QoE-Optimized MultiPath Scheduling for Video Services in Large-Scale Peer-to-Peer CDNs

Dehui Wei, *Member, IEEE*, Jiao Zhang, *Senior Member, IEEE*, Xiang Liu, *Graduate Student Member, IEEE*, Haozhe Li, Zhichen Xue, Tao Huang, *Senior Member, IEEE*, Linshan Jiang, and Jialin Li

***Abstract*— Video content providers such as Douyin implement Peer-to-Peer Content Delivery Networks (PCDNs) to reduce the costs associated with Content Delivery Networks (CDNs) while still maintaining optimal user-perceived quality of experience (QoE). PCDNs rely on the remaining resources of edge devices, such as edge access devices and hosts, to store and distribute data with a Multiple-Server-to-One-Client (MS2OC) communication pattern. MS2OC parallel transmission pattern suffers from severe data out-of-order issues. PCDNs offer significant cost savings by using multiple low-cost edge devices. However, due to its unique characteristics, including pull-based streaming transmission, many heterogeneous paths, and large receiving buffers, directly applying existing schedulers designed for Multipath TCP (MPTCP) to PCDN fails to meet the two goals of high aggregate bandwidth and low end-to-end delivery latency. To tackle this issue, we provide a detailed overview of Douyin's self-developed PCDN video transmission system and introduce the first QoE-enhanced packet-level scheduler for PCDN systems, named Pscheduler. Pscheduler evaluates path quality with a congestion-control-decoupled algorithm and employs our proposed *path-pick-packet* method for data distribution, ensuring a smooth video playback experience. Additionally, we propose a redundant transmission algorithm to enhance task download speeds for segmented video transmission. Our extensive online A/B tests, involving 100,000 Douyin users generating tens of millions of video data points, demonstrate that Pscheduler achieves an average improvement of 60% in goodput, a 20% reduction in data delivery waiting time, and a 30% reduction in rebuffering rates. Furthermore, we conducted simulation experiments that further validate the effectiveness of Pscheduler, confirming its improvements in performance metrics under various network conditions.**

***Index Terms*— Large-scale A/B test, peer-to-peer CDNs, QoE-enhanced, scheduling and forwarding, video transmission.**

Received 31 May 2024; revised 24 November 2024; accepted 9 December 2024. Date of publication 13 January 2025; date of current version 20 February 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB2901700, in part by the National Natural Science Foundation of China (NSFC) under Grant 62132022 and Grant 62372053, and in part by Singapore Ministry of Education (Academic Research Fund Tier 1 and 2) under Grant MOE-T2EP20222-0016 and Grant T1 251RES2409. An earlier version of this paper was presented at the 43th IEEE International Conference on Computer Communications (InfoComm 2024) [DOI: 10.1109/INFOCOM52122.2024.10621211]. *(Corresponding author: Jiao Zhang.)*

Dehui Wei is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the School of Computing, National University of Singapore, Singapore 119077 (e-mail: dehuiwei@bupt.edu.cn).

Jiao Zhang and Tao Huang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the Purple Mountain Laboratories, Nanjing 210000, China (e-mail: jiaozhang@bupt.edu.cn; htao@bupt.edu.cn).

Xiang Liu and Jialin Li are with the School of Computing, National University of Singapore, Singapore 119077 (e-mail: liuxiang@comp.nus.edu.sg; lijl@comp.nus.edu.sg).

Haozhe Li and Zhichen Xue are with ByteDance Ltd., Beijing 100098, China (e-mail: lihaozhe.2021@bytedance.com; xuezhichen@bytedance.com).

Linshan Jiang is with the Institute of Data Science, National University of Singapore, Singapore 119077 (e-mail: linshan@nus.edu.sg).

Digital Object Identifier 10.1109/JSAC.2025.3528809

I. Introduction

VIDEO applications have emerged as an essential medium for global information exchange due to their numerous advantages, such as diverse content and effective communication capabilities [1]. As one of the world's leading video providers, Douyin serves hundreds of millions of daily users, and the annual traffic cost exceeds 600 million yuan. Therefore, saving CDN traffic costs while maintaining user QoE has become a vital issue for short video platforms [2].

Video content providers such as Douyin [3], [4] implement PCDNs [5], [6], [7], [8], [9] to reduce costs associated with CDNs [10], [11] without compromising the high-quality data transmission of massive video services. These applications are placing increasingly sophisticated demands on the network to transfer data for greater reliability on PCDN. Unlike traditional CDNs that rely on large-scale deployment of dedicated servers, PCDNs leverage under-utilized resources on edge devices, such as edge access devices and hosts, to store and distribute data, resulting in a lower unit traffic price. However, these devices are more resource constrained and offer less stable performance than their CDN counterparts. Therefore, PCDNs have a low unit price of traffic but have the disadvantage of insufficient remaining bandwidth of a single device. Hence, PCDNs use multiple devices for parallel transmission with a pull-based streaming multipath transmission protocol to support the viewing needs of a single user. Therefore, it is critical to efficiently distribute video data to the paths between different low-cost devices and the client to satisfy the quality requirements of video data transmission.

To provide similar user experiences, PCDN uses multiple edge devices to serve a single user through parallel transmission [12], [13]. Such an architecture leads to a *Multiple-Server-to-One-Client (MS2OC) pattern*, a significant deviation from traditional multipath transmission schemes, such as CMT-SCTP [14], MPTCP [15] and multipath-enabled QUIC (MPQUIC) [16], which follow a One-Server-to-One-Client (OS2OC) pattern, targeting the scenario of using multipath (usually two paths) in One-Server-to-One-Client.

The PCDN architecture brings new requirements in video content serving. Firstly, a majority (80%) of paths in PCDNs

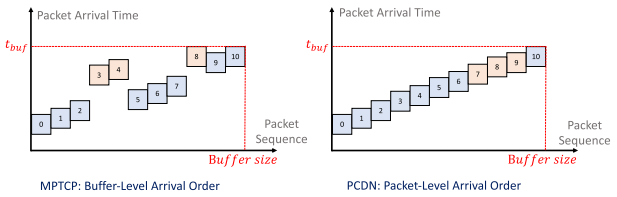


Fig. 1. Difference in order goals between MPTCP and PCDN.

do not individually meet users' video streaming demands with satisfactory quality. PCDNs therefore require *aggregating* the bandwidth of multiple paths during a single transmission for a fast video download rate. Secondly, video data received out-of-order cannot be delivered for playback, leading to rebufferring and poor video quality. PCDN needs to ensure ordered data arrival [17] for time-variance of availability even in the presence of path diversity and a multi-source transmission scheme (MS2OC).

