IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 16, 2021, accepted March 2, 2021, date of publication March 4, 2021, date of current version March 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3064008

Survey on Traffic Management in Data Center Network: From Link Layer to Application Layer

WEIHE LI[1], JINGLING LIU[1], SHIQI WANG[1], TAO ZHANG[2], SHAOJUN ZOU[1], JINBIN HU[3], WANCHUN JIANG[1], (Member, IEEE), AND JIAWEI HUANG[1]

[1]School of Computer Science and Engineering, Central South University, Changsha 410083, China
[2]Hunan Province Key Laboratory of Industrial Internet Technology and Security, Changsha University, Changsha 410022, China
[3]School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410077, China

Corresponding author: Jinbin Hu (jinbinhu@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872387 and Grant 61872403, in part by the China Education and Research Network (CERNET) Innovation Project under Grant NGII20170107, and in part by the Project of Foreign Cultural and Educational Expert under Grant G20190018003.

ABSTRACT Due to the explosive growth of all kinds of Internet services, data centers have become an irreplaceable and vital infrastructure to support this soaring trend. Compared with traditional networks, data center networks (DCNs) have unique features, such as high bandwidth, low latency, many-to-one communication mode, shallow buffered switches, and multi-root topology. These new characteristics pose a lot of challenges to previous network technics (e.g., Ethernet, Equal Cost Multi-path (ECMP), TCP), making them hard to adapt to DCNs and leading to severe performance degradation. In order to solve these challenges, DCNs have attracted a lot of attention from the industry and academia in recent years, and many new mechanisms in different layers are proposed to improve the transmission performance of data center networks. In the meantime, many surveys have emerged currently to introduce the current research of data center networks. However, previous surveys of DCNs mainly focus on only one specific network layer, making them difficult for readers to know about advanced researches on a holistic level. To help readers comprehend the current research progress of data center networks quickly, we employ a multi-layered top down taxonomy to classify the literature and propose several probable dimensions for future research in this area.

INDEX TERMS Data center network, congestion control, load balancing, RDMA.

I. INTRODUCTION

In recent years, with the boom of advanced technologies such as cloud computing, big data, and distributed storage, more and more companies have built large data centers to provide various online services like web search, online gaming, recommender systems, etc. The data center has become an essential and irreplaceable infrastructure for the ever-growing Internet services and applications. The data center network (DCN) plays a critical role in a data center, as it supports intra-data center communications among a large amount of computing resources [1].

Nevertheless, unlike the traditional networks such as Local Area Networks (LAN) and Wide Area Networks (WAN), DCN faces a series of new challenges and requirements. For example, it is common for a data center to contain thousands of servers, and the scale of the data center is soaring dramatically at an exponential speed, increasing the challenges on network design in terms of interconnection, cost, and robustness [2]. Besides, the miscellaneous services and applications in the data center provide varieties of distinct traffic characteristics. All these new features together pose many challenges to data centers. To tackle these significant technical challenges, DCNs has attracted the attention of industry and academia, and a lot of new designs from different network layers have been proposed to improve network performance in DCNs.

In order to improve asset utilization and reduce capital expenditures, Ethernet is being enhanced as a unified data center structure in the link layer, which supports IP communication traffic, storage data and high-performance computing traffic [3]. To support these kinds of traffic, researchers

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

have developed some new approaches, such as Quantized Congestion Notification (QCN) [4] and Priority-based Flow Control (PFC) [5]. In March 2010, QCN has been ratified to be the standard for the end-to-end congestion management of Data Center Ethernet [4]. Nevertheless, QCN has several drawbacks that significantly impair its performance, especially under the incast scenario [6], [7]. To avoid excessive packet loss and provide low latency, many schemes have been proposed in recent years. Besides, different cloud computing applications pose some new requirements for DCNs, such as low latency and high throughput. Existing traffic control schemes in DCNs suffer from the intricate kernel processing, increasing the packet delay and thus decreasing the network throughput [8]. Remote direct memory access (RDMA) [9] is regarded as a solution to meet the demand of low transmission latency via bypassing the kernel processing. Thus, many new mechanisms based on RDMA have been proposed recently.

To support various services, a lot of multi-rooted tree topologies such as Fat-tree [1] and Clos [10] are employed to utilize the multiple paths between the source and destination hosts to provide high bandwidth in the data center. To achieve better application performance, how to balance traffic and thus achieve the low latency and high throughput becomes a significantly crucial issue in large-scale data center networks. Equal Cost Multi-path (ECMP) [11] is the standard load balancing scheme to randomly map each flow to one of the available paths via flow hashing. ECMP is simple and effective but suffers from some performance problems such as hash collisions and the inability to reroute flows adaptively. In order to balance traffic better, a lot of new load balancing designs in network layer have emerged in DCNs.

Applications in DCNs generate a mix of short and long flows with different performance requirements. In general, the short flows are typically delay-sensitive and are prone to suffer from the head-of-line blocking and packet reordering problems. Since the traffic pattern in data center network is quite distinct from the traditional network, traditional TCP cannot be directly employed in the data center network environment, as it tends to run out of switch buffer for higher network throughput, deteriorating the transmission performance [12]. To address these challenges, a series of work has been introduced in the transport layer.

With the development of cloud computing, a variety of applications have recently been widely deployed in data centers. Key-value stores have become an essential part of large-scale applications. A single request may generate key-value operations on many servers, and the tail delay of the key-value access operation can affect the response time of the request to a great extent [13], [14]. Thus, how to design a good replica selection algorithm in the application layer to improve users' experience has become a hot topic in recent years.

In order to help readers quickly understand the current research of data center networks, some surveys have emerged in recent years [2], [3], [8], [10]. Reference [8] presents a survey of RDMA that includes traditional RDMA, traditional DCNs, and RDMA-enabled DCNs. Reference [10] reviews a survey of load balancing mechanisms and discusses future design directions in data center networks. Reference [3] summarizes distinct schemes for the flow transmission in DCNs. Reference [2] provides a comprehensive review of infrastructure and network topologies in DCNs. However, previous surveys of DCNs mainly focus on only one specific network layer, making it difficult for readers to know about advanced researches on a holistic level.

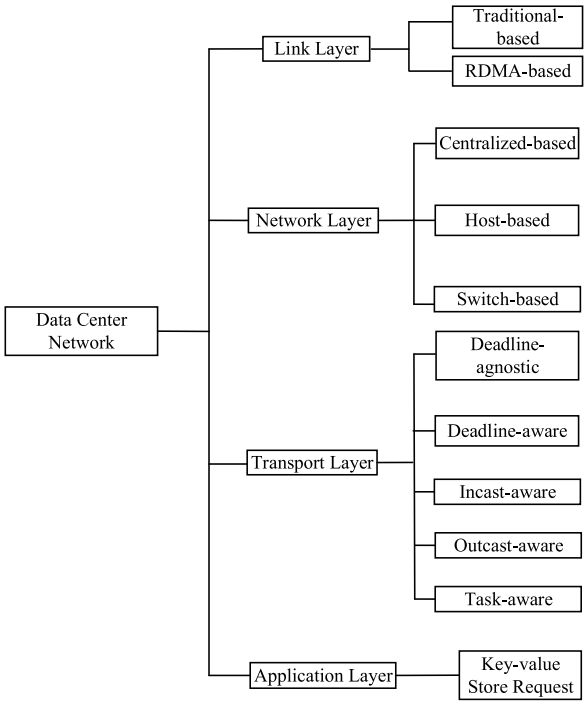


FIGURE 1. Structure of this survey.

Although the network adopts a hierarchical structure, there is a coupling relationship between different layers. In recent years, many mechanisms are no longer designed in a specific layer. On the contrary, they adopt the idea of cross-layer fusion. For example, NDP [24] uses the combination of the transport layer and network layer to optimize the traffic transmission in DCNs. In addition, the design of some link-layer schemes draws on some mechanisms in the transport layer, such as ECN, to strengthen their functions. From the development trend, some mechanisms of different layers can merge with each other, thereby enhancing the function of DCNs. Therefore, unlike previous surveys that only focus on one specific layer, this study reviews kinds of literature from different network layers to provide readers with a holistic picture of DCNs via a multi-layered top down taxonomy. Fig.1 demonstrates the structure of this survey to give an idea of subsections quickly. Note that besides the multi-layered top down taxonomy, there exist some other methods to classify different schemes in a survey paper. How to use other approaches to organize the literature of data center networks is a direction that needs to be studied further.

In this review, we focus on the data center literature and search the on-line library databases with the keywords: data center network, congestion control, QCN, RDMA, load balancing, transport protocols, and key-value store. We first found and downloaded around 200 papers, and then picked and summarized over 110 pieces of literature on traffic control from different network layers in data centers over the past decade (from 2008 to 2020). Since this survey paper discusses a lot of information, we pick some representative schemes and add a summary table as the Appendix to help readers understand the content more intuitively. The main journals include in this review: *IEEE/ACM Transactions on Networking, IEEE Journal on Selected Areas in Communications, IEEE Transactions on Communications, IEEE Transactions on Cloud Computing, IEEE Communications Surveys & Tutorials, IEEE Transactions on Parallel and Distributed Systems*. The main conferences include in this review: *ACM SIGCOMM, USENIX NSDI, IEEE INFOCOM, ACM CoNEXT, ACM EuroSys, IEEE ICNP, IEEE ICDCS, IEEE/ACM IWQoS, ACM SOSR, ACM ICPP*.

We make the following contributions:

- Unlike previous surveys concentrate on one network layer, this paper uses a multi-layered top down taxonomy to classify the literature in different network layers to help researchers understand the advanced work on a whole level.
- This survey introduces schemes on different layers in a detailed fashion and summarize the advantages and disadvantages of each type methods thoroughly.
- This survey also indicates some future research directions to guide readers to make more contributions in this field.

In the rest of the survey, we elaborate on the schemes from each layer. In Section II, we introduce some terminologies in the data center networks. In Section III, we illustrate the work in the link layer. We introduce all kinds of load balancing schemes in network layer according to their deployment location in Section IV. Section V describes a series of newly emerged transport protocols designed for DCNs. Section VI elaborates on the design of replica selection algorithms in the application layer. Section VII illustrate some cross-layer designs. Section VIII describes several future prospects in data center networks.

II. BACKGROUND

In order to enable readers to have a basic understanding of the data center, this section introduces some terminologies related to the data center networks.

- **Elephant/Mice Flow:** The flow arrival process of tremendous traffic exhibits ON/OFF patterns, increasing the burstiness and unpredictability of traffic [15]. The traffic in data center can be characterized by heavy-tailed distribution; around 90% of data is provided by around 10% of TCP flows (elephant flows), while about 90% of TCP flows provide only about 10% of data (mice flows). In addition, most mice flows are delay-sensitive and most elephant flows are throughput-sensitive [16].
- **Bisection Bandwidth:** The maximum capacity between any two servers in data centers is defined as the bisection bandwidth, which is used to compare the potential throughput between any two halves of the network topology [17].
- **Incast:** In the Incast communication pattern, a large number of senders simultaneously transmit data block to a single receiver, and every sender cannot send new data block until all the senders complete the current transmission ones [18]. As the number of senders increases, TCP Incast happens when at least one flow experiences timeout due to full window loss or tail loss, which leads to TCP goodput becomes lower than the capacity of the bottleneck link in one or even two orders of magnitudes.
- **Outcast:** When multiple flows from different input ports compete for a same output port at a switch, one set of flow's packets are dropped consecutively due to buffer overflow, leading to TCP Outcast occurs [19].
- **Task:** A task is made up of hundreds of flows, and all flows should finish their transmission before the task is treated complete [20]. Current flow-based mechanisms treat all flows separately, impairing user's experience owing to the stalled flows.

III. LINK LAYER

Ethernet has become the primary network protocol of data center networks due to its multiple advantages, such as low cost and rich connectivity. However, Ethernet was originally designed for the local area network and was not optimized for data center networks. To improve classical Ethernet performance in data center networks, The IEEE 802.1 Data Center Bridging task group proposes several specifications of congestion notification schemes to mitigate packet loss in data center networks. Besides, since remote direct memory access (RDMA) allows the network interface cards (NICs) to directly transfer data to and from application memory, it is an excellent mode to alleviate the processing time of kernel processing and thus reduce packet delay. The designs based on RDMA spring up like mushrooms in recent years. In the following, we will elaborate on these schemes.

A. TRADITIONAL-BASED SCHEMES

Quantized Congestion Notification (QCN)[1] [4] is a standard end-to-end congestion management for data center Ethernet, which consists of two modules: Congestion Point (CP) and Reaction Point (RP). Fig.2 depicts the architecture of QCN. The main aim of Congestion Point is to maintain the switch's buffer level at a desired level. CP samples the incoming data packets with a probability in the switch to determine whether the network is congested or not. If the network is congested, CP will generate the feedback packet, including

[1] https://github.com/hsr/qcn-emulator

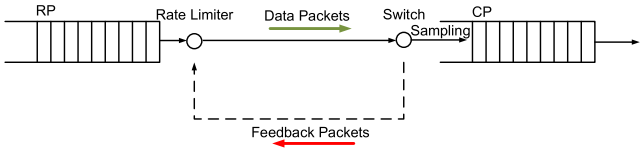


FIGURE 2. QCN architecture.

the congestion information and send it back to the source. Reaction Point (RP) is used to adjust the sending rate at the source side. It decreases the sending rate according to the quantized congestion feedback value contained in the feedback packet and increases the sending rate by itself to recover bandwidth and probe for free available bandwidth. Although QCN can control link rates to cope with congestions in several round trip time, it also suffers from some problems, like frequent queue oscillation, unfairness, link utilization degradation, and so forth.

The performance of QCN relies on both parameters setting and network configurations to a great extent. When system parameters or network configurations change, it might cause frequent queue oscillation, leading to link utilization degradation. QCN also depends on the sliding mode motion to approach to the equilibrium point. Nevertheless, QCN might fail to reach to that point due to inappropriate parameter setting, making it unstable in some situations. To address these issues, Sliding Mode Congestion Control (SMCC) [6] is proposed. Since the sliding mode motion is insensitive to system parameters and external disturbances, SMCC exploits this advantage to guarantee the system to enter into the sliding mode motion under any situations, making the system robust and stable. Besides, to accelerate the responsive speed, the queue length and its derivative information are fed into the feedback packet separately.

Since a CP delivers each feedback packet to an RP randomly, the random feedback leads to unfairness among RPs. To achieve better QCN fairness when multiple flows share a bottleneck link, fair QCN (FQCN) [5] recognizes congestion via queue and per-flow monitoring and feedbacks the congestion information through multi-casting to guarantee the convergence to statistical fairness. Moreover, the stability and fairness of FQCN are analyzed through the Lyapunov function.

QCN detects congestion via a combined value F_b that is the weighted sum of rate variance and queue variance. The incomplete binary search might make QCN fail to find the appropriate sending rate. QCN is also slow to converge to the desired target queue, leading to the resource under-utilization. Fast and Simple QCN (FSQCN) [21] distinguishes whether a network is congested by both combined quantity F_b and the rate variance term in F_b to deal with the first problem. Since the switch detects the network environment more accurately and swiftly, FSQCN utilizes the switch to directly notify the sender to transmit data at the line rate when there exists spare bandwidth to improve the bandwidth utilization.

Although QCN has been investigated on simulations and implementations for several years, the in-depth theoretical analysis is insufficient. To better understand QCN's mechanism in a theoretical fashion, literature [22] utilizes the phase plane method to analyze the QCN system. It builds a fluid-flow model for the QCN system firstly, and then sketches phase trajectories of the rate increase and decrease. The analytical results demonstrate that the QCN's stability is mainly promised via the sliding mode motion. It also shows that whether QCN can enter into the sliding mode lies in the parameters setting and network configurations. Literature [23] derives the uniformly asymptotic stability condition of the QCN system based on the fluid model. Then under the condition that QCN is uniformly asymptotically stable, [23] analyzes the buffer occupancy of the QCN switch and gives an explicit formula for the boundary of buffer occupancy.

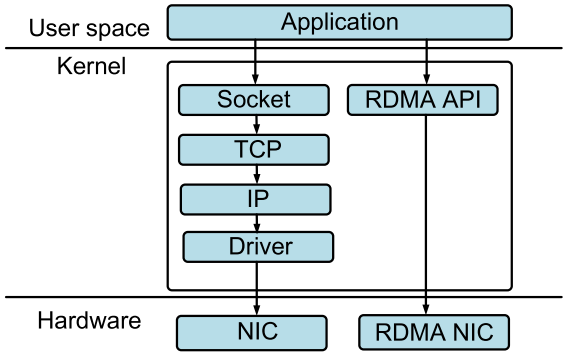


FIGURE 3. Difference between socket programming and RDMA programming.

