32	2	2	1
24	48	2	2	1
32	64	2	2	1
48	96	2	2	1

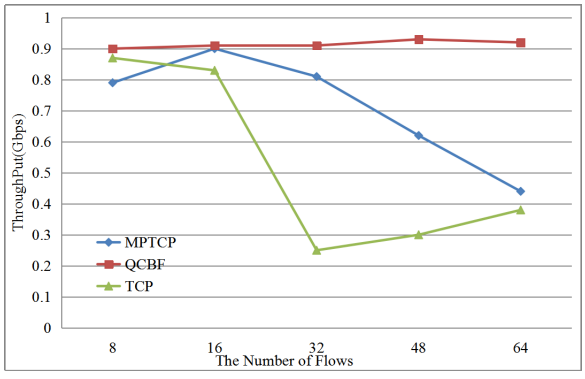

FIGURE 5. Comparison of throughputs for different senders.

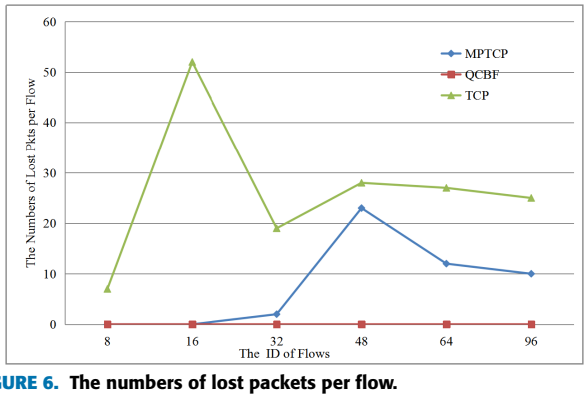

FIGURE 6. The numbers of lost packets per flow.

as a comparison scheme to verify the performance of QCBF transmission control method.

The RTO$_m in$ is set to 200 ms, which is a typical round trip time in DCN. The topology data of the experimental simulation is shown in Table 1. We set the link bandwidth C to 10 Gbps and the size of data is 128 KB. The resource cache size of each ToRs cluster is set to 512 kb.

We perform 6 sets of experiments and simulate the process in which multiple multi-homed hosts send data to the same receiver through two ToRs until the ToRs cluster buffer pool is filled, then we get the statistics of the data packets average delay. The topology data of the experimental simulation is shown in Table 1.

Fig 5 and Fig 6 depict the throughput of different senders in Incast transmission pattern and the number of timeout retransmissions per second respectively

The results show that the throughput of MPTCP drops sharply when the number of senders exceeds 32. The QCBF

is not affected by the number of senders, and the throughput has been stable. Because the packets injected into the network under QCBF will not exceed traffic of Network, lost of packets won't occur, thus the Incast problem has been solved. But MPTCP will lose packets because the source cannot respond quickly to the burst.

V. CONCLUSION

In this paper, a MPTCP Transmission Control method QCBF is proposed to avoid throughput collapse and achieve the load balance of MPTCP data transmission and improve the throughput of the receiving en. Based on ToRs cluster cache allocation we design a queue cache balance factor to calculate a sub-flow congestion window value. The simulation experiment shows that the proposed algorithm outperforms the traditional congestion control algorithm in reducing the number of retransmissions timeout, reducing the transmission delay, and improving the throughput, which can avoid the MPTCP Incast problem in the DCNs and have the characteristics of minimal packet loss and rapid convergence. Also it provides a theoretical basis for the related research of MPTCP Incast. As the research progresses, it is found that there is still a lot of work worthy of follow-up attention and study. First,we will differentiates different types of streams to provide time-delayed services and combine the QCBF and the switches scheduling mechanisms, which can allow the switch to prioritize the scheduling of streams with higher priority. Further, SDN technology can be used to design a more complete bandwidth guarantee mechanism.

ACKNOWLEDGMENT

The authors would like to thank International Networks Service and Bio-computing Innovation Team from the college of Computer and Communication Engineering in China University of Petroleum (East China). Thanks to the team members for their discussion and technical support.

REFERENCES

[1] K. Habak, K. A. Harras, and M. Youssef, "Bandwidth aggregation techniques in heterogeneous multi-homed devices: A survey," *Comput. Netw.*, vol. 92, no. 19, pp. 168–188, Dec. 2015.

[2] P. Xue *et al.*, "Overview of multipath routing optimization technology based on MPTCP," *Comput. Res. Developent*, vol. 53, no. 11, pp. 2512–2529, 2016.

[3] Y. Ren, Y. Zhao, P. Liu, K. Dou, and J. Li, "A survey on TCP Incast in data center networks," *Int. J. Commun. Syst.*, vol. 27, no. 8, pp. 1160–1172, Aug. 2015.

[4] M. A. Öztürk, V. Cojocaru, and R. C. Wade, "Toward an ensemble view of chromatosome structure: A paradigm shift from one to many," *Structure*, vol. 26, no. 8, pp. 1050–1057, Aug. 2018.

[5] M. Abd-El-Malek, W. V. Courtright, II, C. Cranor, G. R. Ganger, J. Hendricks, A. J. Klosterman, M. Mesnier, M. Prasad, B. Salmon, R. R. Sambasivan, S. Sinnamohideen, J. D. Strunk, E. Thereska, M. Wachs, and J. J. Wylie, "Ursa Minor: Versatile cluster-based storage," in *Proc. Conf. File Storage Technol.*, Dec. 2005, pp. 13–16.

[6] J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool," *Commun. ACM*, vol. 53, no. 1, pp. 72–77, Jan. 2010.