A naive solution is to apply existing schedulers designed for MPTCP, such as BLEST [18] and ECF [19], to PCDNs. We, however, observe that they fail to meet the above PCDNs objectives due to the fundamental differences in design goals:

- **Low aggregate data download rate:** MPTCP is designed to achieve an aggregate transmission bandwidth no less than that of the fastest path (in terms of bandwidth). In the presence of path speed diversity, MPTCP suffers from Head-of-Line (HOL) blocking where a slow path prevents packet acknowledgment on the fast path.HOL can lead to buffer filling and stalling of the fast path [20], [21]. The aggregate bandwidth is therefore limited by the receive rate of the slowest path R_{min}. MPTCP schedulers address this issue by applying an aggressive penalty to the slower paths when HOL occurs, suspending it from sending data and improving the rate of the faster paths. As a result, the aggregate receive rate in MPTCP typically approaches the bandwidth of the fastest path, i.e., $R_{agg} \approx R_{max}$. This, however, does not meet the goal of a PCDN, where $R_{agg} \approx \sum R$.
- **High end-to-end data delivery latency:** MPTCP schedulers optimize for buffer-level ordering, i.e., the latency of receiving all packets in the buffer from both slow and fast paths. As shown in Fig.1, such designs tolerate the unordered arrival of individual packets within the buffer. However, out-of-order packet arrival delays the average data delivery latency, which in the worst case approaches the longest Round-Trip Time (RTT) among all the paths. This is in direct conflict with PCDN's goal of low end-to-end data delivery latency.

We believe that PCDN's new deployment model and performance goals fundamentally necessitate a new transmission scheduling mechanism for the data routing on multiple optimality criteria. However, proposing an efficient scheduler for PCDN faces many challenges. 1) MS2OC results in a *pull-based transmission pattern* where the client schedules individual video data `requests` to the set of source servers. Each data response therefore only opens a *single* flow control window in *one* source server connection. Such a flow control scheme differs significantly from MPTCP (or OS2OC in general), where one ACK can release multiple windows due to accumulated acknowledgments. It severely restricts the possible flow selection decisions the client can make. Without pausing the transmission after a data response, the scheduler will converge to a round-robin policy and also thereby eliminate the clock asynchronous problem between the sender and receiver [22]; 2) Existing multi-path schedulers heavily rely on path information provided by the Congestion Control (CC) algorithms. However, CCs can lead to *large cumulative errors in path condition prediction* due to phase fluctuations, particularly in the PCDN targeted unstable network environment. Real-world deployment experiences reveal that a single user requires up to 10 upload servers to fulfill its viewing demand, which further amplifies error accumulation and increases the latency for the forwarding; 3) The need for a larger number of paths per client increases the impact of *network heterogeneity*.

To address the above challenges, **we propose Pscheduler, a packet-level scheduler for PCDN systems**. Pscheduler consists of three components: a CC-decoupled path quality estimator, a Queue-Direct-Access data distribution algorithm, and a redundancy algorithm. Concretely, 1) Pscheduler takes a *"zero-dependency"* path status estimation approach, decoupling the scheduler from the congestion control algorithms. Consequently, path information can be predicted more accurately without the impact of CC iterations. 2) Pscheduler leverages the estimated path information to schedule the packet sequence numbers for each path. This *path-aware* packet scheduling maximizes the chance of in-order reception at the receiver. In particular, Pscheduler adopts a *Queue-Direct-Access* approach to forward and retrieve packets from the global request queue, effectively addressing the dynamic network issue in PCDNs. 3) Pscheduler features a redundancy algorithm to enhance transmission reliability without wasting bandwidth. The algorithm determines if there is remaining space at the end of each short flow to accelerate the download process.

We implement Pscheduler in Douyin's PCDN system and conduct large-scale online A/B testing with approximately 100,000 Douyin users whose APP updated to Pscheduler to participate in the test, generating tens of millions of video data. We observed that Pscheduler achieved an average improvement of 60% in goodput, a 20% reduction in data delivery waiting time, and a 30% reduction in rebuffering rate. Furthermore, controlled experiments show that Pscheduler outperforms other state-of-the-art schedulers, making it a better scheduling candidate for PCDN systems.

Contributions: In summary, the key contributions of this work are:

- We provide the comprehensive detail of **a new video delivery network – PCDN** and analyze the challenges it poses in multipath data packet scheduling.
- We propose the first **compute-aware MS2OC scheduling mechanism for PCDN, Pscheduler**, to route the data and ensure smooth video playback.
- We **implement Pscheduler in the PCDN video system of Douyin**.

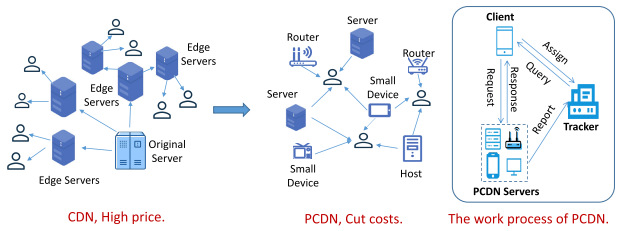


Fig. 2. The high traffic cost is driving the transition from CDN to PCDN.

- We **conduct large-scale online tests of Pscheduler through the Douyin App**. The evaluation results with tens of millions of video data samples provide strong evidence that Pscheduler is effective in improving video QoE metrics.

Our prior work [23] presented a practical large-scale online A/B test. Based on the work [23], we make the following new contributions: 1) we re-organized §I, §II and §VI to highlight the motivation, challenges and insights to cope with multi-source transmission scheme scenarios. We reviewed more related work to present a detailed background; 2) we add new diagrams, redraw some of the diagrams and improve the presentation of several sections to make the paper clearer, especially for the design of our method in §IV; 3) we add more detailed simulations in §V-D to study how Pscheduler may affect the system's performance considering the metrics.

In what follows, we first detail Douyin's PCDN including the architecture, the protocol, and advantages in §II. We present the key aspects of motivation in §III. We propose Pscheduler – the first MS2OC scheduling mechanism in PCDN in §IV. In §V and §V-D, we implement PScheduler in Douyin's PCDN system and present the evaluation results through both practical large-scale online and simulation tests. §VI describes the related work. Finally, we draw the main conclusions in §VII.

II. Background

A. PCDN-Multipath Video Distribution Network

This section focuses on introducing the network architecture and transmission protocol of the Douyin self-developed PCDN, which has been operating on Douyin for four years.

1) PCDN Network Architecture: To combat the high costs associated with traditional CDNs, video content providers like Douyin have turned to PCDNs in their underlying networks, as illustrated in Fig.2. CDNs rely on a tree-like interconnected server network to provide high-speed and stable bandwidth services [24]. The core idea of CDN is to load balance the original content to the user's nearest edge server to reduce data transmission delay. The external CDN stores *all* videos. As this paper focuses on the design of our PCDN system, we treat the external CDN as a black-box; readers interested in a centralized CDN design may refer to prior literature [25].

However, deploying CDN requires significant investment in infrastructure such as Internet data centers (IDCs) and dedicated edge servers, leading to high traffic rent fees [26]. In contrast, PCDNs utilize massive amounts of fragmented under-utilized computing storage and network resources available on the edge devices to distribute data. This approach enables the building of a low-cost, high-quality content distribution network that helps Douyin effectively manage its content delivery and maintain a competitive edge in the market.

The Douyin self-developed PCDN consists of three key components: 1) A distributed network of edge devices with under-utilized resources. 2) A client-side library integrated into users' mobile devices. 3) A centralized tracker cluster operating on public network to help the computation and time-variable routing. Each device within PCDN serves as an uploader, caching a range of videos, typically storing entire video segments for efficiency. The system of devices is designed for simplicity: the tracker makes decisions regarding file distribution, while the client library manages data transmission. Trackers serve as a control plane in PCDN, collecting data on video popularity from client requests and operational states from devices to execute fine-grained video pre-allocation across devices. Consequently, the tracker maintains a detailed mapping table, linking each video ID to a list of devices that store the corresponding video. When the client requests a video, it first consults the tracker to locate the devices containing the video file. We partition each video into equal-length (in duration) *segments*. The default segment length is 10 seconds. Following this, the client establishes connections with these devices to commence data transmission. Due to the limited and fluctuating available bandwidth of individual devices in PCDN, it adopts a multipath parallel transmission approach from multiple dispersed devices to ensure video service quality. The protocol's specifics are detailed below.