B. RDMA-BASED SCHEMES

A data center network connects thousands of servers in the data center and utilizes a traffic control mechanism to transmit all servers' data. Numerous new applications and services pose new prerequisites on traffic control of DCNs, such as low latency and high throughput. Traditional TCP/IP stacks cannot meet these requirements due to the high CPU overhead and sophisticated kernel processing. To tackle this issue, some new schemes are introduced recently.

Remote Direct Memory Access (RDMA) [9] is considered as an excellent approach to meet requirements. Compared with traditional traffic control mechanisms, the NIC-based data transfer dramatically decreases CPU overhead and overall latency by avoiding data copies between application memory and data buffers and bypassing the kernel processing [8]. Fig.3 shows the difference between socket programming and RDMA programming. In traditional socket programming, user space's applications need to initialize the socket in the kernel. Then the socket transmits data through TCP, IP and NIC drivers before accessing NIC. Data delivery between different kernel layers needs CPU processing and memory replication. In comparison, RDMA bypasses the processing in the TCP/IP stacks and delivers data from the RDMA NIC application through RDMA application programming interface (API), decreasing the transmission latency and CPU overhead.

1) TRADITIONAL RDMA TRAFFIC CONTROL

To employ RDMA in Ethernet and IP networks, several protocols have been proposed. The most representative schemes are RoCE and iWARP. To construct RDMA on lossless Ethernet, RDMA over Converged Ethernet v2 (RoCEv2)[2] applies Priority-based Flow Control (PFC) [5] to avoid packet loss at the link layer, while iWARP[3] implements RDMA in the lossy networks through deploying TCP/IP stack in the NIC. In the following, we will elaborate on the design details of the two protocols.

RoCEv2 is a network protocol that enables RDMA to adapt to existing network infrastructure. RoCEv2 can convert the RDMA package to a UDP package. The UDP header is used for ECMP, and IP header is used for routing. RoCE avoids packet reordering via go-back retransmission and preventing packet loss through PFC.

Traditional RDMA design employs go-back-0 scheme to tackle out-of-order packets. When a sender transmits a volume of data packets to the receiver, the receiver will discard the packet and request the sender to deliver all the packets again if there is a data packet out of order. It leads to the application's throughput degradation since the application cannot process these data until it receives all the packets. To solve this problem, RocEv2 utilizes the go-back-N scheme. The receiver only lets the sender retransmit packets after the last acknowledged one. This fashion can mitigate the throughput decline but still waste the bandwidth for delivering extra packets.

PFC is a layer two congestion control approach. Downstream switches can deliver PFC pause packets to upstream switches to throttle the traffic. When one ingress queue of the downstream switch reaches the PFC threshold, it will transmit a PAUSE frame to its corresponding upstream egress queue. Once an upstream port receives a pause frame, it quits data delivery for a given priority level according to the priority and pause duration carried in the pause frame. After the pause duration has expired, the upstream port will restart data transmission. However, PFC easily leads to unfairness problem. When multiple flows from several different input ports incur congestion on a single output port, all ports can be suspended, regardless of their fair bandwidth share. Besides, PFC also suffers from head-of-line blocking problem. Fig.4 shows an example of head-of-line blocking problem. Flow $f1$, $f2$ and $f3$ share ingress queue 1 of the switch, but they are forwarded toward three separate egresses. When egress 1 receives a pause frame, it will send a pause frame to ingress 1 to pause flow $f1$. At the same time, flow $f2$ and $f3$ are also blocked by the pause frame of ingress 1.

Another protocol is iWARP that implements RDMA via moving TCP/IP stack into the NIC. It eliminates the CPU overhead through offloading transport processing from the CPU to the NIC. iWARP builds a reliable connection by TCP's end-to-end loss recovery [7]. Owing to end-to-end loss recovery, iWarp cannot guarantee ultra-low latency like RoCEv2. iWarp is also likely to suffer from a series of TCP problems in the data center environment like poor incast performance. Moreover, it is very complex to implement the full TCP stack in hardware, leading to higher NIC processing costs. Due to these drawbacks, the iWarp technology has lagged behind RoCEv2.

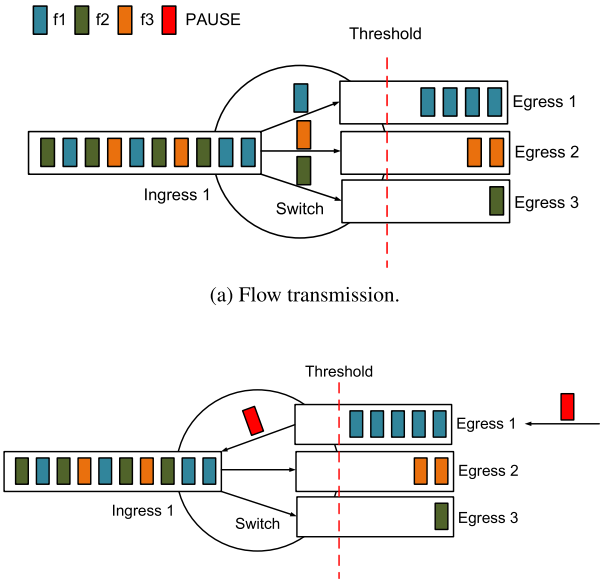


FIGURE 4. Example of head-of-line blocking using PFC.

2) DCN-BASED RDMA TRAFFIC CONTROL

To meet the demands of delay-sensitive flows and throughput-intensive flows in data center networks, many DCN-based RDMA traffic control schemes have been introduced in recent years.

The most representative congestion control scheme in RoCEv2 is DCQCN[4] [7]. DCQCN is a rate-based congestion control. The switch marks the packet when the queue length exceeds the preset marking threshold via Explicit Congestion Notification (ECN). The receiver periodically notifies the sender through a signal packet named Congestion Notification Packet (CNP). Then the sender utilizes explicit rate control and adjusts the flow's rate based on the received CNP packets. Although DCQCN can provide fast convergence and achieve high link utilization, it still employs PFC to avoid packet loss. To reduce the PFC-trigger events, the switches' parameters are required to set scrupulously, such as the ECN threshold, the PFC threshold and the buffer threshold for storing PFC pause packet. The network situation decides the parameter setting, and there is no universal way to configure the switch. Besides, DCQCN cannot suppress the large-scale incast well. When a lot of servers send traffic synchronously, the performance of DCQCN will decrease dramatically. This issue incurs large queue length at the

[2]https://github.com/fpgasystems/fpga-network-stack
[3]https://github.com/linux-rdma/rdma-core
[4]https://github.com/bobzhuyb/ns3-rdma

congestion point, causing a long queueing latency. DCQCN also utilizes fixed period for rate increase to seize the available capacity, leading to poor scalability [25]. To address the problems mentioned above, DCQCN+ [25] employs dynamic parameters scheme. To adapt parameters, senders must know the scale of incast event. DCQCN uses two types of information, including the CNP period carried in CNPs and the flow rate. In the design of DCQCN+, all congested flows of the same receiver share the CNPs equally through time multiplexing. Therefore, the number of congested flows on the receiver can be obtained via CNP period. The CNP period information is carried by an available field in CNPs without the help of additional packets. Since flows with larger rate usually have more packets in the congested queue, leading to a larger possibility to receive a CNP and decrease rate. Thus, all congested flows converge to the almost same rate, which is corresponding to the number of the flow. Based on this information, DCQCN+ adjusts its parameters with two rules. The first rule is the recovery timer should always be larger than the CNP period to receive a CNP and the second one is the total throughput should not increase as the number of congested flows grows. DCQCN+ also uses an additional exponential growth phase to accelerate the flow recovery process after congestion. To accelerate the convergence speed and reduce buffer occupation of DCQCN in incast scenario, Predictive PFC (P-PFC) [26] proactively triggers PFC pause through monitoring the derivative of the buffer occupation, so as to maintain the buffer at a low level and reduce the tail latency. P-PFC checks whether the queue length of egress port exceeds the ECN threshold on the switch. Based on the observation that switch can predict that congestion will happen by monitoring the derivatives of buffer change, P-PFC performs prediction when the congestion emerges. Under congestion, P-PFC firstly judges whether PFC should be triggered. If the PFC should be triggered, it chooses the appropriate ports to send the PFC packets and estimates the pause time. The experimental results show that P-PFC can greatly reduce tail latency compared with standard PFC in many scenarios.

To improve the efficiency of multipath transmission, MPTCP[5] [49] allows a TCP connection to utilize multiple paths to obtain high throughput and avoid packet reordering in data center networks. Nevertheless, the NIC of RoCE does not have enough memory to store each path's congestion status and the packet information of each flow. If the NIC uploads all these status into host memory, the frequent data exchange between the NIC and host memory will rise the latency and decrease the NIC throughput. To enable multiple paths' usage in RDMA, MP-RDMA[6] [27] is introduced. Specifically, MP-RDMA employs an ECN-based multipath ACK-clocking mechanism that can allocate packets to multiple paths without adding per-path status in a congestion-aware fashion. To avoid packet reordering, it actively deletes slow paths and adaptively select a set of fast paths with similar delays, mitigating out-of-order level effectively. Even though MP-RDMA can greatly improve the overall network utilization, its implementation is based on field-programmable gate array (FPGA) and PFC, leading to high deployment cost and poor scalability in data center networks.

All the protocols mentioned previously rely on PFC and go-back-N transmission to ensure lossless network. However, these technics have a lot of drawbacks [28]. In order to remove RDMA's dependence on PFC, IRN [29] deploys RDMA in the lossy network and tackles the packet loss issue in hardware. It employs selective retransmission that the receiver does not discard out-of-order packets and selectively acknowledges them. Thus, the sender retransmits only the lost packets. To reduce queuing in the network, packet-granularity traffic control sets an upper bound for the number of out-of-order packets. The NIC on the sender side firstly calculates the network's bandwidth-delay product (BDP) and uses it to restrict the number of in-flight packets. Only when the number of packets in flight is less than the BDP, a new packet is transmitted by the sender. IRN performs better than RoCE without requiring a lossless network. However, IRN cannot avoid triggering ACK timeouts.

C. SUMMARY

We sum up the current progress in the link layer in the following:

As the standard link congestion control protocol in Data Center Ethernet, QCN provides low latency in Data Center Ethernet by keeping a small queue length at switch buffer. Nevertheless, QCN suffers from various problems, such as unfairness, queue oscillation and so on, leading to performance degradation. To solve the deficiency of QCN, a series of advanced schemes have been proposed to optimize QCN from multiple perspectives, such as parameter settings, theoretical analysis, etc., significantly improving the stability, robustness and fairness.

RDMA is becoming more and more popular in data centers because it can avoid the considerable delay in complicated kernel processing. The typical mechanisms in traditional RDMA networks are RoCE and iWARP. RoCE utilizes PFC to avoid packet loss at the link layer. iWARP implements RDMA by deploying TCP/IP stack in NIC. However, PFC easily encounters the unfairness problem. Moreover, moving stack into NIC is too complicated. To make RDMA more suitable for the data center network, a series of RDMA-based congestion control mechanisms have emerged. The most representative scheme is DCQCN. Nonetheless, DCQCN employs fixed parameters, making it hard to adapt to various network environments. To enhance the QCN performance, DCQCN+ turns its parameters adaptively according to the network status. Besides, many novel approaches are introduced to improve the transmission efficiency in RDMA-based data center networks.

Even though a series of schemes have been proposed in RDMA, there is still much room for improvement in the

[5]https://github.com/multipath-tcp/mptcp
[6]https://github.com/alex–m/mp-rdma

design of RDMA. Some current work employs advanced hardware (such as FPGA) to control RDMA traffic in DCN. These hardware-based approaches are money-consuming for DCNs with commercial RDMA NICs (such as RoCE). How to strike a balance between overhead and performance is a crucial issue. IRN and MP-RDMA utilize existing RDMA NICs to obtain similar performance of new hardware-based mechanisms. Another possible solution is to deploy traffic control in the user-space so that RDMA can only be used for data transmission without any modification, and we can flexibly use existing or customized congestion control schemes as needed.

IV. NETWORK LAYER

Modern data center networks are commonly constructed in multi-rooted tree topologies with multiple available paths between any given pair of hosts to provide high bandwidth [1], [30]–[32]. To fully utilize the resource of parallel paths and obtain better application performance, how to balance the traffic across different paths becomes an essential issue in data center networks.

Equal Cost Multipath (ECMP)[7] [11] is the standard load balancing approach in data center. The switches firstly employ the five-tuple in the packet header for hash calculation. Then it maps an outgoing port for each flow according to the hash value. Though ECMP is very simple, it easily suffers from hash collisions and the inability to adapt to network asymmetry, like link failures and switch failures [33]. Therefore, to guarantee the high bisection bandwidth in the data center, many load balancing mechanisms have been proposed in the network layer.

Nowadays, the load balancing mechanisms in the data center network can be categorized into three types according to their deployment location: centralized-based, host-based and switch-based. In the following, we will introduce related load balancing schemes in detail.

A. CENTRALIZED-BASED LOAD BALANCING SCHEMES

To better balance traffic according to real-time traffic information and global link utilization, it is vital to obtain global network information for load balancing schemes. With the popularity of the Software-Defined Networking (SDN) network in recent years, many mechanisms employ a centralized controller to gather the congestion information of the whole network and reroute flows to an appropriate path. Table 1 lists the centralized-based schemes which have been proposed in recent years.

1) SDN BACKGROUND

The hierarchical structure of the traditional network is the key to the success of the Internet. However, traditional network equipment has many complex protocols, making them hard for operators to optimize the network. In addition, it is also difficult for researchers to deploy new protocols on a large scale. To tackle traditional network inefficiencies, software-defined networking (SDN) has gained much attention in academia and industry due to its centralized control and programmability. For instance, Google's B4 [34], [35] utilizes SDN to run its private backbone network, connecting data centers across the world. The software-defined network control stacks enable flexible and centralized control, providing considerable benefits. Facebook develops and deploys FBOSS [36], [37], a data centre switch software, to reduce the complexity of managing data center networks and test new ideas at scale on time.

SDN decouples the control plane from the data plane, allowing for centralized control of the network. At the control plane, the programmable controller can collect global network information, which is convenient for operators to manage the network and deploy new protocols. The switch only provides data forwarding function at the data plane, which can quickly process data packets to meet the demand for increasing traffic. The data plane and the control plane use the SDN Control-Data-Plane Interface (CDPI) for communication [38]. The controller issues rules to the switch through the standard interface, and the switch only needs to perform corresponding actions by these rules. Therefore, SDN can effectively reduce the equipment load and assist network operators to control the infrastructure better.

2) REROUTING-BASED

To address the long flow collisions problem of ECMP, Hedera[8] [39] estimates large flows and leverages the global information to schedule them to uncongested paths. In Hedera, the controller detects flow rate at edge switches. If the rate of one flow is larger than a preset threshold, e.g., 10% of the link capacity, this flow is treated as a large one. Then it utilizes the Global First Fit scheduler or Simulated Annealing scheduler to choose a path with enough bandwidth to accommodate the large flow. Although Hedera achieves higher bisection bandwidth than ECMP, it takes a long time to finish information collection and large flow rerouting, which cannot adapt to the highly variable traffic in data center networks. Moreover, it is unfriendly to the short flows. To alleviate the high overhead of large flows detection in Hedera, Mahout [41] monitors large flows at the end-host side. The host utilizes a shim layer to monitor the socket buffer. If the socket buffer surpasses a threshold, the corresponding flow is marked as a large one. Then the host will notify the centralized controller with the DSCP field in the packet header. After receiving the marking information, the controller will reroute this flow to a least-congested path. The other flows are scheduled in an ECMP fashion. The deployment of Mahout needs to modify the host kernel, which leads to poor scalability. To reduce the completion time of short flows with high bursty, MicroTE [40] finds out that, although the traffic of data center is bursty, a part of the traffic is still predictable on a short timescale. Then MicroTE counts on a centralized controller to monitor

[7]https://github.com/dharmesh91/Multipath-Routing-in-Data-centers

[8]https://github.com/strategist333/hedera

TABLE 1. Comparison of centralized-based load balancing mechanisms.