[7] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proc. Int. Conf. Netw. Protocols*, 2000.

[8] S. Ferlin, Ö. Alay, T. Dreibholz, D. A. Hayes, and M. Welzl, "Revisiting congestion control for multipath TCP with shared bottleneck detection," in *Proc. IEEE INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[9] F. Zhou, T. Dreibholz, X. Zhou, F. Fu, Y. Tan, and Q. Gan, "The performance impact of buffer sizes for multi-path TCP in Internet setups," in *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl.*, Mar. 2017, pp. 9–16.

[10] S. Shin, D. Han, H. Cho, J.-M. Chung, I. Hwang, and D. Ok, "TCP and MPTCP retransmission timeout control for networks supporting WLANs," *IEEE Commun. Lett.*, vol. 20, no. 5, pp. 994–997, May 2016.

[11] A. Kesselman and Y. Mansour, "Optimizing TCP retransmission timeout," in *Proc. Int. Conf. Netw.*, 2005.

[12] S.Pang *et al.*, "Performance evaluation of MPTCP incast based on queuing network," *IEEE Access*, 2019.

[13] A. Kulkarni and T. Mohsenin, "Low overhead architectures for OMP compressive sensing reconstruction algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 6, pp. 1468–1480, Jan. 2017.

[14] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 1–12.

[15] B. Khasnabish, "Mechanisms for optimizing LAG/ECMP component link utilization in networks," Tech. Rep., 2014.

[16] M. Zhu, L. Wang, Z. Qin, N. Ding, J. Fang, T. Liu, and Q. Cui, "BELIA: Bandwidth estimate-based link increase algorithm for MPTCP," *Inst. Eng. Technol. Netw.*, vol. 6, no. 5, pp. 94–101, Sep. 2017.

[17] Y. Özcan, F. Guillemin, P. Houzé, and C. Rosenberg, "Fast and smooth data delivery using MPTCP by avoiding redundant retransmissions," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–7.

[18] M. Li, A. Lukyanenko, S. Tarkoma, and A. Ylä-Jääski, "MPTCP incast in data center networks," *China Commun.*, vol. 11, no. 4, pp. 25–37, Apr. 2014.

[19] A. Walid, Q. Peng, J. Hwang, and S. Low, "Balanced linked adaptation congestion control algorithm for MPTCP," Tech. Rep., Jul. 2016.

[20] M. Kheirkhah and M. Lee, "AMP: A better multipath TCP for data center networks," in *Proc. Conf., IFIP Netw.*, Jan. 2019.

[21] J. Ye, L. Feng, Z. Xie, J. Huang, and X. Li, "Fine-grained congestion control for MultiPath TCP in data center networks," *IEEE Access*, vol. 7, pp. 31782–31790, 2019.

[22] B. Kimura and A. Loureiro, "MPTCP linux kernel congestion controls," Tech. Rep., Dec. 2018.

[23] S. Tariq and M. Bassiouni, "Performance evaluation of MPTCP over optical burst switching in data centers," in *Proc. Int. Telecommun. Symp. (ITS)*, Aug. 2014.

[24] M. Fukuyama, N. Yamai, and N. Kitagawa, "Throughput improvement of MPTCP communication by bicasting on multihomed network,"in *Proc. 5th ICT Int. Student Project Conf. (ICT-ISPC)*, May 2016.

[25] G. Jereczek, G. Lehmann-Miotto, D. Malone, and M. Walukiewicz, "Approaching incast congestion with multi-host Ethernet controllers," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2018, pp. 1–6.

[26] P. Dell, "On the dual-stacking transition to IPv6: A forlorn hope?" *Telecommun. Policy*, vol. 42, no. 4, pp. 501–582, Aug. 2018.

[27] C. Zhang, G. Feng, and Y. Xiao, "Research of data center network topology," *Intell. Comput. Appl.*, vol. 4, no. 5, pp. 94–96, Mar. 2014.

[28] R. Peñaranda, C. Gómez, M. E. Gómez, P. López, and J. Duato, "IODET: A HoL-blocking-aware deterministic routing algorithm for direct topologies," in *Proc. IEEE 18th Int. Conf. Parallel Distrib. Syst.*, Jan. 2013, pp. 702–703.

[29] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, 2010, pp. 15–34.

SHANCHEN PANG received the degree from the Tongji University of Computer Software and Theory, Shanghai, China, in 2008. He is currently a Professor with the China University of Petroleum, Qingdao, China. His current research interests include theory and application of Petri net, service computing, and trusted computing.

JIAMIN YAO received the B.S. degree in networks engineering and accounting from the Shandong University of Science and Technology, Qingdao, China, in 2017. She is currently pursuing the master's degree with the China University of Petroleum. Her research interests include network protocols and cloud computing.

XUN WANG received the degree from Guangxi Normal University of Mathematical Science, Guilin, China, in 2010. She is currently an Associate Professor with the China University of Petroleum, Qingdao, China. Her current research interests include wisdom medical and bioinformatics.

TONG DING received the B.S. degree from the Nanjing University of Posts and Telecommunications, Nanjing, in 2016. He is currently pursuing the degree with the China University of Petroleum, Qingdao, China. His current research interests include DNA computing, membrane computing, and bioinformatics.

LI ZHANG received the B.S. degree in networks engineering and accounting from the Shandong University of Science and Technology, Qingdao, China, in 2017. She is currently pursuing the master's degree with the China University of Petroleum. Her research interests include software test and cloud computing.