2) Pull-based Multipath Transmission Protocol: In PCDN, devices are invisible from each other, necessitating the client to manage the control across flows from these devices by a pull-based transmission protocol to help do the routing. This process involves the client establishing individual connections with a set of devices and sending `request` packets to each. Each request packet asks for a corresponding data packet and transfers one segment, as shown on the right of Fig.3. In the context of video transmission, each device opens a flow control window. The client library strategically decides which request to send through which connection, based on the availability of free windows for effective flow control and scheduling. Upon receiving a request, a device responds with the requested data packet. If these packets arrive in the correct sequence, they are promptly forwarded to the player for decoding and playback. As a result, the client side needs to schedule data packets across different connections to ensure that the data arrives in the correct order, preserving the integrity and continuity of the video stream to meet the demands on the network for better quality.

B. Scheduling in PCDN

The heterogeneity of paths in multipath transmission (such as different RTTs and different packet loss rates), brings significant challenges to the orderly transmission of forwarded data [27]. Most existing scheduling algorithms focus on the HoL problem in MPTCP. This refers to the situation where

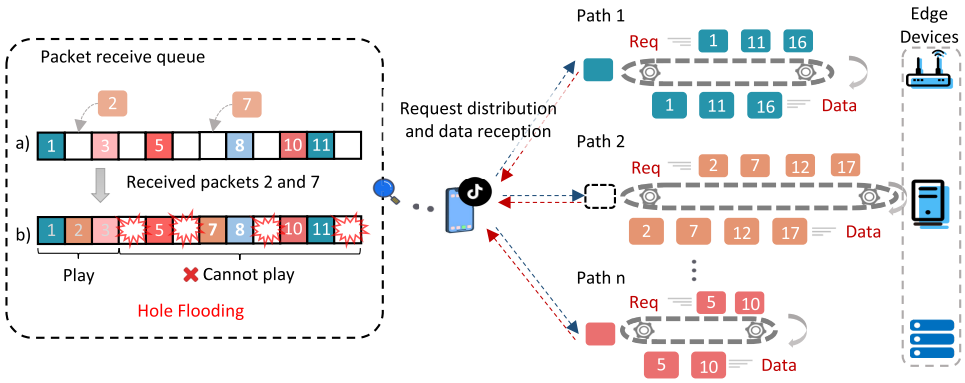


Fig. 3. Data likes rotating on conveyor belts of different lengths. Out-of-order data can cause hole flooding.

packets with lower sequence numbers on the slower path arrive later at the receiving end than packets with higher sequence numbers on the faster path. Although the fast path data packets are already in the receive buffer, the receiving end cannot confirm them because, in TCP/MPTCP, the ACK sequence number is usually determined by the highest data packet sequence number that arrives at the receiving end in turn [21]. This leads to the fast path being idle. When PCDN transmits data, it adopts the method of sending request packets and replying to data packets. The late arrival of data packets transmitted by the slow path to the client will not cause blockage of the fast path, but the out-of-order data packets cannot be delivered to the player for playback, causing delay jitter affects user experience. In addition, compared to MPTCP which usually uses two paths for transmission, based on real deployment experience, a user's viewing demand typically requires support from 2 to 10 devices in PCDN. This inevitably increases the heterogeneity among paths, posing a greater difficulty in designing the scheduling mechanism [28]. To facilitate updates, the entire transmission protocol is implemented in user space based on UDP.

C. Typical Multipath Scheduling Mechanisms

Douyin deployed the default scheduler in the Linux system, minRTT, that distributes packets to the path with the smallest RTT, which cannot solve the above problems. Some advanced multipath scheduling mechanisms have been proposed in recent years, such as BLEST [18], ECF [19], XLINK [29]. BLEST and ECF will pause the slow path packet transmission when they predict that a HOL blocking will occur. XLINK uses the QoE feedback to calculate the remaining playback time and combines the data packet transmission time for re-injection operation, which reduces the request completion time and video rebuffer rate. They both sacrifice aggregate bandwidth to achieve improvements in other metrics.

III. Motivation

In this section, we discuss the limitations of existing multipath schedulers when applied to a PCDN deployment.

A. Aggregate Bandwidth and End-to-End Latency Trade-Offs

MPTCP schedulers fail to guarantee high throughput while minimizing end-to-end latency required by PCDN systems. MPTCP, similar to TCP, provides in-order and reliable data delivery to applications, which means that the receive buffer must temporarily store out-of-order data packets. Let's consider a scenario with one fast path and one slow path, where their respective bandwidths and RTTs are denoted as B_f and B_s, RTT_f and RTT_s. To avoid sacrificing aggregate throughput, the default minimum scheduler buffer size is [30]:

$$Buf_{agg} = (B_f + B_s) \times RTT_s \tag{1}$$

At this point, even though the fast path is not blocked for sending, it needs to wait for the arrival of packets from the slow path before delivery. During this period, the number of window rounds N that the fast path has to wait for is as shown in Eq.(2). The end-to-end delivery latency of packets in each round decreases by RTT_f compared to the previous round. So the sum of end-to-end delivery latency is computed as Eq.(4), CWND is short for the Congestion Window. Therefore, the average end-to-end delivery latency at this buffer size is given by Eq.(3). It can be observed that when RTT_f, B_f and B_s remain constant, the larger RTT_s, the higher the average end-to-end delivery latency.

$$N_{round} = RTT_s / RTT_f \tag{2}$$

$$L_{sum} = RTT_s \times CWND_s + \sum_{i=0}^{N-1} (RTT_s - i \times RTT_f)) \times CWND_f \tag{3}$$

$$\begin{aligned} L_{avg} &= L_{sum} / Buf_{agg} \\ &= RTT_s \times (1 - \frac{1}{2 \times RTT_f \times (B_f + B_s)}) \\ &\quad + \frac{1}{2 \times (B_f + B_s)} \end{aligned} \tag{4}$$

Then how about decreasing the buffer size to less than Buf_{agg}? If so, MPTCP schedulers cannot fully utilize the aggregate bandwidth. They will tend to block the slow path to fully utilize the fast path. At this point, the minimum achievable end-to-end delivery latency is RTT_f, but the aggregate bandwidth is also approximately equal to B_f [31]. Therefore, *MPTCP schedulers can only achieve a trade-off between high aggregate bandwidth and low end-to-end latency, but cannot simultaneously achieve both.*

We conducted experiments to demonstrate this phenomenon. Specifically, we replicated the representative BLEST and

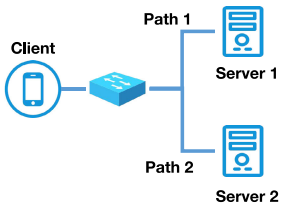
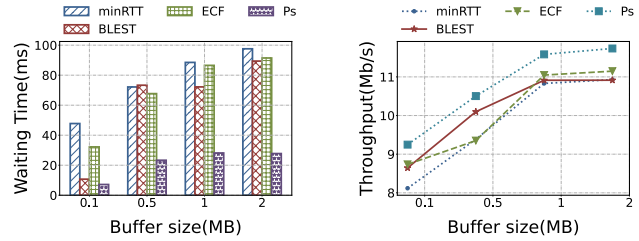


Fig. 4. Topology.