Name	Year	Congestion Information	Scheduling Granularity	Advantages	Disadvantages
Hedera [39]	2010	Global	Flow	Avoiding the hash collision of long flows	Long control period and inaccurate long flow recognition
MicroTE [40]	2011	Global	Flow	Improve the performance of the long flows	Can not react to traffic dynamics timely
Mahout [41]	2011	Global	Flow	High efficiency of flow detection	Need to modify host kernel
FastPass [42]	2014	Global	Packet	Achieving low queueing delay	Difficult to deploy on a large scale
FDALB [43]	2016	Global	Flow	Mitigating flow collision events and increasing the scalability of controller	Difficult to obtain the optimal threshold
Freeway [44]	2016	Global	Flow	Solving the resources competition of long and short flows	Hard to use multiple paths flexibly
SAPS [45]	2018	Global	Packet	Suitable for asymmetric networks	Need a long time for controller to obtain topology information
SOFIA [46]	2018	Global	Flow	Reducing the overhead of the controller and the link congestion	The size distribution of flows affects the calculation of threshold greatly

which ToR pairs have predictable traffic and first schedule them. Then it employs weighted equal-cost multipath to route the unpredictable flows.

3) FINER-GRANULARITY-BASED

The rerouting-based controller load balancing schemes generally utilize the flow granularity, making it unable to flexibly seize the available capacity of multiple paths. To address this issue, several load balancers based on packet granularity have been proposed.

Fastpass[9] [42] is a centralized transmission control architecture with finer granularity. It employs a centralized arbiter to decide when each packet should be transmitted and which path it should follow. It consists of three main components: the first one is a fast timeslot allocation algorithm to determine when to send each packet; the second one is a scalable path assignment algorithm to assign a path to each packet; the last one is a replication strategy for the central arbiter to deal with the arbiter and network failures. However, Fastpass assigns timeslot and path for each packet that incurs high overhead, making it hard to deploy at a large scale.

SAPS [45] is an SDN based approach that utilizes packet spraying over symmetric virtual topologies, which is a subset of paths between source and destination with symmetric capacity and delay. The centralized control plane detects asymmetries and data plane enforces packet spraying. Therefore, it can obtain high performance over asymmetric topologies. The SDN controllers will compute an optimized route for each connection request to minimize routing cost and link congestion.

4) FLOW-DIFFERENTIATION-BASED

Modern data centers include a mix of short flows with strict latency requirement and long flows requiring large throughput. To cope with resource competition between heterogeneous flows, various schemes that isolate the mixed traffic have been proposed.

[9]https://github.com/yonch/fastpass

FDLAB [43] partitions the flows into long flows and short flows according to the sent bytes. The flow is marked as a long one if its sent bytes exceeds a presetting threshold. For the short flows, they are transmitted in an ECMP manner. For the long flows, they are rerouted via a controller according to the global information. However, FDLAB only senses the traffic distribution at the sender side, making it hard to calculate the optimal threshold for distinguishing the long flows and short flows.

To reduce the resource competition between long flows and short flows, Freeway [44] schedules the long flows and short flows to different paths. To guarantee the requirement of both kinds of flows, the centralized controller firstly gathers traffic and link status information. Then short flows are transmitted through ECMP, and long flows are centrally rerouted to avoid resource competition. Although the isolation can mitigate bandwidth competition, the coarse-granularity makes it challenging to utilize the multi-paths flexibly.

When a new request appears in the data center network, the controller will receive a packet-in message. However, the flow arrival rate can peak at millions of packets per second, and the frequency of packet-in events will incur excessive latency. To reduce the computation overhead, the controller only schedules the large flows. SOFIA [46] is an online algorithm that learns the best segmentation threshold without a prior knowledge of traffic characteristics based on stochastic approximation.

B. HOST-BASED LOAD BALANCING SCHEMES

Although the controller-based load balancing schemes can make better scheduling decisions via global information, it takes a pretty long time for the controller to collect this information, making it hard to handle flows timely. Moreover, the overhead of the controller makes it hard to deploy in production data center networks. To reduce the overhead and be fast to process each flow, a series of host-based load balancing schemes have been proposed. Table 2 demonstrates the host-based approaches which have been emerged in current years.

TABLE 2. Comparison of host-based load balancing mechanisms.

Name	Year	Congestion Information	Scheduling Granularity	Advantages	Disadvantages
FlowBender [47]	2014	Local	Flow	Alleviating hash collision problem	Random and passive rerouting
CLOVE [48]	2016	Global	Flowlet	No modifications to the hardware switch or virtual machine	INT technology is not universal
Hermes [33]	2017	Global	Short flows: flow, long flows: packet	Suitable for asymmetric networks	Rerouting decision is over-conservative
MPTCP [49]	2011	Oblivious	Subflow	Increasing the performance of long flows	Unfriendly to short flows
MMPTCP [50]	2019	Oblivious	Short flows: packet, long flows: subflow	Accelerating the transmission of short flows and improve the throughput of long ones	Need to modify the TCP/IP protocol stack
MPTCP_OPN [51]	2020	Global	Subflow	Adjusting the number of subflow flexibly and reducing timeout events	Difficult to measure the accurate packet loss rate

1) REROUTING-BASED

To address the hash collisions of ECMP, FlowBender [47] will firstly sense the link congestion via the congestion signal like ECN. After detecting the congestion, FlowBender modifies the packet header to trigger re-hashing. Based on the new hash value, the flow can be sent to a new path to alleviate the network congestion. Due to the random path selection, it is hard to obtain optimal performance.

To achieve better path selection, CLOVE[10] [48] is deployed in the virtual switches of end-host hypervisors that requires no changes to the existing physical switches and network stack. It utilizes standard ECMP and learns about network paths via a traceroute mechanism. After perceiving the congestion through signals like ECN, it will modify the packet header information in the hypervisor switch to reroute traffic to less congested paths. However, INT technology is not ubiquitous recently, which makes it hard to deploy.

To gracefully handle network uncertainties like failures, traffic dynamics and link asymmetry, Hermes[11] [33] leverages RTT, ECN and probing packets to sense uncertainties. Fig.5 shows the overview of Hermes. The sensing module utilizes two congestion signals, RTT and ECN, to sense congestion and link or switch failures in the network. When both the detected RTT and ECN signals show congestion, the path is considered to be congested. When both RTT and ECN signals show light congestion, the path is considered a non-congestion path. Otherwise, the path is considered to be a grey path. When Hermes detects that there are more than three flow timeout events on a path or no ACK packet is received, the path is considered a failed one, and this type of path is ignored during rerouting. Reroute module uses package level scheduling granularity. Hermes monitors the flow status and path conditions instead of vigorously changing paths and employs a timely yet cautious rerouting mechanism. The flow is scheduled to a new path only if it brings enough performance gains. Although Hermes is suitable for asymmetric networks, the over-conservative rerouting scheme might easily link under-utilization.

[10]https://github.com/snowzjx/ns3-ecn-sharp/tree/master/src/clove
[11]https://github.com/snowzjx/ns3-load-balance

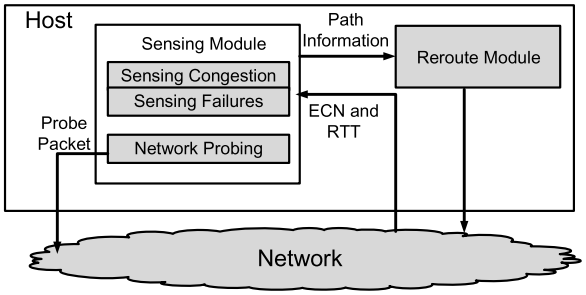


FIGURE 5. Hermes overview.

Although flowlet-based schemes are robust to asymmetric topology, they are vulnerable to suffer from the congestion mismatch problem in re-scheduling flowlets to another path [33]. When a flowlet is transmitted to a new path, its current sending rate is usually not consistent with the new path. Therefore, the flowlet may lead to packet loss or buffer overflow, degrading the network performance.

2) TRAFFIC-SPLITTING-BASED

To increase the link utilization, MPTCP [49] uses multiple subflows as the rerouting granularity to avoid packet reordering and reduce flow completion time. MPTCP achieves full link utilization via running out the link buffer, resulting in considerable queuing latency and packet loss. This can degrade the performance of latency-sensitive short flows. To cope with this problem, MMPTCP [50] distinguishes the long and short flows based on the number of bytes have been sent and employs different transmission schemes for long and short flows. For the short flows, packet granularity is used to seize available path resources and reduce flow completion time quickly. For the long ones, the subflow granularity is adopted to guarantee high throughput.

Existing MPTCP designs are agnostic to the number of subflows, leading to performance degradation. If the number of subflows is small, the bandwidth resources of multiple paths are wasted. On the contrary, if the number of subflows is large, the congestion window of each subflow is small, easily resulting in timeout under heavy congestion. To address this

TABLE 3. Comparison of switch-based load balancing mechanisms.

Name	Year	Congestion Information	Scheduling Granularity	Advantages	Disadvantages
TinyFlow [52]	2014	Local	Flow	Solving the head-of-line and hash collision problem	Reordering problem of long flows
WCMP [53]	2014	Global	Flow	Achieving better load balancing in asymmetric topologies	Inflexible rerouting and hashing collision problem
RepFlow [54]	2014	Oblivious	Flow	Obtaining lower average flow completion time and tail latency for short flows	Bring more overhead under heavy congestion
Expeditus [55]	2017	Global	Flow	Monitoring three-tier Clos topology timely	Need customized hardware
RPS [56]	2013	Oblivious	Packet	Making full use of all path resources	Frequent packet reordering
DRILL [57]	2017	Local	Packet	Fast response to congestion	Frequent packet reordering
QDAPS [58]	2018	Local queue information or global delay	Packet	Mitigating packet reordering problem	High computing overhead of switch
RMC [59]	2019	Global	Packet	Avoiding unnecessary fast retransmission and reduce flow completion time	Need delay measurement
CAPS [16]	2019	Global	Short flows: packet, long flows: packet or flow	Mitigating packet reordering problem	Increasing network redundance
OPER [60]	2019	Global	Packet	Alleviating packet reordering problem	Need supports of both ends
APS [15]	2020	Global	Short flows: packet, long flows: flow	Ensuring that the short flows meet their deadlines and maintaining high throughput for the long flows	Inaccurate distinguishment between long and short flows
CONGA [61]	2014	Global	Flowlet	Robust to asymmetric networks	High feedback delay and customized switch, leading to poor scalability
HULA [62]	2016	Local	Flowlet	Low overhead of forwarding table	Not a global optimal decision
LetFlow [63]	2017	Local	Flowlet	Suitable for asymmetric topologies	Random selection of paths
AG [64]	2019	Global	Adaptive granularity	Reducing the effect of disorder and ensures high link utilization	High measurement overhead
TLB [65]	2019	Local	Adaptive granularity	Adjusting the granularity of long flows path switching	Packet reordering in asymmetric topology

issue, MPTCP_OPN [51] adjusts the number of subflows flexibly according to the network status. Under light congestion, MPTCP_OPN uses more subflows to enhance link utilization. On the other hand, under the heavy congestion, MPTCP_OPN reduces the number of subflows to avoid timeout events caused by full window loss.

C. SWITCH-BASED LOAD BALANCING SCHEMES

The controller-based load balancing approaches are hard to react to short flows timely, while the host-based ones generally need to modify the TCP/IP stack, making it hard to deploy in a productive environment. Therefore, it is an excellent choice to sense congestion at the switch side and send traffic to different paths. In general, the switch-based schemes can be divided into four types according to its scheduling granularity: flow-based, packet-based, flowlet-based and adaptive-granularity-based ones. Table 3 summarizes the switch-based schemes which have been introduced currently.

1) FLOW-BASED

ECMP assigns each flow via static hash calculation, leading to the high latency for short flows and low throughput for long flows. To alleviate this issue, TinyFlow [52] partitions the long flows into 10KB short flows. To avoid the risk of packet reordering, the short flows are not split anymore. Then all the flows are scheduled in an ECMP manner. However, TinyFlow cannot adapt to the asymmetric network due to its unawareness of global congestion information.

To obtain better load balancing in the asymmetric scenario, WCMP [53] delivers fair per-flow bandwidth allocation according to the changing topology. WCMP relies on SDN running commodity switch to obtain traffic information and network topology. Then it assigns a different weight for each path based on the available link capacity and distributes traffic to different available paths according to the weight. However, utilizing flow granularity makes it hard to reschedule traffic timely via congestion information.

In data center networks, short flows are usually plagued by long flows, even though some paths are under-utilization. RepFlow [54] is a simple yet effective mechanism that duplicates each short flow and transmits each flow via ECMP to reduce the completion times, without any modifications to switches or host kernels. Unfortunately, it will aggravate the congestion level of the network.

In previous schemes, the ToR switch generally maintains the end-to-end congestion information for all paths from itself to other ToR switches. However, this kind of information feedback method is challenging to employ in a complex three-layer Clos topology. Expeditus [55] is a congestion-aware load balancing scheme designed for

three-tier Clos topology. The two main modules of Expeditus are local congestion monitoring and two-stage path selection. Each switch locally monitors the utilization of its uplinks, ensuring the real-time congestion information and low measurement overhead. Then it uses two-stage path selection to aggregate congestion information across different switches and make load balancing decisions.

2) PACKET-BASED

Flow-based load balancing approaches can avoid packet reordering, but easily suffer from the long-tailed delay. Moreover, since the long flows cannot switch among available paths flexibly, it leads to the low-utilization problem, increasing the flow completion time. To better utilize the link resources, a lot of load balancing schemes based on packet granularity are proposed.

RPS [56] is a simple packet-based scheme that randomly sprays data packets of a flow to all paths. Under the symmetry topology, RPS performs well due to the full bandwidth utilization. However, RPS leads to severe packet reordering, which reduces the congestion window size and throughput.

To quickly alleviate the congestion caused by burst traffic, DRILL[12] [57] proposes a load balancing strategy for microburst traffic. DRILL [57] makes load balancing decision based on the local switch information. To avoid many flows choose the least congested paths simultaneously, DRILL uses a method similar to the ''power of two choices'' paradigm. When choosing a new packet path, the path will be selected from the best path with the shortest queue length in the last round and two paths randomly chosen in this round. It does not need to change the protocol stack and hardware, making it easy to be deployed in production data centers. Nevertheless, the oblivious of global congestion makes it experience packet reordering inevitably under asymmetry topologies.

Since previous packet-level load balancers easily suffer from packet reordering under network asymmetry, QDAPS [58] selects paths for packets based on the queuing delay of the output buffer and makes the packet arrived earlier be forwarded before the later packets to avoid this problem. QDAPS consists of three modules. The first one is Queueing Delay Estimation, which estimates the end-to-end delay between leaf switches. The second one is Packet Reorder Resolution, which lets the earlier arrival packets be transmitted ahead of the later ones to alleviate packet reordering. The last module is Large Flow Rerouting. Because the later arrival packets wait longer than the earlier arrival ones, the large flow may suffer from long queueing delay, reducing its throughput. This module reroutes packets to the output port to address this problem with the shortest queue length when the current queue length exceeds a threshold. QDAPS needs to store the queueing delay and calculate the appropriate outport for each packet, leading to high computation overhead.

To better cope with packet reordering, RMC [59] firstly estimates the end-to-end delay based on the local queue length and global path latency. When the end-to-end delay of earlier packet is larger than the later one, the switch marks the later one as reordering packet via a reserved bit in the TCP header. To handle network asymmetry, RMC utilizes a network coding approach to reduce long-tailed flow completion time.

To alleviate the negative impact of short and long flows on each other in data center networks, CAPS [16] sprays the short flows' packets to all available paths, while the long ones are transmitted via ECMP to several paths. Moreover, CAPS adopts forward error correction (FEC) technology to deal with the packet reordering and head-of-line blocking problem of short flows. However, CAPS introduces a lot of redundant packets that exacerbate network congestion.

While network coding schemes can mitigate the out-of-order packet problem under network asymmetry, it will lead to extra traffic overhead, more considerable queueing delay. OPER [60] utilizes systematic coding to achieve self-adaptive load balancing in the data center network. When the network is less-congested, OPER utilizes redundant packets to alleviate packet reordering. On the contrary, the data packets can replace the switches' redundant packets to avoid larger queueing delay.

To deal with the resource competition problem between short and long flows, APS [15] sprays the packets of short and long flows on respective paths based on the ON/OFF traffic characteristics to alleviate the impact of this problem. If the short flows are in the OFF period, APS will scatter the long flows' packets to all paths to achieve high throughput. When the two kinds of flows are mixed, APS isolates short and long flows on different paths. For the short flows, they are scheduled on packet granularity to obtain high link utilization. The long flows are rerouted by ECMP to mitigate the out-of-order problem.

3) FLOWLET-BASED

Since flow-based and packet-based schemes either suffer from hash collision or severe packet reordering, they potentially cause low network utilization and performance degradation. To overcome this issue, a fine-grained load balancing called flowlet is proposed, which is a burst of packets from a flow that is separated via a sufficient time gap.

CONGA[13] [61] is a network-based distributed congestion-aware load balancing scheme which is designed to work for the two-tier Leaf-Spine topologies. It leverages the overlay technology to implement a leaf-to-leaf feedback loop to obtain the global status and switches flowlets with the global congestion information. Fig.6 illustrates the working procedure of CONGA.