(a) Waiting time between a packet received and delivered.

(b) Aggregate bandwidth.

Fig. 5. Trade-off between aggregate bandwidth and latency of different schedulers under different receive buffer sizes.

ECF mechanisms in mininet [32] and integrated the PCDN transmission protocol environment into the mininet. The experimental topology, as shown in Fig.4, consists of a client connected to two servers via a router, with a bottleneck having a bandwidth of 30Mbps and a loss rate of 0.1%. The RTTs of Path 1 and Path 2 are 30ms and 90ms, respectively. In Fig.5, we present the results showing the average waiting time between a packet received and delivered per packet and throughput for different mechanisms with varying receiving buffer sizes. Notably, we observed a direct proportionality between the waiting time and the buffer size, while the throughput displayed an inverse proportionality to it. Indeed, Pscheduler (Ps) can perform well in both aspects. It effectively manages waiting time and optimizes throughput, achieving a good balance between the two objectives.

B. Challenges in Aggregating Bandwidth Using Complete CWND

To avoid wasting any bandwidth, it is imperative to leverage the complete CWND without impeding the progress of slower paths. However, achieving this objective poses a notable challenge—maintaining the sequential arrival of packets when utilizing the entire CWND. In PCDN, the client adopts a request-reply pull-based transmission mode that each data response only opens a single flow control window in one source server connection, which is commonly integrated in the client apps (e.g., Douyin and Xigua [4]). When a user downloads data from different devices, data is like rotating on conveyor belts (paths) of different lengths as shown in Fig.3. The longer the conveyor belt, the longer the path RTT. When the transmission is in a steady state, the data rotation speed is equal to the path bandwidth. For the path i, a data packet is received at each interval of $\frac{packet\_size}{bandwidth_i}$. As the packet size is fixed, the arrival times of data packets will vary for each path if the path bandwidths are different. With the PCDN

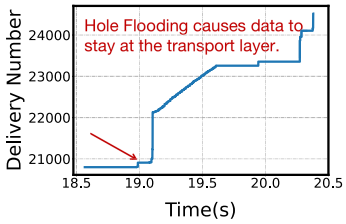


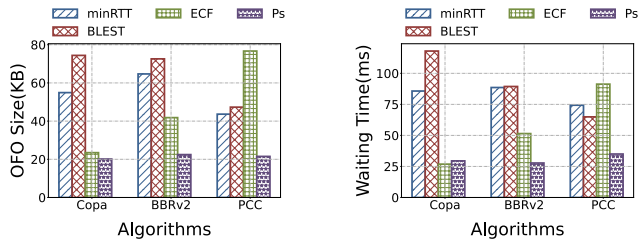
Fig. 6. The data shows a stepped delivery.

transmission feature, the client observes only one available transmission window across all paths each time it receives a data packet. Schedulers are triggered after receiving the packet, determining the path for sending subsequent packets. Without blocking the transmission, the next packet is sent to the path with an available window. Consequently, the client will distribute requests to different paths almost randomly, causing data request packets to be sent in order and data packets to be received out-of-order [27], can't be delivered.

For example, taking Fig.3 left, at a certain moment, the receive queue is queue a). At this time, packets 2 and 7 arrive in the queue, which can only deliver ordered data packets 1∼3 to the player. Packets after 3 are unordered and incomplete and cannot be played even if packet 7 arrives because there are holes between packets 3 and 7 (packet 3 is received from the path not shown in Fig.3, e. g., Path 4). With the increase in the number of paths, the heterogeneity among paths grows, and the correlation between packets arriving becomes smaller. The presence of holes between packets also becomes more frequent, which is referred to as "hole flooding". Out-of-order data in the receive buffer of the transport layer cannot be delivered to the upper layer, increasing end-to-end latency and jitter, and video playback may rebuffer (§ V). Fig.6 records the delivery of data over a period of time without Pscheduler in PCDN, highlighting the ladder-like delivery pattern caused by the hole flooding problem. In summary, to ensure smooth video playback under PCDN, the scheduler must effectively manage strict packet-level arrival ordering.

C. Coupled With CC Algorithms Cause Imprecise Path Information

To achieve strict packet arrival ordering, schedulers need to accurately predict the packet arrival time on different paths. However, existing multipath schedulers rely heavily on CC algorithms to estimate the transmission time of each path and often perform very differently under different CC algorithms. To investigate this issue, we conducted experiments in the same environment as before, with a receiving buffer size of 0.5KB. We choose to use some representative CC algorithms [33], [34], [35] like Copa [36], BBRv2 [37], and PCC [38], to transmit files. The results are shown in Fig.7. We take BLEST using the BBRv2 [39] algorithm as an example. During probing, BBRv2 deliberately increases the congestion window, which can mislead the scheduling algorithm into believing that the path is improving, resulting in more packets being sent on the slow path. Conversely, during



(a) The size of out-of-order data.

(b) Waiting time between a packet received and delivered.

Fig. 7. The performances of different schedulers under different congestion control algorithms.

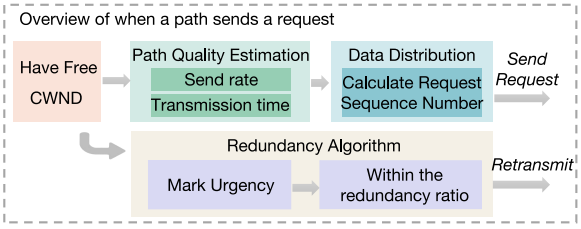


Fig. 8. The Overview of Pscheduler.

the emptying phase, BBRv2 intentionally reduces the CWND, causing the scheduling algorithm to send fewer packets on the slow path. Different CC algorithms have varying behaviors in network bandwidth detection and emptying, causing the scheduler performance to fluctuate with time. Such a scheduling mechanism is not CC-agnostic. When the transport protocol migrates to the user state, the CC algorithm can evolve quickly. Continually adapting the scheduling algorithm with different CCs would be labor-intensive and affect the advantages of the scheduling mechanism [40], [41].

IV. Design

The primary design concept of Pscheduler is to ensure smooth video playback by enabling data to arrive in an orderly manner in MS2OC heterogeneous scenarios. However, achieving this goal faces three main challenges. First, path quality is difficult to assess. Many heterogeneous paths make the network situation faced by the scheduler more complex and changeable. Furthermore, since a larger receiving buffer can solve the problem of holes, the orderly arrival of data is important. Evaluating network quality based solely on RTT and bandwidth is insufficient to guarantee accuracy. Continuous packet loss or packet loss on a large RTT path will cause the data to fail to arrive for a long time, or even miss the data delivery time when the bandwidth is sufficient. Second, the CC algorithm is challenging to decouple. Existing scheduling algorithms rely on the CWND or sending rate provided by the CC algorithm to estimate the data transmission time. However, the CC algorithm has a periodic dynamic balance process and a delay in path state perception. As a result, different CC algorithms can have varying impacts on the same scheduling algorithm. Thirdly, it is challenging to respond quickly to path fluctuations due to the time drifting. Path quality in PCDN is fluctuating. If the data request packets are distributed in advance to the corresponding path waiting to be sent out, sending packets in the original order may cause more serious disorder when the network fluctuates. However, the redundant transmission of data to deal with network fluctuations can result in the waste of bandwidth and make the insufficient PCDN transmission rate even worse.

The design of Pscheduler is centered around ensuring smooth video playback and cost-saving under PCDN. To meet these requirements, Pscheduler needs to achieve strict packet-level ordering while efficiently utilizing the throughput of all available paths.

A. Framework

As illustrated in Fig.8, Pscheduler comprises three key components to address the above challenges:

1) **Path Quality Estimation (§IV-B)** that is independent of CC algorithms to assess the quality of paths.

2) **Data Distribution (§IV-C)** to guarantee sequential data reception and adapt to dynamic networks.

3) **Redundancy Transmission (§IV-D)** algorithm for smooth video playback, which will be activated in exceptional situations to enhance user QoE.

Pscheduler calculates the packet sequence number that should be sent for the current path based on accurate path estimation information. Then, it adopts the "Queue-Direct-Access" approach to retrieve packets from the overall request queue and send them out. Video transmission is always segmented to avoid waste, and each segment is downloaded as a range. The redundancy algorithm determines if there is remaining space at the end of each range to accelerate the download process. In contrast to previous approaches, Pscheduler features a unique compute-aware forwarding redundancy mechanism that is only triggered in exceptional circumstances. In contrast to previous approaches, Pscheduler features a unique redundancy mechanism that is only triggered in exceptional circumstances but plays a critical role in guaranteeing smooth video playback.

B. CC-Decoupled Path Quality Estimation

To achieve packet-level arrival ordering in the packet distribution of Pscheduler, two variables of the network transmission paths are required to be obtained: the transmission rate and the end-to-end transmission time of sending a data packet along different paths.

1) Transmission Rate Estimation: Previous mechanisms calculate the transmission rate by dividing CWND by RTT, which is obtained from CC algorithm decisions. However, the CC algorithms are often in an exploration state and may not accurately represent the actual transmission rate (§III-C). To decouple from CC algorithms, Pscheduler employs a transmission rate calculation method that is independent of the CWND, as shown in Eq.(5):

$$R = \frac{datasize}{endtime - starttime} \tag{5}$$

where $datasize$ stands for the size of the received data, $starttime$ represents the time when the first packet is received in each interval, and $endtime$ is the time when the last packet is received in each interval. We calculate the rate at intervals

of every received CWND packet, although CWND is not a parameter in the rate calculation equation. This approach avoids the impact of transient behavior in CC algorithms, ensuring a more precise estimation of the connection's transmission rate.

2) Transmission Time Estimation: Pscheduler avoids directly using RTT to estimate the transmission time of packets. Instead, it calculates the average transmission time required by a path to successfully complete a data packet transmission as Eq.(6), including the parameter of packet loss ratio ($loss$). This approach prevents the unnecessary sending of packets on the paths with small RTT but high packet loss. The probability that a packet is successfully transmitted after being lost $n-1$ times is $loss^{n-1} \times (1-loss)$, and the transmission time is $((n-1) \times rto + rtt)$, $n \in N+$. Thus, the average transmission time of the data packet is:

$$E = \sum_{n=1}^{\infty}(loss^{n-1} \times (1-loss) \times ((n-1) \times rto + rtt))) \\ = rtt + rto \times \frac{loss}{1-loss} \tag{6}$$

where rto is the timeout retransmission time and $loss$ stands for the current packet loss rate. We use TFRC [42] to calculate the *loss* rate, store flight data packets in a queue, and use this queue to calculate the current packet loss rate.

In addition, due to the pull-based transmission mode of PCDN, both the sending of data request packets and the receiving of data packets are on the client side. The transmission time of a data packet will change from OWD (one-way delay, the server sends data to the client) to RTT (round-trip delay, client requests data and the server replies), which means that the clock synchronization problem between the sender and receiver will be eliminated in the path quality estimation of Pscheduler. Therefore, using a transmission time-based scheduling algorithm in PCDN to forward the packet will be more accurate.

C. Queue-Direct-Access Data Distribution

To achieve packet-level data scheduling instead of connection-level scheduling, Pscheduler requires greater flexibility. We consider two data distribution methods: "packets pick paths" and "paths pick packets". Ultimately, to enable the mechanism to flexibly combat network fluctuations and achieve strict arrival ordering, we choose the "paths pick packets" method.

1) Packets Distribution Based on Sub-Buffers (Packets pick Paths): To efficiently schedule request packets on demand, one approach is to employ individual pending task queue buffers for each path, as depicted in Fig.9a). Request packets are pre-allocated to the sub-buffer to rearrange their sending order and achieve the desired arrival order of data packets. However, this method lacks flexibility when the network experiences fluctuations, as the request packets in the sub-buffer may be out of order based on their pre-allocated sequence. Consequently, the pre-allocated packets cannot achieve strict packet-level arrival ordering, leading to a loss of flexibility.

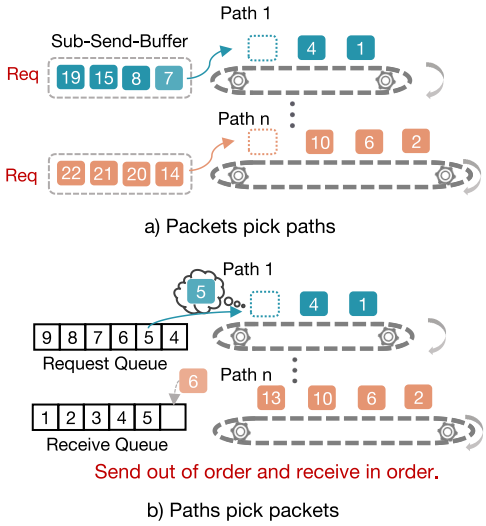


Fig. 9. Two packet distribution modes.

2) Queue-Direct-Access Packets Distribution (Paths Pick Packets): Pscheduler introduces a scheduling mode, *path-pick-packet* (Fig.9b), to enhance the flexibility of request packet transmission order. Unlike pre-allocation, this method dynamically selects request packets based on network quality, directly choosing packets from the overall queue for available path space. This process is akin to a rotating conveyor belt, where the queue learns which request packet to pick after each delivery, avoiding the need for a worker to pre-move packets in a fixed order. Despite only deciding one path's window per slot, Pscheduler accurately selects the specific packet, achieving strict packet-level arrival order.