Since CONGA is implemented in custom silicon on a switch, it needs a long time for hardware design. Besides,

[12]https://github.com/snowzjx/ns3-ecn-sharp/tree/master/src/drill-routing

[13]https://github.com/snowzjx/ns3-ecn-sharp/tree/master/src/conga-routing

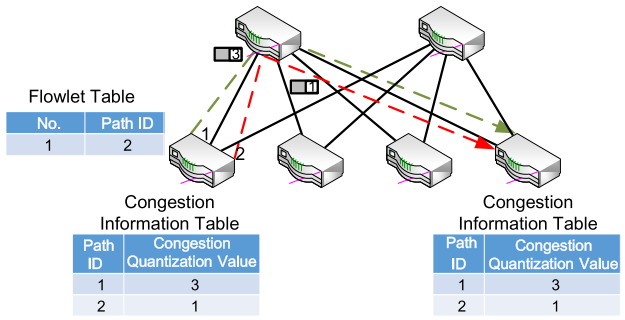


FIGURE 6. Working procedure of CONGA.

CONGA is made only for two-tier Leaf-Spine topologies, which hampers its scalability. To address these problems, HULA[14] [62] leverages the programmable switch and P4 language to make the algorithm can be modified as desired by the network operator. HULA only maintains the congestion information for the next best hop and picks up the next hop instead of the entire path.

To mitigate the overhead of path detection, LetFlow[15] [63] is a pretty simple load balancing scheme that randomly chooses a path for each flowlet. The size of a flowlet is elastic and vary according to the traffic conditions on different paths. On the slow paths, flowlets tend to be smaller due to high latency and low per-flow bandwidth, while flowlets get more extensive because the timeouts are less likely to trigger a new flowlet on the fast path. In this fashion, Letflow can significantly handle the network asymmetry and enhance overall performance. Nevertheless, it is hard to retain the best load balancing performance due to the random path selection.

4) ADAPTIVE-GRANULARITY-BASED

The flow-level approaches map each flow to one path, and the flowlet-level approaches only reroute flowlets when a new flowlet emerges. Both kinds of schemes easily suffer from low utilization of network resources due to their coarse switching granularities. The finer packet granularity is prone to profound packet reordering due to asymmetric topology, leading to performance degradation.

To better handle network asymmetry, AG [64] adaptively adjusts the switching granularity based on different degrees of topology asymmetry. AG is deployed on the switch side and has three main modules. Topology Asymmetry Measurement module measures the number of congested paths via the one-way delay to get asymmetric topology status. Packet Train Adjustment module is used to adjust the packet train's size, which is a series of packets belonging to one congestion widow in a flow. It increases the switching granularity to mitigate packet reordering under high degrees of topology asymmetry and decrease the granularity to achieve high bandwidth utilization otherwise. Packet Train Scheduling module randomly chooses a path for each packet train to avoid synchronization effect of herd behaviour.

Current load balancing schemes are generally agnostic to the long flows and reroute them via the same granularity. Therefore, the short flows suffer from long-tailed queueing latency, and the throughput of long flows reduces significantly. To address these problems, TLB [65] adaptively adjusts the switching granularity of long flows based on short ones' status. Under the heavy load of short flows, the long flows increase its granularity to leave more paths for short ones and decrease its granularity under the light load of short flows to achieve high throughput.

D. SUMMARY

In this section, we summarize current load balancing schemes as follows:

The centralized scheme based on the central controller obtains the global network status information. It can accurately perceive the link failure and allocate the optimal path for each flow. Moreover, the centralized schemes separate the switch's control plane from the data plane, realizing the network's flexible control. The switch only needs to perform basic forwarding operation according to the controller's flow table, which reduces the overhead of switch. However, obtaining and maintaining global information requires a certain amount of deployment overhead. Considerable feedback and control delays also degrade the load balancing performance under dynamic burst traffic. The computing ability of the load-balancing method based on a single central controller is quite limited. When multiple controllers are used to handle the traffic together, they are prone to synchronization problems.

Compared with the centralized scheduling scheme, the host-based load balancing scheme has better scalability. The host can modify the load balancing mechanism according to its own needs and is not affected by other nodes. Compared with the switch-based load balancing mechanism, the host-based one can perceive the end-to-end congestion status and make more accurate rerouting decisions. Nevertheless, the end-to-end feedback delay based on host-based load balancing is generally too large to adapt to the highly dynamic traffic, using at least one round-trip delay to perceive global congestion. Moreover, the host-based load balancing scheme usually needs to modify the host's protocol stack, making it challenging to deploy in a data center environment.

The switch-based load balancing mechanism is independent of the host's network stack, and it can immediately serve all traffic once deployed. It also can sense the load in the network in real-time. Therefore, the traffic can be quickly rerouted by the switch to achieve better load balancing performance. However, switch-based load balancing mechanism usually requires customized switch silicon, which is very expensive. Although programmable switches can implement load balancing algorithms prototypes, they also need specialized software switch to support complex load balancing algorithms.

[14]https://github.com/rachitnigam/Hula-hoop

[15]https://github.com/snowzjx/ns3-ecn-sharp/tree/master/src/letflow-routing

There are still some problems in the design process of the load balancing mechanism. The data center traffic can be divided into delay-sensitive small flows, and long flows requiring high throughput. Some load balancers are agnostic to the mixed traffic and schedule these flows without distinction. Therefore, the short flows suffer from the long-tailed queuing delay and reordering problems, while the throughput of long flows is also significantly reduced owing to low link utilization. To improve the performance of traffic transmission, it is necessary to distinguish them and optimize them separately. Besides, the size of the rerouting granularity also affects the load balancing performance. When the switching granularity is too large, it is easy to lead to insufficient network utilization. On the contrary, it will cause packet reordering, enormously increase the completion time of short flows. Therefore, adjusting the switching granularity according to the network status is a problem to be tackled.

V. TRANSPORT LAYER

Datacenter networks host a plethora of applications and services with different types of traffic pattern. Many applications' flows are delay-sensitive like MapReduce, web search and online retail, which have a stringent requirement for the delay. Meanwhile, the links of data center networks have ultra-high bandwidth and ultra-low propagation latency. The data center network is a large scale local area network, and the traffic transmission is mostly in the high concurrency mode. Therefore, network congestion characteristics are mainly reflected in the queuing and packet loss at the switch side. Traditional TCP is initially designed for the wide-area network transmission environment that utilizes packet loss as the congestion signal and tends to exhaust the switch buffer in exchange for higher network throughput. If traditional TCP is directly used in the data center network environment, it will deteriorate the flows' transmission performance, especially the short ones.

In order to avoid large queueing and packet loss, and to improve the flows' transmission performance, that is, to provide low-latency transmission for short flows (meeting their deadline requirements), while ensuring the high throughput of long ones, industry and academia have conducted a lot of researches [66]–[69]. The transmission mechanisms can be roughly divided into the following five categories.

A. DEADLINE-AGNOSTIC SCHEMES

The first category is deadline-agnostic. Table 4 presents deadline-agnostic transmission schemes introduced in current years.

The most representative scheme is DCTCP[16] [12]. DCTCP utilizes the information of each queue's length to mark packets. The arrival packet is marked via ECN when the queue length is larger than the preset marking threshold. The DCTCP source decreases the congestion window size via a factor that depends on the fraction of marked packets. The larger the fraction, the bigger the factor. In this fashion, DCTCP obtains small queueing delay and high throughput. Nevertheless, when long and short flows coexist in the switch buffer, the data packets of long flows are still likely to be queued ahead of the short ones. Therefore, the performance of the short flow is still not fully guaranteed.

To further reduce the queuing latency of short flows, ECN* [70] employs dequeue marking to accelerate the congestion information delivery for instant queue length based on ECN. Nevertheless, its packet marking is based on a fixed threshold, making little contribution to reducing the queuing delay.

In order to further improve the transmission performance of short flows and to make the sender actively treat long and short flows differently, L^2DCT[17] [71] schedules flows with the Shortest Remaining Processing Time (SRPT) mechanism and adjusts the congestion window according to the congestion level, which is similar to DCTCP. L^2DCT does not need the knowledge of the flow size in advance, does not require the hardware support and is convenient to implement in the Linux kernel.

The ECN mechanism can tell the switch whether the queue length exceeds a given threshold, but this measurement is coarse-grained. Therefore, to design more fine-grained congestion feedback mechanisms, the researchers introduce a series of mechanisms with delay as the congestion signal. DX [72] and TIMELY [73] leverage latency as the congestion signal to achieve ultra-low queueing with high utilization. To obtain accurate latency measurements, DX develops both software and hardware solutions to measure the network-side latency. It then exploits the queueing delay to decide whether to increase or decrease the window size in the next round in an Additive Increase Multiplicative Decrease (AIMD) fashion. To achieve better delay measurement, TIMELY [73] uses recent NIC to measure round trip time in data center networks with enough precision and utilizes the rate of RTT variation to predict the onset of congestion. In this manner, it keeps low latency and guarantees high throughput. Besides, the ON/OFF pattern of traffic in data center networks influences the increasing congestion window, easily leading to severe packet loss at the beginning of ON period under heavy congestion. To address this problem, TCP-TRIM [74] proposes a window inheritance mechanism, in which the congestion window size in the previous ON period is selectively reused. It also employs probe packets to detect the end-to-end delay to regulate the congestion window at the end host side to ensure the bottleneck link's high utilization.

Since ECN-based schemes fail to perceive the transient congestion and the delay-based schemes is oversensitive to congestion, making them less competitive when coexisting with the traditional TCP, FFC [75] combines these two-dimensional signals to achieve both fast convergence and TCP friendliness. The experimental results show that FFC converges fast and reduces the flow completion time significantly compared with DX and DCTCP. To control

[16] https://github.com/myasuda/DCTCP-Linux

[17] https://github.com/FujiZ/ns-3

TABLE 4. Comparison of different deadline-agnostic transmission mechanisms.

Name	Year	Congestion Signal	Transient Congestion	Modification Requirements	Advantages	Disadvantages
DCTCP [12]	2010	ECN	Not aware	Sender, Receiver	Keep the queue at a low level, thus greatly mitigating the packet losses and TCP timeouts	When long and short flows coexist together, short ones easily encounter head-of-line blocking problems
ECN* [70]	2012	ECN	Not aware	Sender, Receiver	Reduce the queuing delay of short flows	Fixed marking threshold easily incurs excessive packet losses and unfairness problems
L^2DCT [71]	2013	ECN	Not aware	Sender, Receiver	Short flows are given higher bandwidth than long ones, greatly reducing their flow completion time	Hard to differentiate short and long flows
DX [72]	2015	Delay	Aware	Sender, Receiver	Using packet delay as the congestion signal can effectively reduce flow completion time	Accurate packet delay is hard to obtain
TIMELY [73]	2015	Delay	Aware	Sender, Receiver	Utilizing delay at end-host is an efficient way to measure the real-time congestion extent	Need specialized NIC to gain precise delay
TCP-TRIM [74]	2017	Delay	Aware	Sender, Receiver	Using probe packets and delay-based congestion control ameliorates the transmission performance of concurrent HTTP traffic	Difficult to obtain precise packet latency
FFC [75]	2020	ECN+Delay	Aware	Sender, Receiver	Using ECN and RTT can achieve fine-grained congestion measurement, thus helps flow quickly converge to an appropriate sending rate	Need to modify the host kernel
DDT [76]	2019	ECN	Not aware	Switch	Achieving better fairness when ECN-enabled TCP flows coexist with traditional TCP flows	Correct value of dynamic threshold is hard to obtain
HPCC [77]	2019	Precise link load information	Aware	Sender, Receiver, Switch	Employing INT to guarantee ultra-low latency and high bandwidth utilization	INT technique is not universal, making it difficult to deploy on a large scale

TABLE 5. Comparison of different deadline-aware transmission mechanisms.

Name	Year	Congestion Signal	Transient Congestion	Modification Requirements	Advantages	Disadvantages
D^3 [78]	2011	None	Not aware	Sender, Receiver, Switch	Allocating higher bandwidth to urgent flows, thus increasing the meeting deadline ratio	Execution overhead is large
D^2TCP [79]	2012	ECN	Not aware	Sender, Receiver	Reducing the missed deadline ratio and guaranteeing the TCP friendliness	Conservative additive increase of congestion window makes a lot of short flows miss their deadlines
A^2DTCP [80]	2015	ECN	Not aware	Sender, Receiver	Improving the available bandwidth utilization and mitigating the missed deadline ratio	High missed deadline ratio under heavy congestion

the competitiveness of the coexisting TCPs, DDT [76] monitors the switch queue length in real-time and adjusts the ECN-making and packet-dropping thresholds dynamically. As shown in Fig.7, the switch queue maintains two thresholds, including the ECN marking threshold and the packet dropping threshold. The switch firstly categorizes active flows with different protocol types and counts the number of flows in each category. The two thresholds are then adaptively regulated linearly to achieve better fairness when ECN-enabled TCP flows coexist with traditional TCP flows.

With the prevalent of In-network telemetry (INT) technic in recent years, HPCC[18] [77] leverages the precise link load information obtained from INT to compute accurate flow rate. In this fashion, HPCC can make full use of the free bandwidth while avoiding congestion and maintain near-zero in-network queues for ultra-low latency. Nevertheless, HPCC brings about large implementation overhead like 8Bytes in each INT Metadata header.

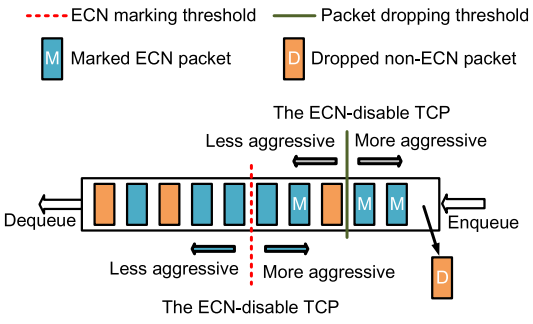


FIGURE 7. Working procedure of DDT.

[18]https://github.com/hpcc-systems/HPCC-Platform

B. DEADLINE-AWARE SCHEMES

Many services in data center employ Online Data Intensive applications (OLDI) such as web search, online retail, and advertisement. In order to maintain a good user experience, data center network operators have been looking for methods to reduce response times. The user request can be quickly broken into hundreds of flows on the data center network's back-end. To ensure good interactivity, these flows are allocated as different communication deadlines ranging from 10ms to 100ms [12]. If some flows miss the deadline, the aggregator (the server used to collect response or distribute requests) will not accept the data it carries, which will degrade the final result's accuracy. The deadline information can be obtained through the application layer [78]. Therefore, to guarantee the users' experience and the operator's revenue, many mechanisms optimize the transmission performance in a deadline-aware fashion and we summarize them in Table 5.

As the typical deadline-aware protocol, D^3 [78] exploits the application deadline information to allocate the network bandwidth. Specifically, applications expose the information of size and flow deadline at flow initiation time. Based on the information, end-hosts calculate the flows' rate before the transmission and deliver the rate to destination. Switches along the path to the destination assign each flow rate to greedily meet as many deadlines as possible and notify the sender of the granted rate via ACKs on the reverse path. Since the sender, switch and receiver of D^3 need modifications; the execution overhead is relatively large. Besides, D^3 allocates the rate to flows by their arrived order, indicating that the scheduling discipline of D^3 is FIFO. Therefore, some near-deadline flows arriving later might miss their deadlines.

To further reduce the execution overhead and meet the demand of deadline-aware flows, D^2TCP[19] [79] considers both the congestion control and deadline requirements via gracefully adjusting the extent of window reduction. It gives flows with small deadlines higher priority and transmits these flows firstly. D^2TCP adjusts the window size based on ECN feedback and deadline information. In this fashion, D^2TCP meets the deadline restriction of short flows and achieves high bandwidth for background flows. It also requires no modifications to the switch hardware and coexists with traditional TCP well.

Since the design of D^2TCP neglects the ultra-high bandwidth of data center networks, the conservative additive increase of congestion window causes flows to take an unnecessarily long time to obtain their maximum available bandwidth, making a lot of deadline-sensitive flows miss their deadlines. Moreover, when near-deadline and far-deadline flows coexist in bottleneck links, they are treated indiscriminately [79]. As a result, near-deadline flows can easily miss its deadline due to the switch's queue buildup. To cope with these issues, A^2DTCP [80] adjusts the congestion window increasing speed according to the network congestion extent and the probability of missing deadline. Through utilizing congestion avoidance with an adaptive increase speed, A^2DTCP speeds up the bandwidth detection, achieving low missed deadline ratio and high bandwidth utilization.