Algorithm1 is the detailed design of the Queue-Direct-Access Packets Distribution algorithm. Whenever a path i has an available window, Pscheduler calculates the average transmission time E_i of the path according to Eq.(6) (line 3). Then traverse all connections and calculate their average transmission time E_j (lines 4-5). The smaller E means that the connection can complete the transmission of a data packet faster. Suppose the E_f of the first path is T, and the throughput is D; the E_s of the second path is $2T$, and the size of the data packet is S, then the arrival time of the data packet is $sendtime + E$. In $(E_s - E_f)$ time, the first path can send $\frac{T*D}{S}$ packets. The second path directly sends the packet whose position is $\frac{T*D}{S}+1$ in the task queue which can ensure that the data arrives in an orderly manner. At this time, the arrival time of the packets at positions $\frac{T*D}{S}$ and $\frac{T*D}{S}+1$ in the task queue is both $curtime+2*T$. Therefore, Pscheduler calculates offset according to Eq.(7) (lines 7-8). Then the second path sends the packet whose pos is $Offset$ in the total task queue. It can be observed that throughout the entire calculation process, the buffer size variable is not utilized. Therefore, Pscheduler is buffer-size agnostic and remains unaffected by buffer inflation.

$$Offset = \sum_{j \neq i}^{n} \frac{(E_i - E_j) \times R_j}{packetsize}(E_i > E_j) \tag{7}$$

Algorithm 1 Packet Distribution Algorithm

Input: connection i, connection set C
Output: $offset$

```
begin
    offset = 0
    calculate average packet transmission time of
    connection i E_i via Eq.(6)
    for j ∈ C and j ≠ i do
        E_j = rtt_j + rto_j × loss_j / (1 − loss_j)
        if E_j < E_i then
            calculate R_j via Eq.(5)
            Offset+ = (E_i − E_j) × R_j / packetsize
    return Offset
```

D. Redundancy Transmission

Pscheduler features a distinct redundancy algorithm, strategically triggered to enhance transmission reliability that doesn't waste the bandwidth. It considers data packet urgency, redundancy percentage, and task tail redundancy. In particular, tail redundancy is important as sporadic packets at the end of each range can limit the transmission of the next task segment.

1) Mark Packets Urgency: If the packet cannot arrive in time for its playback deadline, it will be marked as urgent, calculated as Eq.(8). Where $playtime_{remain}$ is the remaining time until the certain packet is played, and $sendtime$ is the time required to forward, as estimated as Eq.(6). The $threshold$ is the dangerous water level of the playback buffer. Once the remaining playback time is lower than the $threshold$, it is very likely to cause rebuffering. Considering that packets may be lost, Pscheduler adds a maximum path RTT above this threshold, so that even if packets are lost, there is a chance for retransmission.

$$playtime_{remain} - sendtime < threshold + RTT_{max} \quad (8)$$

2) Percentage of Redundancy: The second step is to ensure that redundant packets will not reduce user QoE by limiting the percentage between the number of redundant packets and the task download speed. We set a cap on the percentage of redundancy used. This approach guarantees that redundancy will not be activated when the bandwidth is insufficient to sustain the client's throughput requirements. Additionally, it ensures that the download speed of non-redundant data remains equal to or greater than the playback speed. Assuming the client's playback bitrate is F, the average download speed is $V_{arg} = \sum_i^n R_i$, $redundant\_data\_size$ the size of redundant data currently being transferred, $flight\_data\_size$ is the total data size being transmitted in the current network. Redundancy can be triggered only when $F < V_{arg}$, and the redundant data size should meet the following conditions:

$$\frac{redundant\_data\_size}{flight\_data\_size} \leq \frac{V_{arg} - F}{V_{arg}} \quad (9)$$

By satisfying these two criteria, a packet will only activate the redundancy algorithm if it is lost multiple times and the buffer data is depleted, striking a balance between redundancy and performance. Each redundant packet is vital in ensuring the minimum requirement, significantly decreasing the risk of stuttering.

3) Task Tail Redundancy: For segmented task transmission, Pscheduler designed a tail redundancy method. At the end of each range, there will be a situation where the number of remaining data packets is less than the total CWND. If the tail data packet is lost or does not reach the receiving end as expected due to network fluctuations, the link will appear in an idle waiting phase. In addition, if the previous task is not completed, the next range cannot transfer. But it is surprising that no existing works have addressed this critical problem. Therefore, once a connection i is detected to $offset > task\_size$, and $offset < 2 * task\_size$, which is used for redundant transmission. Each tail packet will only be redundant once.

V. Implementation and Evaluation

A. Implementation

We implement Pscheduler in the PCDN of Douyin. The scheduler will be triggered when the client needs to send a new "data request packet" by DataTimer(), Addnewtask(), or receiving a packet. DataTimer() is triggered every 100ms and will query the tracker for available nodes. Addnewtask() is called every time a new task is created. When the scheduling algorithm is triggered, the multipath scheduler will allocate request packets on the connections that have idle CWND and are still in the connection state.

B. Evaluation Setup

We verify the performance of Pscheduler by large-scale A/B tests from Douyin real users as well as simulations.

1) Online Large-Scale A/B test (§ V-C): We select approximately 100,000 Douyin users to participate in the Pscheduler experiment. Our analysis covers over 10 million video plays per day. To conduct this large-scale online test, we employ the A/B testing methodology, wherein two user groups simultaneously used Pscheduler and basic minRTT for one month. The basic minRTT is the method used in Douyin before Pscheduler. Our A/B tests show the benefits of Pscheduler in terms of both continuous and discontinuous speed improvements, as well as the reduction of data packet out-of-order, which directly impacts QoE. From a QoE perspective, we focus on reducing rebuffering rates and improving the upward delivery speed of data. Additionally, we pay attention to the mechanism's wastage rate. These metrics are the most important indicators to ensure Douyin's QoE, as users use their phones to play the packet (segment) from the PCDN, and processing the packets on their edge devices (phones) is fast. To test the impact of redundancy algorithms on the results, we evaluated two variants of Pscheduler: Pscheduler (the full version of Pscheduler) and Pscheduler-NR (Pscheduler without redundancy).

2) Simulator (§ V-D): Given the challenges associated with conducting detailed experiments in large-scale real-world environments, we implemented part of PCDN's transmission protocol in Mininet. This simulation platform allows us to

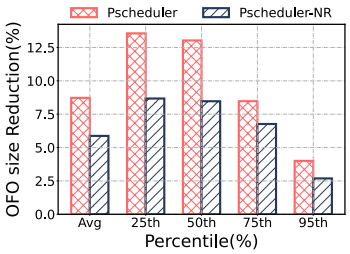


Fig. 10. [A/B test] The reduction in out-of-order data average size.

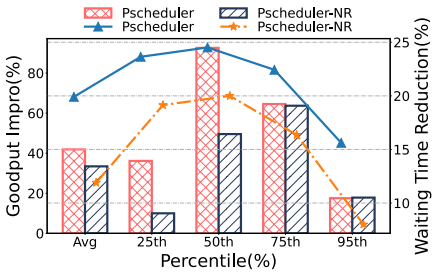


Fig. 11. [A/B test] Goodput vs. Data waiting time.

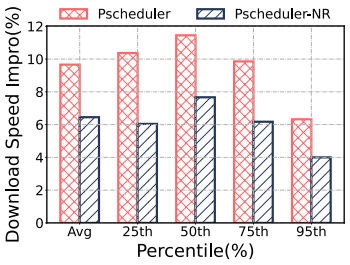


Fig. 12. [A/B test] The improvement in video download speed.

replicate various network environments and perform controlled experiments across different scenarios. By utilizing Mininet, we can manipulate network variables and analyze how different algorithms perform under conditions such as heterogeneous round-trip times (RTTs) and varying congestion control (CC) algorithms.