C. INCAST-AWARE SCHEMES

TCP incast is one of the significant problem in data center networks. It leads to catastrophic throughput degradation when multiple servers simultaneously send data to a single receiver with high bandwidth and low latency. The heavy traffic burst overflows the shallow switch buffer leading to packet loss. Therefore, excessive retransmission events result in under-utilization of the link bandwidth. To address this tough problem, a miscellaneous of protocols have been proposed. We list these schemes in Table 6.

TCP incast is one of the significant problems in data center networks. It leads to catastrophic throughput degradation when multiple servers simultaneously send data to a single receiver with high bandwidth and low latency. The heavy traffic burst overflows the shallow switch buffer leading to packet loss. Therefore, excessive retransmission events result in under-utilization of the link bandwidth. To address this challenging problem, a miscellaneous of protocols have been proposed. We list these schemes in Table 6.

Typically, the TCP minimum retransmission timeout (RTO) is set to 200ms by default on the Internet, while the propagation round-trip time in data center networks is less than 1 ms. As a result, when clients request data from multiple servers, flows that encounter RTO is delayed by 200ms in a data center network, resulting in TCP throughput collapse. FGRTO [81] finds that eliminating minimum RTO and enabling finer-grained RTO can successfully avoid TCP incast collapse in data center applications. This approach is practical and useful in the data center environment. However, this solution may trigger a large number of spurious timeouts and unnecessary retransmissions, which wastes available bandwidth and reduces transmission performance. Some other schemes also try to mitigate TCP incast problem by modifying TCP parameters, like reducing duplicate ACK threshold. Even though these schemes could mitigate the negative impact of timeout events, they still cannot fundamentally solve the TCP incast problem.

Since the receiver side knows the throughput of all TCP connections and the available bandwidth well, ICTCP[20] [18] performs incast congestion avoidance at the receiver side. The main goal of ICTCP is to reduce the packet loss before incast congestion rather than recovery after loss. ICTCP uses the available bandwidth on the network interface to control the window increase of all flows and adjusts the receive window based on the ratio of the difference between the measured and expected per-connection throughput over the expected throughput. When the ratio is small, ICTCP increases the size of its receive window. Otherwise, the size of receive window decreases. Another receiver-driven congestion control scheme is PAC [82]. To tame TCP incast congestion,

[19] https://github.com/ycaoae/D2TCP_Kernel

[20] https://github.com/Niwedita17/Implementation-of-ICTCP-in-ns3

TABLE 6. Comparison of different incast-aware transmission mechanisms.

Name	Year	Congestion Control Fashion	Switch Support	Modification Requirements	Advantages	Disadvantages
FGRTO [81]	2009	Window-based	No	Sender, Receiver	Reducing minimum RTO is easy to deploy in data center	Cannot tackle TCP incast problem fundamentally
ICTCP [18]	2013	Window-based	No	Receiver	Adjusting the receiving window on the receiver side to throttle aggregate throughput, thereby reducing incast congestion	Cannot avoid the micro-burst
PAC [82]	2014	Window/recovery-based	Yes	Receiver	Altering the sending rate of the senders through controlling the ACKs at the receiver to alleviate incast problem	Need the support of switch and cannot avoid the micro-burst
ARS [83]	2016	Window-based	No	Aggregator	Regulating the number of concurrent TCP flows based on the congestion status obtained from the transport layer and thus greatly mitigating incast	Tough to get accurate optimal batch size
PS [84]	2020	Window-based	Yes	Switch	Utilizing packet slicing can effectively reduce the number of timeout events and require no modification of TCP protocols	Precise incast probability is difficult to acquire
AP [85]	2017	Window-based	No	Sender	Adjusting the burstiness via the flow concurrency to reduce the incast problem without modifying the switch	The TCP stack of the sender needs to be modified and the accurate delay is hard to obtain
PLATO [86]	2014	Window-based	Yes	Sender, Receiver	Using a packet labeling scheme to aid the sender to avoid frequent timeouts, thereby reducing the TCP incast problem	Requiring modification on the switch
LTTP [87]	2014	Window-based	Yes	Sender, Receiver	Decreasing the number of timeout events through employing UDP and coding technic	Transmitting lots of redundant packets, and therefore reducing bandwidth utilization

PAC considers ACK not only as the acknowledgement for received packets but also the trigger for new packets. PAC intercepts and releases ACKs so that ACK-triggered in-flight traffic can fully utilize the bottleneck link without running into incast congestion collapse. Nevertheless, PAC relies on the aid of switch to get accurate congestion status through ECN feedback.

In contrast with the enhanced TCP protocols focusing on the congestion window adjustment, PS [84] finds that controlling the IP packet size decreases the incast probability much more effectively than controlling the congestion windows under severe congestion. Inspired by this investigation, PS adjusts IP packet size on widely used COTS switches. PS design makes a tradeoff between reducing incast probability and the cost of increasing header overhead. The design utilizes standard ICMP signalling, which requires no TCP stack changes and is compatible with a variety of data center transport control protocols. To better handle micro-burst, AP [85] finds that mitigating packet burstiness is an efficient fashion to reduce the incast probability. AP adjusts the burstiness and the time interval between consequent packets according to the flow concurrency dynamically. AP can keep compatibility on existing transport layer protocols without any changes on hardware switch.

Virtual Machines (VMs) allow users to run applications in different operating systems and configurations in the current data centre. However, it is tough to upgrade the VM's TCP/IP stacks to cope with TCP incast. Therefore, PLATO [86] utilizes a packet labelling system on the switch to improve the loss detection capabilities of TCP NewReno. When PLATO used with Weighted Random Early Detection (WRED) functionality at the switch, it allows TCP to detect packet loss via three duplicate ACKs, instead of waiting for a long RTO period, thus avoiding the throughput collapse. Furthermore, LTTP [87] utilizes UDP instead of TCP for data transmission since TCP's timeout is the root reason for TCP incast. To achieve reliable packet delivery, LTTP employs digital fountain code technique. LTTP also adopts TCP Friendly Rate Control (TFRC) to alter the data sending rates and maintain bandwidth utilization at servers. However, LTTP inevitably transmits lots of redundant packets, which degrades bandwidth utilization.

D. OUTCAST-AWARE SCHEMES

In the commodity data center networks with multi-rooted tree topologies, the close TCP flows are easily suppressed by the far ones, resulting in low throughput and large tail flow completion time [19], [88]. Fig.8 shows a TCP outcast scenario in which multiple servers simultaneously send their TCP flows to a single aggregator node via multi-hop paths. A small number flows (i.e., close flow) are sent from the server under the local ToR switch, while a great number of flows (i.e., far flow) are sent from the servers under other ToRs. The close and far flow sets arrive at corresponding input ports of the bottleneck switch and send to the aggregator node's output port. Due to the difference in the number of close

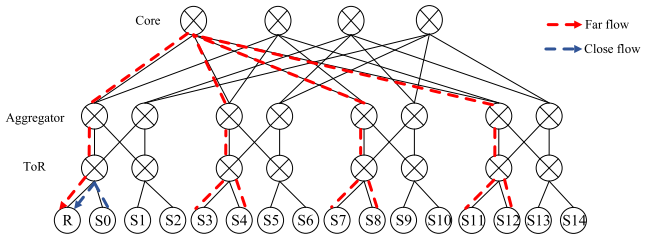


FIGURE 8. Fat-tree topology.

and far flows, the probability that the far-flow occupies the output port is larger, while the close flows have to experience more drop-tail packet losses, thereby reducing throughput. To mitigate TCP outcast problem, a series of schemes have been introduced in recent years. We can roughly divide the existing mechanisms into two categories: switch-based and host-based.

Generally speaking, the switch-based scheme is the most direct way to achieve throughput fairness. SAB [89] allocates the same advertisement window for all TCP flows according to the switch's buffer size. Based on the buffer size's exact congestion status, SAB obtains a higher link utilization rate without incurring packet loss. To reduce the high latency of stalled flow caused by timeout events, CP [90] discards the payload of a packet on an overloaded switch and accurately informs senders to use fast retransmission to recover lost packet.

Although switch-based scheme provides fast and accurate fairness control, it requires considerable system modifications, making it difficult to deploy at a large scale. Compared with the switch-based solution, the sender-based approach has the advantages of low cost and high flexibility. TCP Pacing distributes packets across entire RTT to avoid bursty packet loss at the sender side. This method alleviates TCP outcast to a certain degree, but still cannot avoid the buffer overflow [19].

TFC [91] employs token flow control, window acquisition phase and packet delay function to achieve high link utilization, ultra-low latency, fast convergence, and rare packets dropping. However, it is very complicated and requires modification of the sender, receiver and switch. To handle TCP outcast gracefully, OFTCP [19] adjusts advertisement windows of ACK packets to ensure fair bandwidth sharing of concurrent flows at the receiver side. In addition, it also randomly defers the sending time of flows based on real-time network status, mitigating the micro-burst under high flow concurrency.

E. TASK-AWARE SCHEMES

The data center hosts a variety of applications and is committed to a high-quality user experience with low latency [92]. Modern data centers employ the tree-based, divide-and-conquer algorithms to improve responsiveness, where data or computation is distributed across thousands of servers. In this fashion, the workload of each server is reduced, thus reducing the user waiting time. However, traditional transport protocols and scheduling schemes focus only on optimizing flow-level metrics, such as flow completion times and deadline missing ratio. A task consists of hundreds of flows, all of which should complete their transmissions before a task is considered complete. Existing flow-based mechanisms treat all flows in isolation, compromising the user experience due to stalled flows. Thus, to guarantee the transmission efficiency of tasks, a lot of mechanisms have been proposed, and we sum up these approaches in Table 7.

The most typical work from the view of the task is Coflow [20], which proposes a networking abstraction named coflow to express the communication requirements of parallel programming paradigms. Each coflow is several flows between two groups of machines with associated semantics and a collective objective. The semantics enable the network to take different actions on the collection to achieve the mutual end goal.

To accelerate the coflow transmission, Orchestra [93] schedules coflows via a FIFO manner with the central controller. The centralized coflow-aware mechanism needs extensive feedback and control delays that might negatively impact tiny tasks' overall execution time. To solve this problem, Baraat [94] is a decentralized task-aware scheduler based on FIFO without requiring prior information of flows. However, Baraat requires switch modification, making it hard to deploy in practice. OPTAS [98] is proposed as a lightweight scheduling scheme for tiny tasks without changes to applications or switches. OPTAS identifies small tasks via monitoring system calls and buffer footprints. To avoid head-of-line blocking, tiny tasks are assigned higher priorities than the large ones. The congestion window size and ACK delay time are calculated at the receiver side to transmit tiny tasks in a FIFO fashion. To reduce communication time of data-intensive jobs and ensure predictable communication time, Varys [95] uses Smallest-Effective-Bottleneck-First (SEBF) scheme that schedules a coflow based on its bottleneck's completion time greedily. Then it utilizes the Minimum-Allocation-for-Desired-Duration (MADD) algorithm to allocate rates to their flows. In this fashion, Varys decreases the average coflow completion time without starving any coflow and obtains high network utilization. To accelerate the coflow completion time, routing and scheduling must be jointly considered at the level of a coflow rather than individual flows instead of considering scheduling only. RAPIER [97] integrates routing and scheduling to optimize the coflow transmission in the data-intensive networks. Based on the priori coflow information (i.e., coflow size), RAPIER determines which paths to carry these flows, when to start sending them, and at what rate to serve them in a centralized manner. However, Varys and RAPIER require a lot of prior knowledge, which is an ideal scheduling mechanism and cannot be easily achieved. Aalo [96] improves Varys as a nonclairvoyant scheduler in that prior knowledge about the flow size is not required in advance. Even though these

TABLE 7. Comparison of different task-aware transmission mechanisms.

Name	Year	Distributed	Required Coflow Information	Application Modification	Advantages	Disadvantages
Orchestra [93]	2011	No	Yes	Yes	Utilizing a global management architecture and a series of algorithms to reduce average transmission time	Centralized control requires large delay
Baraat [94]	2014	Yes	No	Yes	Effectively dealing with heavy tasks via altering the level of multiplexing and thus providing benefits for a set of workloads	Need to modify switch, making it hard to deploy in practice
OPTAS [98]	2016	Yes	Yes	No	Greatly decreasing the transmission time of tiny coflows	Priori coflow knowledge is hard to get
Varys [95]	2014	No	Yes	Yes	Reducing the average coflow completion time and guaranteeing high network utilization	Acquiring prior knowledge
RAPIER [97]	2015	No	Yes	Yes	Decreasing coflow completion time to a great extent via combining routing and scheduling	Hard to apply on a large scale due to requiring a lot of prior knowledge
Aalo [96]	2015	No	No	Yes	Mitigating coflow completion time and ensuring high performance under high dynamics	The scheduler only makes its own decisions based on local information instead of the whole task, increasing the possibility of performance degradation
CODA [99]	2016	No	No	No	Identifying coflow without modifying applications	The model obtained by machine learning is hard to adapt to different network environments
TaTCP [101]	2019	Yes	Yes	Yes	Reducing the task completion time via a receiver-driven fashion	Need to pass information to the kernel manually

schemes can mitigate coflow completion time to some extent, they fall short in isolating coflows and providing predictable performance. To address these problems, Coflex [100] allows network operators to specify the desired level of isolation guarantee by exposing a tunable fairness knob. To reduce the CCT, Coflex employs the smallest effective-bandwidth-first (SEBF) approach that preferentially schedules a coflow with the smallest bottleneck's completion time.

Previous coflow-based solutions generally count on modifying applications to extract coflow information, making them hard to deploy in practice. CODA [99] is the first work that uses machine learning to recognize different coflows without any applications modifications. CODA employs an incremental clustering algorithm to perform application-transparent coflow identification. To alleviate identification errors, CODA also employs an error-tolerant coflow scheduler. The results show that CODA achieves over 90% identification accuracy.

Aforementioned task-based schemes are either complicated or unable to share task information, leading to insufficient scalability and performance degradation. Reference [101] finds that giving up the bandwidth of leading flows to the stalled ones effectively reduces the task completion time. Based on this investigation, TaTCP [101] effectively decreases the task completion time by sharing the flow-tardiness information through receiver-driven coordination. Moreover, this approach can be easily integrated with the most advanced TCP protocol designed for data centers without modifying the switch.

F. SUMMARY

An array of enhanced TCP protocols and scheduling algorithm have been introduced to mitigate network latency in data centers. We summarize the recent mechanisms in the transport layer as following:

The first category of data center TCP variants is delay-agnostic, which can be divided into ECN-based, delay-based, and mix-based. Since monitoring packet queue length at the switch side can well reflect the extent of the real-time network congestion, the ECN-based TCP protocols have attracted much attention recently. Nevertheless, all the ECN-based schemes are easy to neglect transient congestions because of their coarse-grained congestion measurement. Delay-based schemes utilize the packet delay measured on the host to obtain the real-time congestion extent. However, they are too sensitive to congestion, making them uncompetitive when coexisting with traditional TCP. Mix-based schemes combine the ECN and delay as the congestion signal. ECN signal effectively prevents packet loss, while delay signal can better control end-to-end queuing latency [75]. Thus, mix-based schemes can meet the demand of low flow completion time and high network throughput.

A series of applications operate under soft-real-time restrictions (for example, 300 ms latency), which means the deadline for network communication within the applications. To meet the flows' deadline, deadline-aware protocols are introduced, significantly improving users' experience. Some mechanisms are based on the application deadline information, while some alter the TCP's window adjustment method.

To tackle the severe incast problem, some schemes attempt to adjust the TCP parameters. Nonetheless, these methods cannot fundamentally address incast problem. Some approaches try to deal with incast issue at the receiver side, and some adjust the number of concurrent flows or mitigate packet burstiness to cope with incast.

Traditional transport control protocols generally work at the flow-level and ignore the fact that a task contains many flows, and all of these flows should finish their transmission before the task is complete. Therefore, flow-level schemes cannot provide good performance for the task-based application. For better task transmission, several schemes based on a central controller to obtain the task information. However, central-based approaches take a long period to gather information, increasing the tasks' execution time. Then some decentralized heuristic mechanisms emerge. Heuristic ones need a lot of prior task information, which is hard to get in reality. To solve this problem, machine-learning-based schemes are introduced to guarantee large-scale deployment. However, machine-learning approaches are quite complicated and challenging to adapt to all kinds of network environments.