3) Choices of Baseline: Due to three reasons: 1) Pull streaming; 2) More paths; 3) Larger send/receive buffer (details in §I), most of the previous scheduling algorithms are not very good in MS2OC good run. For example, BLEST suspends the transmission of the slow path to avoid the situation that the fast path cannot send packets due to insufficient receiving buffer space. So the huge receiving buffer makes BLEST degenerate into minRTT. ECF distributes a certain amount of data packets (newly delivered tasks in the send buffer) to a path each time. In PCDN, a large number of data will be delivered at one time, which is equivalent to having a huge send buffer, resulting in the effect of ECF even worse than minRTT [31]. The results in Fig.5 are in line with our expectations, showing that the performance of traditional algorithms is similar to that of minRTT. In addition, XLINK was originally designed for two paths scenarios, and there will be ambiguity when expanding to more than two paths scenarios. Therefore, we choose the original algorithm minRTT as the comparison algorithm. At the same time, we tested two variants of Pscheduler: Ps (Pscheduler full version) and Ps-nr (Pscheduler without redundancy).

C. Large-Scale A/B Test

1) Out-of-order Data Size: The metric of out-of-order data size is the most straightforward indicator of the impact of scheduling mechanisms. After receiving each packet, we measured the current size of out-of-order data and calculated the average. Fig.10 demonstrates that Pscheduler effectively reduces the average amount of out-of-order data by more than 20% compared to the baseline mechanism.

2) Goodput and Data Waiting Time: Fig.11 presents the goodput improvement and data waiting time reduction achieved by Pscheduler at different percentiles compared to the minRTT mechanism. Goodput is defined as the speed of data delivered for playback. In other words, it is calculated as the data in order divided by the duration between the beginning of the range downloading and receiving half the size of the range. The data waiting time metric measures the time between a packet being received and its delivery upward for playback. Goodput and data waiting time are interrelated. If data remains in the buffer for a prolonged period but cannot be played due to out-of-order issues, the goodput will decrease, increasing the likelihood of buffering and stuttering occurrences. Pscheduler efficiently arranges data in order, resulting in a 20%∼80% increase in goodput and 15%∼25% reduction in waiting time, which effectively prevents rebuffering.

3) Video Download Speed: We measure video download speeds for each range being downloaded. Fig.12 shows the speed improvement of Pscheduler and Pscheduler-NR compared to the original algorithm (minRTT) at different percentiles. Pscheduler-NR shows an average improvement of 6% in video download speed, while Pscheduler achieves an average improvement of 10%. At the 95th percentile, Pscheduler still exhibits a 6% enhancement. As video downloads are often segmented based on ranges, with short flows being predominant, the selection of data packet transmission paths becomes crucial, making Pscheduler effective in boosting video download speeds. However, compared to continuous video download speeds (Goodput), the improvement in download speeds may not appear as significant. This is because the primary role of the scheduler is to enhance data ordering, which has already been demonstrated through the increase in continuous download speed and the reduction in rebuffering rates, validating the effectiveness of the algorithm.

4) Video Jump Rate: The rebuffer rate is a crucial metric influencing video Quality of Experience (QoE) and video smoothness. In Douyin's PCDN, rebuffering triggers a switch from PCDN to CDN as a backup. We use the online jump rate as a measure of PCDN transmission failures leading to CDN transition, calculated by dividing jump duration by total playback time. Fig.13 displays the average jump rate throughout the day, comparing Pscheduler to the minRTT

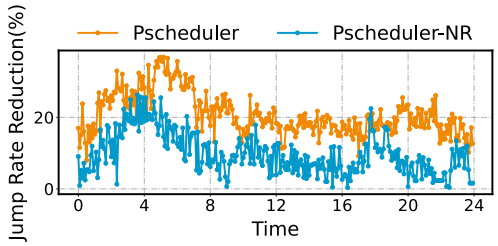
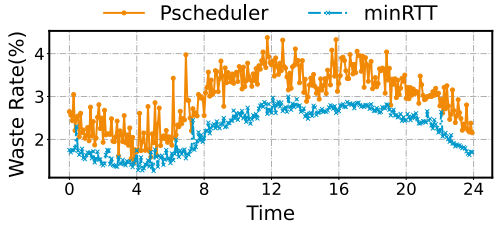


Fig. 13. [A/B test] The reduction in the rate of videos that jump out to the CDN network due to rebuffering.



Waste(%) / Alg \ Hour	[0, 4)	[4, 8)	[8, 12)	[12, 16)	[16, 20)	[20, 24)
Ps	2.17	2.27	3.39	3.44	3.46	2.79
minRTT	1.54	1.61	2.43	2.65	2.66	2.14

Fig. 14. [A/B test] The waste rate of Pscheduler and minRTT within a day.

mechanism. Pscheduler effectively reduces the jump rate by up to 30%.

5) Data Waste Rate: In Fig.14, the data waste rate for a specific day is presented. This rate is calculated by taking the difference between the number of requests sent and the total number of packets in a task, divided by the total number of packets in the task. The waste rate accounts for both packet loss and redundancy. Pscheduler exhibits a slightly higher waste rate than minRTT due to its redundancy algorithm. However, the increase is only 0.74%, while achieving a 10% improvement in video download speed. This suggests that Pscheduler's redundancy strategy employs a minimal amount of redundant data but plays a crucial role.

D. Simulation Evaluation

1) Simulation Setup: We implement Pscheduler and baseline into the PCDN system. In simulation experiments, we use Mininet to simulate different network environments to conduct controlled experiments. The utilization of Mininet allows us to establish diverse network scenarios, varying path numbers, and introducing heterogeneous RTT. This setup facilitates a comparative analysis of different schedulers under varying conditions. Pscheduler demonstrate superior results, achieving higher aggregate throughput and lower packet waiting time across different path numbers. Unless otherwise specified, the CC algorithm defaults to BBR.

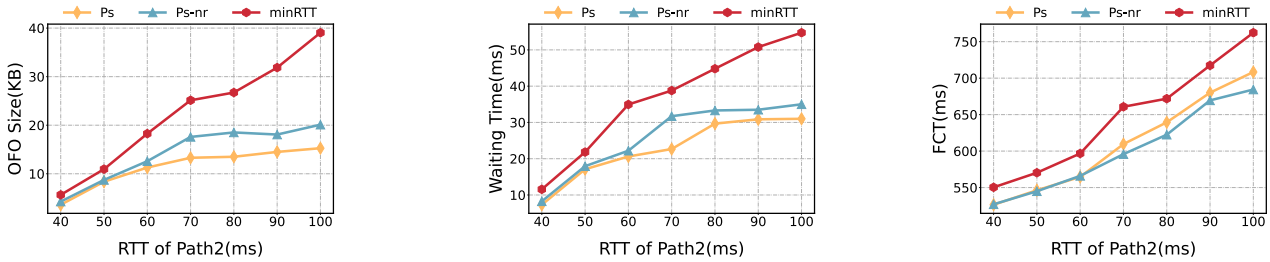
2) Experiment Results: We provide simulation results in this section.

a) Scenario 1: RTT heterogeneity of paths: We first tested the robustness of the Pscheduler on heterogeneous paths. The RTT heterogeneity of the path is one of the main reasons for the disorder. For the convenience of illustration, we first do the scenario of using two paths for transmission. From Fig. 15a and Fig. 15b, it can be seen that as the RTT of the second path increases, the heterogeneity between the two paths increases, and the degree of disorder of the minRTT algorithm also increases almost linearly. Pscheduler estimates the order of data arrival based on transmission time and bandwidth, making the degree of data disorder always kept at a low level. Compared with minRTT, OFO size is reduced by 23.6-61.0%, and wait time is reduced by 21.7-43.3%. Since the client downloads multiple small tasks serially, as the RTT of Path2 increases, it becomes more and more difficult for the throughput to reach the network's upper limit in a short period of time. Hence, the average task completion time gradually increases. Fig. 15c shows that in the overall increasing trend of task completion time, Pscheduler-nr and Pscheduler are always much lower than minRTT, reducing the completion time by 4.3-7.1%. However, since this scenario does not have a loss rate, there is no need to send redundant packets. Sending redundant packets unnecessarily lengthens the FCT slightly.