Although a range of transport mechanisms have been introduced, much research is still needed in designing mechanisms to decrease FCT considering the coexistence of short and long flows as they compete for network resources. Moreover, there exists a congestion mismatch problem between the transport layer and network layer. Current congestion control algorithm only adjusts the rate according to the congestion status of the current path. However, when the load balancing scheme performs route switching, the new path's status may be inconsistent with the current path, easily leading to the mismatch between the sending rate and the path congestion status. For example, when rerouting to an idle path, the low transmission rate will lead to low link utilization; while rerouting to a congested path, high sending rate will further aggravate network congestion. Thus, it is essential to coordinate the transport layer and network layer to improve the overall performance.

VI. APPLICATION LAYER

With the rise of cloud computing technology, data centers are utilized as the modern computing infrastructure. Recently, a range of applications is broadly deployed in modern data centers. In the application layer, key-value stores is a kind of typical applications.

A. KEY-VALUE STORE REQUEST

The key-value stores have become an essential element of large scale applications, like web search, recommender systems, advertising and so forth. A single request may generate key-value operations across tens or hundreds of servers [13], [102], [103]. The tail latency of the key-value access operations dramatically influences the response time of the request.

The framework of current key-value stores is shown as Fig.9. In the large-scale distributed key-value stores, data is usually chopped into small parts, which are duplicated and distributed among different replica servers to achieve parallel access. Since each key-value access operation's service time straightly affects the tail latency of all key-value access operations triggered by the end-user, selecting the best replica server for each key-value access operation on the client becomes very important.

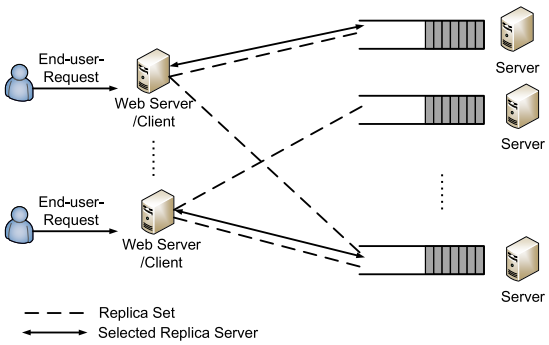


FIGURE 9. Replica selection in key-value stores.

There exist a series of challenges when designing the replica selection algorithms. Firstly, the replica selection algorithm should be uncomplicated enough to compute and coordinate in the key-value stores. Otherwise, clients cannot timely obtain the information of each other and replica servers. Besides, due to the high concurrency of key-value stores, it is hard to obtain the server-side waiting time and service time of key-value access operations. Additionally, since many aspects influence the servers' performance, the service time of key-value access operations is variable with time. Moreover, due to the effect of herd behavior [13], [14], majority of clients prefer to choose the fast servers and the high concurrency results in sharply performance degradation.

Currently, the replica selection algorithms can be roughly divided into three categories based on the needed information: information-agnostic, client-independence and feedback [13].

1) INFORMATION-AGNOSTIC

Information-agnostic algorithms do not rely on any other information. There are three types of the algorithm in this category: fixed, random and round-robin. OpenStack Swift [104] utilizes fixed mechanism. The client selects a fixed replica server for key-value access operations. Only when the fixed one is not available, the other servers are used. In the random algorithm, the client stochastically selects a replica server for each key-value access operation service. In the round-robin algorithm, the client sequentially chooses the server.

2) CLIENT-INDEPENDENCE

Client-independence algorithms only use the information measured by the client that mainly include RTT, RPT and OSK. RTT is the round trip time of the network. RPT means each key-value access operation's response time, which consists of RTT, the service time and waiting time at the

server. OSK indicates the outstanding key-value access operations that have been delivered to the replica server but have not yet received the corresponding value.

These information can be used through three ways: fixed, probability and hybrid. For the fixed fashion, Riak [14] selects the server with the least number of the outstanding key-value access operations. The second method is to determine the likelihood of selecting each replica server based on the measured information. For the hybrid manner, several replica servers are randomly chosen by the client first. Then based on the above network status, the client selects one of these replica servers. For example, MongoDB chooses the several servers based on the RTT and arbitrarily selects one of them.

3) FEEDBACK

Information-agnostic and client-independence schemes do not consider servers' time-varying performance, making them hard to choose the fastest replica server. To better tackle servers' variation and herd behaviour problems, various schemes based on feedback information have been proposed recently. The most representative example is C3 [14]. Fig.10 shows the architecture of C3. C3 monitors the waiting keys' queue length and its service time at the server-side and piggyback this information to the client. Then C3 ranks the available replica servers via the following function. When a key is going to be sent, The replica server with the smallest ψ_s is chosen by the Replica Selection scheduler.

$$\psi_s = \overline{R}_s - \overline{T}_s + \overline{q}_s{}^3 * \overline{T}_s \tag{1}$$

where $\overline{R}_s$ is the Exponentially Weighted Moving Averages (EWMA) of the response times. $\overline{T}_s$ is the EWMA of the piggybacked service time, and thus $\overline{R}_s - \overline{T}_s$ is the delay. $\overline{q}_s$ is the queue size estimation of the waiting keys, which is calculated by

$$\overline{q}_s \stackrel{\Delta}{=} 1 + q_s + n * OSK$$

where q_s is the EWMA of feedback queue size and n is the number of clients.

C3 also utilizes feedback information to control the sending rate. The detailed rate control scheme is borrowed from CUBIC. After the ranking process, the client inquires whether the sending rate is limited. If limited, the subsequent server with larger ψ_s will be inquired. Otherwise, the inquiry stops and the corresponding replica server is chosen to send the key. If the replica servers are all limited, the key is put into the backlog queue until there is at least one server which is not limited again.

Although C3 can reduce the tail latency of key-value access operations, it is too complicated, increasing communication overhead. To reduce the overhead, L2 [13] firstly chooses a subset of the replica servers with the smallest OSK. Then it selects the replica server with the shortest response time among this subset. L2 does not need global feedback information and achieves similar performance compared with C3.

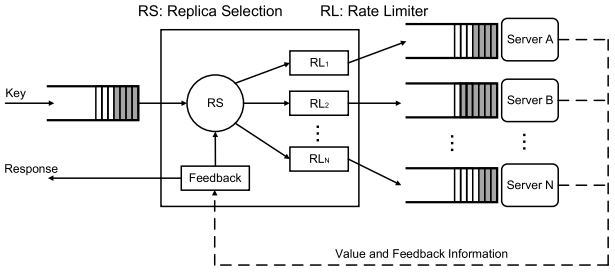


FIGURE 10. C3 architecture.

Besides the high overhead, C3 also suffers from the timeliness issue of the feedback information. Literature [104] shows that 1) Due to the traffic model, the feedback information may not be used for a long time and become outdated. 2) Since the key may need to wait a long time on a heavy-load or low-performance server, the corresponding client cannot receive the server's feedback information in time. Therefore, the Timeliness-aware prediction-based (TAP) algorithm predicts the replica servers' status based on all the previous received queue-sizes when the feedback information is not timely. TAP substitutes the EWMA of the piggybacked queue-size by the predicted value and removes the distributed rate control module of C3. In this way, TAP can alleviate the timeliness problem of C3 to some extents. However, the prediction accuracy cannot be guaranteed under various network conditions, leading to small performance improvement.

B. SUMMARY

We summarize the work in the application layer as following:

For replica selection algorithms, information-agnostic algorithms select replica server in a fixed, random or round-robin fashion. This type of algorithms is simple enough and easy to deploy. However, due to the neglect of network information, it is difficult to choose the fastest replica server, leading to increased latency. Client-independence algorithms use the client's information to select the replica server, and therefore mitigate the herd behaviour and cut off the tail latency compared with information-agnostic ones. Nonetheless, client-independence algorithms are unaware of time-varying performance, making them unable to make optimal decisions. To adapt with the performance and load fluctuations at replica servers, feedback algorithms make the replica selection via piggybacking the replica servers' information within the returned ''value'' in each key-value access operation. With the aid of the feedback information, feedback algorithms can significantly reduce the tail latency. However, the feedback information cannot be utilized for a long time due to the traffic dynamic. It makes their performance significantly worse than the ideal algorithm ORA (Oracle), which assumes the instantaneous status of replica servers. How to fill this performance gap between current feedback-based mechanisms and ORA is a problem.

VII. CROSS-LAYER DESIGN

Nowadays, the realization of some mechanisms does not rely solely on the information on one specific layer but requires

multiple layers of collaboration. In this section, we present some cross-layer designs in detail.

DeTail [105] is a cross-layer mechanism that takes packet priority and loads balancing into consideration. DeTail utilizes link-layer information to mitigate packet loss and performs load balancing at packet granularity in the network layer to reduce tailing latency. In the transport layer, the fast recovery and retransmission mechanism of TCP protocol is turned off to cope with spurious retransmission events caused by packet reordering. In the application layer, the priority of delay-sensitive flow is enhanced to guarantee its performance.

To provide near-optimal flow completion time at the 99th percentile for short flows and minimize average flow completion time for long flows, pFabric [106] approximates the Shortest Remaining Processing Time (SRPT) policy and decouples flow scheduling from rate control. pFabric consists of three main modules: flow prioritization, pFabric switch, and rate control [107]. For flow prioritization, each packet carries a single number in its header that encodes its priority. The flow priority is set to the remaining flow size to be transmitted. The pFabric switch utilizes two simple mechanisms in the network layer: priority scheduling and priority dropping. Whenever a port is idle, the highest priority packets buffered on the port are queued and sent out. Whenever a packet reaches a port with a full buffer, it is discarded if its priority is less than or equal to the buffer's lowest priority packet. Otherwise, the lowest priority packet is discarded to make room for new packets. In order to avoid throughput collapse, pFabric employs a simple rate control mechanism in the transport layer, which inherits the congestion control mechanism of TCP with some modifications, such as adjusting the initial congestion window size, disabling fast retransmission, using Selective ACKs (SACKs) and fixed number of successive timeouts to detect network congestion.

To obtain near-optimal completion time for short flows and high flow throughput in a wide range of scenarios, including incast, NDP [24] uses a packet-granularity forwarding method in the network layer to evenly distribute flows to all parallel paths for transmission to accelerate the transmission speed. The NDP sender randomly sorts all available paths, and then sends a data packet on each path in random order. After sending a round of data, it reorders the paths, effectively avoiding multiple senders from choosing the same path for transmission at the same time. NDP adopts the method of cutting payload to prevent packet loss. When the switch's queue length exceeds a certain threshold (such as 8 data packets), the payload of the data packet is removed, effectively reducing the queueing delay. Compared with the regular packet, the packet header has a higher dequeue priority, which drives fast retransmission and effectively reduces the retransmission delay. Besides, NDP is a receiver driven scheme. The receiver drives the pull packets according to received packets in the transportation layer. The sender sends or retransmits packets according to pull packets, which can ensure the sending rate of packets matches the receiver end's link rate and thus achieve high throughput.

To cope with TCP incast, ARS [83] dynamically adjusts the number of concurrent TCP flows by batching application requests based on the congestion status acquired from the transport layer. Besides, ARS is deployed only at the aggregator-side without making any modifications on numerous workers. The main idea of ARS is to divide a round of requests into several batches. Specifically, after sending a batch of requests, the aggregator will not issue a new batch of requests until all the requested data in the current batch is successfully received. The next batch's size relies on the congestion status obtained from the transport layer.

To reduce the FCT of time-sensitive flows and help avoid throughput-collapse situations, literature [108] proposes a light-weight cross-layer approach named T-RACKs. It implements a shim layer between the virtual machines layer and the end-host NIC to bridge the gap between huge RTO of TCP and the actual round trip times experienced in the data center.

VIII. FUTURE DEVELOPMENT TRENDS AND PROSPECTS

In the previous sections, we have summarized a series of existing works in the data centre's various layers. Now we look ahead to the future trends of data center networking.

(1) RDMA has recently been widely used in data center networks, which can greatly reduce the transmission delay and CPU overhead. There are roughly three types of RDMA networks, named Infiniband, RoCE, and iWARP. Among them, Infiniband is a network designed for RDMA, which guarantees reliable transmission from the hardware level, while RoCE and iWARP are both Ethernet-based RDMA technologies. Infiniband networks possess the best performance, but customized network cards and switches are expensive, making them difficult to deploy on a large scale. In comparison, RoCE and iWARP only need to use a particular network card, and the price is relatively lower. However, their performances are inferior compared with Infiniband. Therefore, proposing a better network architecture that considers the compromise between performance and price is a research direction for RDMA network in the future. Besides, a majority of RDMA network architectures rely on PFC mechanism. Nevertheless, PFC exists a variety of drawbacks, such as unfairness and head-of-line blocking problem, significantly decreasing the throughput and overall performance. Some schemes, like [27], substitute PFC with special hardware (FPGA) that are impractical to deploy in large-scale networks. Thus, it is essential to find a scheme to alleviate the RDMA's reliance on PFC. Moreover, RDMA can combine with some new transport protocols, e.g., QUIC [109], to reduce the network delay via bypassing the host's kernel.

(2) The data center network has the characteristics of high bandwidth and low latency. Nevertheless, the calculation cycles, the number of CPU cycles needed to process each

TABLE 8. Summary of schemes in data center networks.

Name	Author and year	Objectives	Methods	Findings
QCN [4]	IEEE 802.1, 2010	To improve classical Ethernet performance and provide low latency in data center networks	QCN consists of two modules: Congestion Point (CP) and Reaction Point (RP). CP samples the incoming data packets with a probability in the switch to decide whether the network is congested or not, and RP is used to adjust the sending rate at the source side based on the quantized congestion feedback value carried in the feedback packet.	QCN controls link rates to tackle congestions in several round trip times, but it suffers from some issues, like frequent queue oscillation and unfairness.
DCQCN [7]	Zhu YB et al., 2015	To meet the demands of delay-sensitive flows and throughput-intensive flows in data center networks	The ECN-based switch marks packets when the queue length surpasses the preset marking threshold, and the receiver periodically informs the sender by a signal packet. Then the sender employs explicit rate control and alters the flow's rate according to the received packets.	DCQCN provides fast convergence and achieves high link utilization.
DCQCN+ [25]	Gao YX et al., 2018	To suppress the large-scale incast problem	DCQCN+ adjusts the period for rate increase dynamically according to the scale of incast event.	DCQCN+ can handle incast congestion of at least 2,000 flows both in simulation and testbed, and achieve lower latency.
MP-RDMA [27]	Lu YW et al., 2018	To enable multiple paths' usage in RDMA	MP-RDMA employs an ECN-based multi-path ACK-clocking mechanism to select the best transmission path for RDMA packets.	MP-RDMA can significantly improve the overall network utilization and reduce average flow completion time compared with single-path RDMA.
Hedera [39]	Al-Fares M et al., 2010	To address the long flow collisions problem of ECMP	Utilize the global information to schedule long flows to less-congested paths.	A central scheduler with global knowledge can significantly outperform the hash-based ECMP load-balancing.
MicroTE [40]	Benson T et al., 2014	To reduce the completion time of short flows with high bursty	Utilize a centralized controller to monitor predictable traffic and schedule them firstly.	MicroTE outperforms ECMP in all 1-second intervals with real data center traffic traces, but it performs poorly when the traffic is less predictable.
Mahout [41]	Curtis AR et al., 2011	To alleviate the high overhead of large flows detection in Hedera	Detect the elephant flow at end-host, and use a shim layer to mark some packets when the buffer exceeds a threshold. After receiving a marked packet, the controller will select a least-congested path for it.	Mahout obtains higher throughput than ECMP with different thresholds and small overhead since it removes the detection process to end-hosts.

packet, lag behind the available network bandwidth. Thus, the CPU processing time is increasingly becoming a detrimental factor for increasing packet delay in high-speed data centers. In general, this can be handled by moving some network functions to the network interface cards, for example, TCP Segment Offloading

TABLE 8. *(Continued.)* **Summary of schemes in data center networks.**

FastPass [42]	Perry J et al., 2014	To achieve zero-queuing in data center networks	Employ a centralized arbiter to decide when each packet should be transmitted and which path it should follow.	FastPass can decrease queue length and flow completion time, and gain similar high throughput as baseline TCP.
FDALB [43]	Wang S et al., 2016	To adapt to traffic dynamics and alleviate the controller overhead	The long and short flows are distinguished according to the flow size distribution on the host side. The switch determines the short flow path, and the global congestion-aware central controller determines the path of the long flow.	FDALB reduces the average FCT of flows compared with ECMP while achieves higher scalability than Mahout.
Freeway [44]	Wang W et al., 2016	To meet the different requirements of short flows and long flows	Schedule the long flows with the centralized controller and short flows with ECMP to different paths.	Freeway obtains higher throughput and lower latency than ECMP.
SAPS [45]	Irteza SM et al., 2018	To make packet spraying effective under asymmetry while maintaining its simplicity	Use the central controller to detect link failures and establish a symmetrical virtual topology, so that each flow can be sprayed in the virtual topology.	SAPS achieves lower flow completion time than state-of-the-art load balancing schemes over various application workloads and asymmetric network scenarios.
SOFIA [46]	De Pellegrini F et al., 2018	To mitigate the overhead of the controller	The controller uses an online learning algorithm to calculate the optimal threshold for distinguishing long and short flows according to the flow size distribution.	The implementation in a real OpenFlow controller demonstrates the viability of SOFIA as a solution in production environments.
FlowBender [47]	Kabbani A et al.,2014	To address the hash collisions of ECMP	FlowBender reroutes flows if they are detected to experience congestion by modifying the hash function.	FlowBender can significantly improve the throughput, reduce the flow completion time than ECMP.
CLOVE [48]	Katta N et al., 2016	To avoid hardware and end-host stack modifications and react to dynamic traffic quickly	CLOVE applies standard ECMP in the physical network and modifies the packet header at software switches to instantly control how switches assign data packets.	CLOVE reduces flow completion time and avoid packet reordering.
Hermes [33]	Zhang H et al., 2017	To handle network uncertainties like failures and mitigate packet reordering	Hermes uses RTT, ECN and probing packets to sense uncertainties and make rerouting decisions.	Hermes can tackle uncertainties like asymmetries and achieve better flow completion time (FCT) than CONGA and CLOVE.
MMPTCP [50]	Kheirkhah M et al., 2019	To avoid long queuing latency and packet loss	In the initial stage, the data packets are randomly scattered to all available paths. When the sent bytes exceed the threshold, the traffic transmission is switched to MPTCP mode.	MMPTCP achieves high throughput for long flows, low latency for short flows, and fair co-existence with legacy transport protocols.

and General Receiving Offloading. However, these technologies cannot support complex strategies. To tackle this problem, SmartNICs have been introduced as a new platform in data center networks, making it possible for network operators to offload complex network strategies flexibly. Therefore, how to better use and design SmartNICs in the data center is a future research direction.

TABLE 8. *(Continued.)* **Summary of schemes in data center networks.**

MPTCP_OPN [51]	Li WH et al., 2020	To tackle bandwidth wastage and timeout events caused by a fixed number of subflows	MPTCP_OPN alters the number of subflows in real-time according to the network packet loss rate.	MPTCP_OPN effectively reduces the timeout probability caused by full window loss and flow completion time by up to 50% compared with MPTCP protocol.
TinyFlow [52]	Xu H et al., 2014	To avoid high latency for short flows and low throughput for long flows	TinyFlow divides the long flow into multiple short flows with 10KB and achieves short flows' rerouting by dynamically changing the output port.	TinyFlow achieves much smaller FCT for all flows in data center networks.
WCMP [53]	Zhou JL et al., 2014	To obtain better load balancing in the asymmetric scenario	WCMP utilizes SDN to deliver fair per-flow bandwidth allocation according to the changing topology.	WCMP avoids bandwidth unfairness when the topology is asymmetric.
RepFlow [54]	Xu H et al., 2014	To reduce the completion time of short flows without any change to switches or host kernels	RepFlow duplicates each short flow and transmits each flow via ECMP	RepFlow provides significant speedup in both mean and 99th percentile FCT for all loads and offers near-optimal FCT when employed with DCTCP.
Expeditus [55]	Wang P et al., 2017	To make load balancing adapt to three-layer Clos topology	The switch monitors the local link load and uses a two-stage path selection mechanism to summarize the cross-layer switch's congestion information, thereby making path selection for each flow.	Expeditus obtains lower average flow completion time and tail latency than ECMP and CONGA under diverse workloads.
RPS [56]	Dixit A et al., 2013	To obtain higher link resources utilization	RPS randomly sprays data packets of a flow to all paths.	RPS achieves much lower flow completion time and better TCP throughput than ECMP.
DRILL [57]	Ghorbani S et al., 2017	To quickly mitigate the congestion induced by burst traffic	DRILL selects the forwarding port for each packet according to the local queue length.	DRILL outperforms state-of-the-art load balancers in Clos networks, especially under high load and incast.
QDAPS [58]	Lv WJ et al., 2018	To avoid packet reordering under network asymmetry	QDAPS selects the appropriate egress port queue for the current packet of the flow according to the remaining queuing time of the previous packet of the same flow	QDAPS greatly mitigates packet reordering problem and reduces flow completion time.
RMC [59]	Zou SJ et al., 2019	To cope with packet reordering and tackle uncertainties in asymmetric networks	RMC utilizes the network coding approach to decrease long-tailed flow completion time	RMC reduces flow completion time by up to 72% compared with existing protocols.
CAPS [16]	Hu JB et al., 2019	To alleviate the negative impact of short and long flows on each other in data center networks	CAPS sprays the short flows' packets to all available paths, while the long ones are transmitted via ECMP to several paths	CAPS reduces flow completion time, but it introduces a lot of redundant packets that exacerbate network congestion.

(3) Recent traffic optimizations, e.g. flow scheduling, load balancing, are usually based on some heuristic method with some presumptions, preventing them from working well under various network conditions. Currently, the learning-based approaches employ deep learning or deep reinforcement learning to schedule

TABLE 8. *(Continued.)* **Summary of schemes in data center networks.**

OPER [60]	Liu S et al., 2019	To avoid extra traffic overhead caused by coding	OPER utilizes systematic coding to achieve self-adaptive load balancing in the data center networks	OPER reduces the flow completion time by up to 71% compared with the state-of-the-art multi-path coding approaches.
APS [15]	Liu JL et al., 2020	To address the problem of resource competition between the long and short flows	APS dynamically separates long flows from short ones on different paths to provide low latency for the short flows.	APS decreases the average completion time for short flows by up to 60% and improves the throughputs for long flows by about 1.68x over the state-of-the-art multipath transmission designs.
CONGA [61]	Alizadeh M et al., 2014	Tackle the problem that the centralized solution is too slow to react to congestion, and the local solution is challenging to collect global congestion information	CONGA leverages the overlay technology to implement a leaf-to-leaf feedback loop to obtain the global status and switches flowlets with the global congestion information.	CONGA can achieve lower flow completion time than ECMP and MPTCP under different workloads.
HULA [62]	Katta N et al., 2016	To break the limitation of limited switch memory and guarantee scalability in real environments	HULA forwards flowlets to the best next hop.	HULA could be run on programmable chipsets, without requiring custom hardware and outperforms CONGA in average flow completion time.
LetFlow [63]	Vanini E et al., 2017	To mitigate the overhead of path detection and improve robustness to asymmetric topologies	Letflow uses a fixed time interval to randomly select the forwarding path for each flowlet.	LetFlow performs much better than other traffic oblivious schemes and achieves lower average flow completions time than CONGA.
AG [64]	Liu JL et al., 2019	To avoid the problems of packet reordering under asymmetric topology and under-utilization of multiple paths	AG adjusts the granularity of path switching according to the degree of topology asymmetry	AG decreases the average and 99th flow completion time by up to 51% and 56% over the state-of-the-art load balancing schemes.
TLB [65]	Hu JB et al., 2019	To avert long-tailed queueing delay and reordering problems of short flows and low throughput of long flows	TLB adjusts the granularity of long flow path switching according to short flow intensity.	TLB significantly diminishes the average flow completion time (AFCT) of short flows by 15%-40% over the state-of-the-art load balancers and delivers high throughput for long flows.
DCTCP [12]	Alizadeh M et al., 2010	To address the inefficiency of tradition TCP in data center networks	DCTCP utilizes Explicit Congestion Notification (ECN) in the network to provide congestion feedback to the end hosts.	DCTCP delivers the same or better throughput than TCP, and provides low latency for short flows.

traffic without any assumptions, thus achieving better performance than heuristic ones. Neural networks in machine learning can be utilized to learn traffic patterns through historic flow traces, leading to future traffic prediction with high accuracy. Through employing these information, traffic scheduling in data center

TABLE 8. *(Continued.)* **Summary of schemes in data center networks.**

ECN* [70]	Wu HT et al., 2012	To reduce the queuing latency of short flows	ECN* employs dequeue marking to speed up the congestion information delivery for instant queue length based on ECN.	dequeue marking is effective for increasing network throughput.
L^2DCT [71]	Munir A et al., 2013	To further improve the transmission performance of short flows and to make the sender actively treat long and short flows differently	L^2DCT schedules flows with the Shortest Remaining Processing Time (SRPT) mechanism and adjusts the congestion window according to the congestion level	L^2DCT reduces the completion for 99th percentile flows by 37% over DCTCP.
DX [72]	Lee C et al., 2015	To design a more fine-grained congestion feedback mechanisms	DX leverages latency as the congestion signal and exploits the queueing delay to decide whether to increase or decrease the window size in the next round in an Additive Increase Multiplicative Decrease (AIMD) fashion.	The latency feedback can be utilized to perform fine-grained congestion control in data center networks and outperform DCTCP.
TCP-TRIM [74]	Zhang T et al., 2017	To avoid severe packet loss at the beginning of ON period under heavy congestion	TCP-TRIM proposes a window inheritance mechanism, in which the congestion window size in the previous ON period is selectively reused, and it employs probe packets to detect the end-to-end delay to regulate the congestion window at the end host side to guarantee high utilization of the bottleneck link.	TCP-TRIM can greatly reduce the HTTP response completion time, while introducing little deployment overhead only at the end hosts.
HPCC [77]	Li YL et al., 2019	To obtain ultra-low latency, high bandwidth and network stability in high-speed networks	HPCC leverages the precise link load information obtained from INT to compute accurate flow rate.	Compared with dcqcn and time, HPCC can shorten the flow completion time, and it will not cause congestion even in the case of large-scale incast.
D^3 [78]	Wilson C et al., 2011	To meet the deadline of short flows	D^3 exploits the application deadline information to let end-hosts calculate the flows' rate before the transmission and deliver the rate to destination	D^3 outperforms TCP in short flow latency and burst tolerance
D^2TCP [79]	Vamanan B et al., 2012	To reduce the execution overhead of D^3 and meet the demand of deadline-aware flows	D^2TCP gives flows with small deadlines higher priority and transmits these flows firstly.	D^2TCP greatly decreases the ratio of missed deadlines compared with DCTCP and D^3.
A^2DTCP [80]	Zhang T et al., 2015	To decrease the deadline-missing ratio due to the conservative additive increase of congestion window	A^2DTCP alters the congestion window increasing speed according to the network congestion extent and the probability of missing deadline.	A^2DTCP significantly reduces the missed deadline ratio and co-exist well with traditional TCP.

networks can be handled gracefully with higher performance. Besides, machine learning mechanisms like deep reinforce learning (DRL) can automatically learn a model by interacting with the environment. For instance, CODA [99] recognizes different coflows among individual flows with an error-tolerant scheduler via machine

TABLE 8. *(Continued.)* **Summary of schemes in data center networks.**

FGRTO [81]	Vasudevan V et al., 2009	To avoid TCP incast collapse in data center applications	FGRTO eliminates the minimum RTO and enables finer-grained RTO.	FGRTO mitigates TCP incast collapse to some extents, but it may trigger a large number of spurious timeouts and unnecessary retransmissions.
ARS [83]	Huang JW et al., 2016	To cope with TCP incast	ARS alters the number of concurrent TCP flows via batching application requests based on the congestion status acquired from transport layer.	ARS greatly reduces the incast probability across different TCP protocols.
PS [84]	Huang JW et al., 2020	To tackle TCP incast throughput collapse.	PS adjusts the IP packet size to decrease the incast probability.	PS greatly improves the goodput of different data center TCP protocols.
AP [85]	Zou SJ et al., 2017	To better handle micro-burst in data center networks	AP adjusts the burstiness and the time interval between consequent packets according to the flow concurrency dynamically.	AP keeps compatibility on existing transport layer protocols without any changes on hardware switch.
OFTCP [19]	Huang JW et al., 2019	To deal with TCP outcast	OFTCP alters ACK packets' advertisement window to guarantee even bandwidth sharing of concurrent flows at the receiver side.	OFTCP gains high throughput fairness and significantly decreases the tail flow completion time.
CODA [99]	Zhang H et al., 2016	To recognize different coflows without any applications modifications	CODA utilizes an incremental clustering algorithm to perform application-transparent coflow identification.	CODA achieves over 90% identification accuracy.
TaTCP [101]	Liu S et al., 2019	To decrease the task completion time	TaTCP efficiently decreases the task completion time by sharing the flow-tardiness information through receiver-driven coordination.	TaTCP obtains 50-70% improvement in decreasing task completion time than other protocols in typical data center applications.
C3 [14]	Suresh J et al., 2015	To tackle servers' variation and herd behaviour problems to choose the fastest replica server	C3 monitors the waiting keys' queue length and its service time at the server-side and piggyback this information to the client to select the fastest server via ranking the available replica servers.	C3 significantly reduces the latencies and the mean, median, and tail and provides higher system throughput.

learning. AuTo [110] develops a two-level DRL system to automatically determine the priority of long and short flows and the sending rate and paths of long flows. However, the training model's accuracy depends on the quantity and quality of collected experiences to a large degree. Unfortunately, the collected data in current DRL algorithms do not invariably reflect the best action of the current state. Therefore, it takes a long time for training and easily converges to a suboptimal policy. Besides, the high dynamic and strong burst of traffic in data center networks make the model obtained by machine learning challenging to adapt to various network scenarios. Therefore, a better mix of machine learning and traffic optimizations in the data center is a promising research trend.

IX. CONCLUSION

In recent years, with the boom of advanced technologies like cloud computing, more and more companies have built large data centers to provide various online services. To support these services, data centers employ multi-rooted tree topologies to utilize the resource of multiple paths to provide high bandwidth. This paper presents a summary and provides a classification of existing schemes from different network

layers in DCNs. To inspire future research, this paper also discusses some potential research trends in the future to help readers make new contributions in this area.

APPENDIX

See Table 8.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, 2008, pp. 63–74.
[2] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and Operations," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 640–656, 1st Quart., 2017.
[3] J. Zhang, F. Ren, and C. Lin, "Survey on transport control in data center networks," *IEEE Netw.*, vol. 27, no. 4, pp. 22–26, Jul./Aug. 2013.
[4] *IEEE 802.1: Data Center Bridging Task Group*. Accessed: Apr. 23, 2010. [Online]. Available: http://www.ieeeS02.orgil/pages/dcbridges.html
[5] Y. Zhang and N. Ansari, "Fair quantized congestion notification in data center networks," *IEEE Trans. Commun.*, vol. 61, no. 11, pp. 4690–4699, Nov. 2013.
[6] W. Jiang, F. Ren, R. Shu, and C. Lin, "Sliding mode congestion control for data center Ethernet networks," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1404–1412.
[7] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 523–536, Sep. 2015.
[8] Z. Guo, S. Liu, and Z.-L. Zhang, "Traffic control for RDMA-enabled data center networks: A survey," *IEEE Syst. J.*, vol. 14, no. 1, pp. 677–688, Mar. 2020.
[9] Infiniband Trade Association. (2015). *InfiniBandTM Architecture Specification Volume 1 Release 1.3*. [Online]. Available: https://cw.infiniband.org/document/dl/7781
[10] J. Zhang, F. R. Yu, S. Wang, T. Huang, Z. Liu, and Y. Liu, "Load balancing in data center networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2324–2352, 3rd Quart., 2018.
[11] C. Hopps, *Analysis of an Equal-Cost Multi-Path Algorithm*, document RFC 2992, 2000.
[12] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.
[13] W. Jiang, H. Xie, X. Zhou, L. Fang, and J. Wang, "Understanding and improvement of the selection of replica servers in key–value stores," *Inf. Syst.*, vol. 83, pp. 218–228, Jul. 2019.
[14] J. Suresh, M. Canini, S. Schmid, and A. Feldmann, "C3: Cutting tail latency in cloud data stores via adaptive replica selection," in *Proc. USENIX NSDI*, 2015, pp. 513–527.
[15] J. Liu, J. Huang, W. Lv, and J. Wang, "APS: Adaptive packet spraying to isolate mix-flows in data center network," *IEEE Trans. Cloud Comput.*, early access, Apr. 3, 2020, doi: 10.1109/TCC.2020.2985037.
[16] J. Hu, J. Huang, W. Lv, Y. Zhou, J. Wang, and T. He, "CAPS: Coding-based adaptive packet spraying to reduce flow completion time in data center," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2338–2353, Dec. 2019.
[17] J. A. Aroca and A. F. Anta, "Bisection (band) width of product networks with application to data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 570–580, Mar. 2013.
[18] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast congestion control for TCP in data-center networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 345–358, Apr. 2013.
[19] J. Huang, S. Li, R. Han, and J. Wang, "Receiver-driven fair congestion control for TCP outcast in data center networks," *J. Netw. Comput. Appl.*, vol. 131, pp. 75–88, Apr. 2019.
[20] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. ACM HotNets*, 2012, pp. 31–36.
[21] C. Ruan, J. Wang, W. Jiang, J. Huang, G. Min, and Y. Pan, "FSQCN: Fast and simple quantized congestion notification in data center Ethernet," *J. Netw. Comput. Appl.*, vol. 83, pp. 53–62, Apr. 2017.
[22] W. Jiang, F. Ren, and C. Lin, "Phase plane analysis of quantized congestion notification for data center Ethernet," *IEEE/ACM Trans. Netw.*, vol. 23, no. 1, pp. 1–14, Feb. 2015.
[23] C. Ruan, J. Wang, J. Huang, and W. Jiang, "Analysis on buffer occupancy of quantized congestion notification in data center networks," *IEICE Trans. Commun.*, vol. E99.B, no. 11, pp. 2361–2372, 2016.
[24] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wojcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proc. ACM SIGCOMM*, 2017, pp. 29–42.
[25] Y. Gao, Y. Yang, T. Chen, J. Zheng, B. Mao, and G. Chen, "DCQCN+: Taming large-scale incast congestion in RDMA over Ethernet networks," in *Proc. IEEE INFOCOM*, Sep. 2018, pp. 110–120.
[26] C. Tian, B. Li, L. Qin, J. Zheng, J. Yang, W. Wang, G. Chen, and W. Dou, "P-PFC: Reducing tail latency with predictive PFC in lossless data center networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1447–1459, Jun. 2020.
[27] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and T. Moscibroda, "Multi-path transport for RDMA in datacenters," in *Proc. USENIX NSDI*, 2018, pp. 357–371.
[28] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen, "Tagger: Practical PFC deadlock prevention in data center networks," in *Proc. ACM CoNEXT*, 2017, pp. 889–902.
[29] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for RDMA," in *Proc. ACM SIGCOMM*, 2018, pp. 313–326.
[30] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM*, 2011, pp. 51–62.
[31] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM*, 2008, pp. 75–86.
[32] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, P. Patel, and S. Sengupta, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2009, pp. 51–62.
[33] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proc. ACM SIGCOMM*, 2017, pp. 253–266.
[34] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM*, 2013, pp. 3–14.
[35] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, N. B. Kondapa, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev, S. Padgett, F. Rabe, S. Ray, M. Tewari, M. Tierney, M. Zahn, J. Zolla, J. Ong, and A. Vahdat, "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proc. ACM SIGCOMM*, 2018, pp. 74–87.
[36] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM SIGCOMM*, 2015, pp. 123–137.
[37] S. Choi, B. Burkov, A. Eckert, T. Fang, S. Kazemkhani, R. Sherwood, Y. Zhang, and H. Zeng, "FBOSS: Building switch software at scale," in *Proc. ACM SIGCOMM*, 2018, pp. 342–356.
[38] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[39] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. USENIX NSDI*, 2010, pp. 1–15.
[40] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. ACM CoNEXT*, 2011, pp. 1–12.
[41] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: low-overhead datacenter traffic management using end-host-based elephant detection," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1629–1637.
[42] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized 'zero-queue' datacenter network," in *Proc. ACM SIGCOMM*, 2014, pp. 307–318.
[43] S. Wang, J. Zhang, T. Huang, T. Pan, J. Liu, and Y. Liu, "FDALB: Flow distribution aware load balancing for datacenter networks," in *Proc. IEEE/ACM IWQoS*, Jun. 2016, pp. 1–2.
[44] W. Wang, Y. Sun, K. Salamatian, and Z. Li, "Adaptive path isolation for elephant and mice flows by exploiting path diversity in datacenters," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 1, pp. 5–18, Mar. 2016.

[45] S. M. Irteza, H. M. Bashir, T. Anwar, I. A. Qazi, and F. R. Dogar, "Efficient load balancing over asymmetric datacenter topologies," *Comput. Commun.*, vol. 127, no. 9, pp. 1–12, Sep. 2018.

[46] P. F. De, L. Maggi, A. Massaro, D. Saucez, J. Leguay, and E. Altman Blind, "Adaptive and robust flow segmentation in datacenters," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 10–18.

[47] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "FlowBender: Flow-level adaptive routing for improved latency and throughput in datacenter networks," in *Proc. ACM CoNEXT*, 2014, pp. 149–160.

[48] N. Katta, M. Hira, A. Ghag, C. Kim, I. Keslassy, and J. Rexford, "CLOVE: How i learned to stop worrying about the core and love the edge," in *Proc. ACM HotNets*, 2016, pp. 155–161.

[49] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM*, 2011, pp. 266–277.

[50] M. Kheirkhah, I. Wakeman, and G. Parisis, "Multipath transport and packet spraying for efficient data delivery in data centres," *Comput. Netw.*, vol. 162, no. 1, pp. 1–15, 2019.

[51] J. Huang, W. Li, Q. Li, T. Zhang, P. Dong, and J. Wang, "Tuning high flow concurrency for MPTCP in data center networks," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–15, Dec. 2020.

[52] H. Xu and B. Li, "TinyFlow: Breaking elephants down into mice in data center networks," in *Proc. IEEE 20th Int. Workshop Local Metrop. Area Netw. (LANMAN)*, May 2014, pp. 1–6.

[53] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "WCMP: Weighted cost multipathing for improved fairness in data centers," in *Proc. ACM EuroSys*, 2014, pp. 1–14.

[54] H. Xu and B. Li, "RepFlow: Minimizing flow completion times with replicated flows in data centers," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 1581–1589.

[55] P. Wang, H. Xu, Z. Niu, D. Han, and Y. Xiong, "Expeditus: Congestion-aware load balancing in Clos data center networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3175–3188, Oct. 2017.

[56] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2130–2138.

[57] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro load balancing for low-latency data center networks," in *Proc. ACM SIGCOMM*, 2017, pp. 225–238.

[58] J. Huang, W. Lyu, W. Li, J. Wang, and T. He, "Mitigating packet reordering for random packet spraying in data center networks," *IEEE/ACM Trans. Netw.*, to be published, doi: 10.1109/TNET.2021.3056601.

[59] S. Zou, J. Huang, J. Wang, and T. He, "Improving TCP robustness over asymmetry with reordering marking and coding in data centers," in *Proc. IEEE ICDCS*, Jul. 2019, pp. 57–67.

[60] S. Liu, J. Huang, W. Jiang, J. Wang, and T. He, "Reducing flow completion time with replaceable redundant packets in data center networks," in *Proc. IEEE ICDCS*, Jul. 2019, pp. 46–56.

[61] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, and G. Varghese, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM SIGCOMM*, 2014, pp. 503–514.

[62] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proc. ACM SOSR*, 2016, pp. 1–12.

[63] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, "Let it flow: Resilient asymmetric load balancing with flowlet switching," in *Proc. USENIX NSDI*, 2017, pp. 407–420.

[64] J. Liu, J. Huang, W. Li, and J. Wang, "AG: Adaptive switching granularity for load balancing with asymmetric topology in data center network," in *Proc. IEEE ICNP*, Oct. 2019, pp. 1–11.

[65] J. Hu, J. Huang, W. Lv, W. Li, J. Wang, and T. He, "TLB: Traffic-aware load balancing with adaptive granularity in data center networks," in *Proc. ACM ICPP*, 2019, pp. 1–10.

[66] W. Bai, S. Hu, K. Chen, K. Tan, and Y. Xiong, "One more config is enough: Saving (DC)TCP for high-speed extremely shallow-buffered datacenters," in *Proc. IEEE INFOCOM*, Jul. 2020, pp. 2007–2016.

[67] W. Bai, K. Chen, L. Chen, C. Kim, and H. Wu, "Enabling ECN over generic packet scheduling," in *Proc. ACM CoNEXT*, 2016, pp. 191–204.

[68] J. Zhang, W. Bai, and K. Chen, "Enabling ECN for datacenter networks with RTT variations," in *Proc. ACM CoNEXT*, 2019, pp. 233–245.

[69] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan, and Y. Wang, "Aeolus: A building block for proactive transport in datacenters," in *Proc. ACM SIGCOMM*, 2020, pp. 422–434.

[70] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, and Y. Zhang, "Tuning ECN for data center networks," in *Proc. ACM CoNEXT*, 2012, pp. 25–36.

[71] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2157–2165.

[72] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in *Proc. USENIX ATC*, 2015, pp. 403–415.

[73] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: RTT-based congestion control for the datacenter," in *Proc. ACM SIGCOMM*, 2015, pp. 537–550.

[74] T. Zhang, J. Wang, J. Huang, J. Chen, Y. Pan, and G. Min, "Tuning the aggressive TCP behavior for highly concurrent HTTP connections in intra-datacenter," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3808–3822, Dec. 2017.

[75] T. Zhang, J. Huang, K. Chen, J. Wang, J. Chen, Y. Pan, and G. Min, "Rethinking fast and friendly transport in data center networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2364–2377, Oct. 2020, doi: 10.1109/TNET.2020.3012556.

[76] T. Zhang, J. Huang, S. Zou, S. Liu, J. Hu, J. Liu, C. Ruan, J. Wang, and G. Min, "DDT: Mitigating the competitiveness difference of data center TCPs," in *Proc. ACM APNet*, 2019, pp. 8–14.

[77] Y. Li, R. Miao, H. H. Liu, Y. Zhang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "HPCC: High precision congestion control," in *Proc. ACM SIGCOMM*, 2019, pp. 44–58.

[78] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proc. ACM SIGCOMM*, 2011, pp. 50–61.

[79] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in *Proc. ACM SIGCOMM*, 2012, pp. 115–126.

[80] T. Zhang, J. Wang, J. Huang, Y. Huang, J. Chen, and Y. Pan, "Adaptive-acceleration data center TCP," *IEEE Trans. Comput.*, vol. 64, no. 6, pp. 1522–1533, Jun. 2015.

[81] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained TCP retransmissions for datacenter communication," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 303–314, Aug. 2009.

[82] W. Bai, K. Chen, H. Wu, W. Lan, and Y. Zhao, "PAC: Taming TCP incast congestion using proactive ACK control," in *Proc. IEEE ICNP*, Oct. 2014, pp. 385–396.

[83] J. Huang, T. He, Y. Huang, and J. Wang, "ARS: Cross-layer adaptive request scheduling to mitigate TCP incast in data center networks," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[84] J. Huang, Y. Huang, J. Wang, and T. He, "Adjusting packet size to mitigate TCP incast in data center networks with COTS switches," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 749–763, Jul./Sep. 2020.

[85] S. Zou, J. Huang, J. Wang, and T. He, "Flow-aware adaptive pacing to mitigate TCP incast in data center networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 134–147, Feb. 2021, doi: 10.1109/TNET.2020.3027749.

[86] S. Shukla, S. Chan, A. S.-W. Tam, A. Gupta, Y. Xu, and H. J. Chao, "TCP PLATO: Packet labelling to alleviate time-out," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 65–76, Jan. 2014.

[87] C. Jiang, D. Li, and M. Xu, "LTTP: An LT-code based transport protocol for many-to-one communication in data centers," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 52–64, Jan. 2014.

[88] P. Prakash, A. Dixit, Y. C. Hu, and R. Kompella, "The TCP outcast problem: Exposing unfairness in data center networks," in *Proc. USENIX NSDI*, 2012, pp. 1–14.

[89] J. Zhang, F. Ren, X. Yue, R. Shu, and C. Lin, "Sharing bandwidth by allocating switch buffer in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 39–51, Jan. 2014.

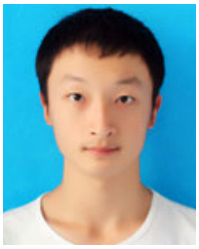
[90] P. Cheng, F. Ren, R. Shu, and C. Lin, "Catch the whole lot in an action: Rapid precise packet loss notification in data center," in *Proc. USENIX NSDI*, 2014, pp. 17–28.

[91] J. Zhang, F. Ren, R. Shu, and P. Cheng, "TFC: Token flow control in data center networks," in *Proc. ACM EuroSys*, 2016, pp. 1–14.

[92] H. Susanto, H. Jin, and K. Chen, "Stream: Decentralized opportunistic inter-coflow scheduling for datacenter networks," in *Proc. IEEE ICNP*, Nov. 2016, pp. 1–10.

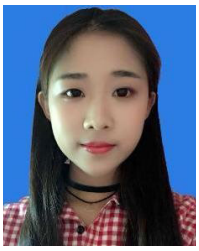
[93] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 98–109.

[94] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 431–442.
[95] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with Varys," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 443–454.
[96] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," in *Proc. ACM SIGCOMM*, Aug. 2015, pp. 393–406.
[97] Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang, "Rapier: Integrating routing and scheduling for coflow-aware data center networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 424–432.
[98] Z. Li, Y. Zhang, D. Li, K. Chen, and Y. Peng, "OPTAS: Decentralized flow monitoring and scheduling for tiny tasks," in *Proc. INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 424–432.
[99] H. Zhang, L. Chen, B. Yi, K. Chen, M. Chowdhury, and Y. Geng, "CODA: Toward automatically identifying and scheduling coflows in the dark," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 160–173.
[100] W. Wang, S. Ma, B. Li, and B. Li, "Coflex: Navigating the fairness-efficiency tradeoff for coflow scheduling," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.
[101] S. Liu, J. Huang, Y. Zhou, J. Wang, and T. He, "Task-aware TCP in data center networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 389–404, Feb. 2019.
[102] V. Jalaparti, P. Bodik, S. Kandula, I. Menache, M. Rybalkin, and C. Yan, "Speeding up distributed request-response workflows," in *Proc. ACM SIGCOMM Conf.*, Aug. 2013, pp. 219–230.
[103] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, "Scaling memcache at facebook," in *Proc. USENIX NSDI*, 2013, pp. 385–398.
[104] X. Zhou, L. Fang, H. Xie, and W. Jiang, "TAP: Timeliness-aware predication-based replica selection algorithm for key-value stores," *Concurrency Comput. Pract. Exper.*, vol. 31, no. 17, pp. 1–13, 2019.
[105] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," in *Proc. ACM SIGCOMM*, 2012, pp. 139–150.
[106] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "PFabric: Minimal near-optimal datacenter transport," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Aug. 2013, pp. 435–446.
[107] R. Rojas-Cessa, Y. Kaymak, and Z. Dong, "Schemes for fast transmission of flows in data center networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1391–1422, 3rd Quart., 2015.
[108] A. M. Abdelmoniem and B. Bensaou, "Curbing timeouts for TCP-incast in data centers via a cross-layer faster recovery mechanism," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 675–683.
[109] A. Langley *et al.*, "The QUIC transport protocol: Design and Internet-scale deployment," in *Proc. ACM SIGCOMM*, 2017, pp. 183–196.
[110] L. Chen, J. Lingys, K. Chen, and F. Liu, "AuTO: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. ACM SIGCOMM*, 2018, pp. 191–205.



WEIHE LI is currently pursuing the master's degree with the School of Computer Science and Engineering, Central South University, China. His research interests include video streaming and data center networks.

JINGLING LIU received the B.E. degree from Central South University, China, in 2016, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. Her current research interest includes the area of data center networks.



SHIQI WANG is currently pursuing the master's degree with the Department of Computer Science and Engineering, Central South University, China. Her research interest includes data center networks.

TAO ZHANG received the Ph.D. degree from the School of Computer Science and Engineering, Central South University, China. He is currently an Associate Professor with the Hunan Province Key Laboratory of Industrial Internet Technology and Security, Changsha University, China. His research interests include congestion control, performance modeling, analysis, and data center networks.

SHAOJUN ZOU is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Central South University, Changsha, China. His current research interests include congestion control and data center networks.

JINBIN HU received the B.S. and M.S. degrees from Beijing Jiaotong University, China, in 2008 and 2011, respectively. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Central South University, China. She also works as a Teacher with the School of Computer and Communication Engineering, Changsha University of Science and Technology. Her current research interest includes the area of data center networks.

WANCHUN JIANG (Member, IEEE) received the bachelor's and Ph.D. degrees in computer science and technology from Tsinghua University, China, in 2009 and 2014, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, Central South University, China. His research interests include congestion control, data center networks, and the application of control theory in computer networks.

JIAWEI HUANG received the bachelor's degree from the School of Computer Science, Hunan University, in 1999, and the master's and Ph.D. degrees from the School of Information Science and Engineering, Central South University, China, in 2004 and 2008, respectively. He is currently a Professor with the School of Computer Science and Engineering, Central South University. His research interests include performance modeling, analysis, and optimization for data center networks.