b) Scenario 2: Dynamic RTT with loss rate: This section examines the algorithm's performance in a dynamic RTT environment. The x-axis in the figure represents the maximum RTT jitter of the two paths. Fig.16 shows that the jitter of the smaller RTT path does not cause minRTT to generate more out-of-order packets, while the jitter of the larger RTT path causes the degree of out-of-order packets to increase rapidly. This is because the generation of out-of-order data packets is mainly caused by the fact that the data packets on the large RTT path arrive at the receiving end late. The jitter of the small RTT path actually causes the OFO data packets to be OFO and has no effect on out-of-order indicators. Only when the data packets on the large RTT path arrive in order, the data in the receive buffer can be delivered in order, and the degree of disorder will increase inevitably once the jitter is messed up. Pscheduler's "paths pick packets" algorithm more flexibly adjusts the packet transmission order, maintaining good performance even in the presence of RTT jitter in the path.

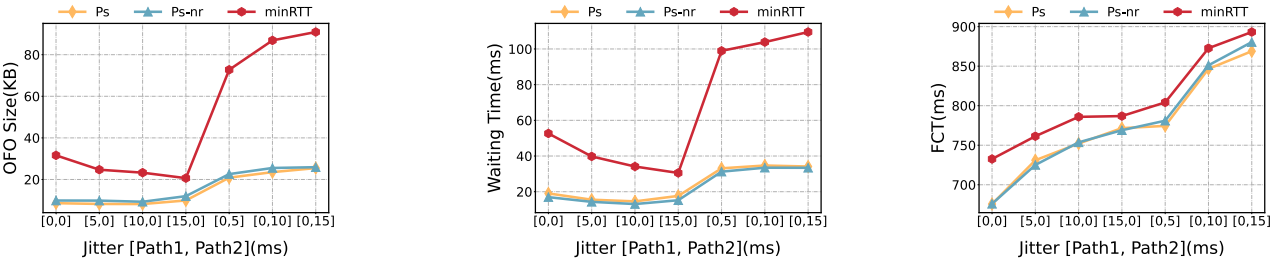
c) Scenario 3: Adaptability to different CC algorithms: Fig.17 records the performance of different scheduling algorithms under different CC algorithms. We reproduce three typical CC algorithms BBR, COPA [36] and PCC [43] in PCDN's multipath transmission protocol, and run minRTT, Ps and Ps-nr scheduling algorithms on these three algorithms respectively. Congestion packet loss caused by different CC algorithms' mechanisms and different queuing delays leads to different degrees of out-of-order transmission. From the motivation experiments, it can be seen that the performance of traditional scheduling algorithms varies greatly under different CC algorithms, even worse than minRTT. But the independent bandwidth estimation mechanism makes Pscheduler perform well under different CC algorithms. Compared with minRTT, the OFO size of Pscheduler is reduced by more than 10%, the waiting time is reduced by more than 30%, and the completion time is reduced by about 10%.

d) Scenario 4: Different number of parallel transfer paths: It can be known from motivation 2.2.1 that the more paths there are, the more serious the hole flooding problem



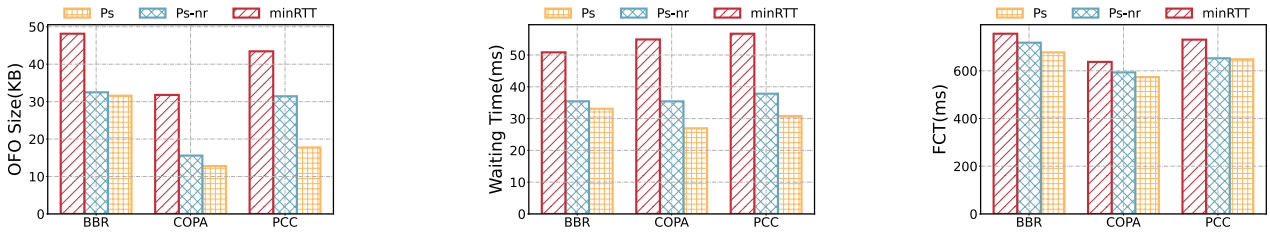
(a) The Size of Out-of-order Data. (b) Waiting Time. (c) Flow Complete Time.

Fig. 15. [Simu] RTT heterogeneity of Paths. The bottleneck bandwidth of the two paths is 10Mbps without loss rate. The delay of the first path is fixed at 30ms, and the delay of the second path is gradually increased from 40ms to 100ms with a change interval of 10ms.



(a) The Size of Out-of-order Data. (b) Waiting Time. (c) Flow Complete time.

Fig. 16. [Simu] Dynamic RTT with loss rate. The bottleneck bandwidth of the two paths is 10Mbps with 0.1% loss rate. The delay of two paths is 30ms and 90ms. The delay of the path is jittered according to the x-axis.



(a) The Size of Out-of-order Data. (b) Waiting Time. (c) Flow Complete Time.

Fig. 17. [Simu] Adaptability to different CC algorithms. The bottleneck bandwidth of the two paths is 10Mbps without loss rate, the delay of two paths are 30ms and 90ms.

will be. Fig.18 records the out-of-order degree and completion time of different scheduling algorithms tested under 2-5 paths. With an increase in the number of heterogeneous paths, the degree of out-of-order transmission and the completion time show a linear increasing trend due to the more severe problem of hole flooding, when using minRTT. However, it can be seen from the figure that as the number of paths increases, the degree of confusion of Pscheduler only increases slightly. Moreover, as the number of paths increases, the total bandwidth of the connection increases, so the task completion time decreases gradually. Compared to minRTT, Pscheduler reduces completion time by an average of 2%.

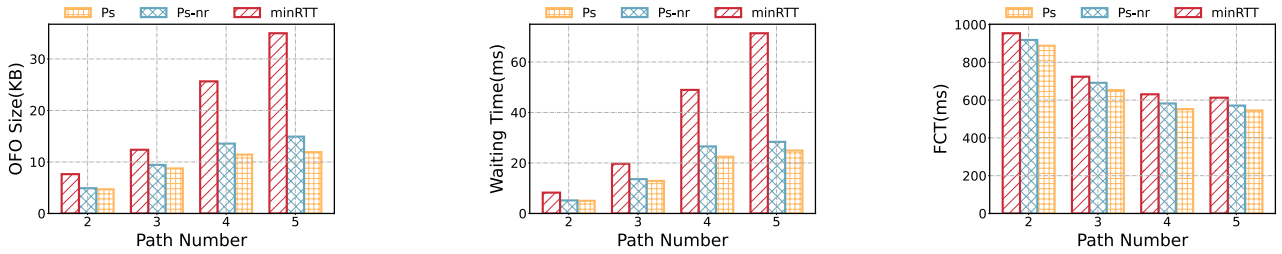
VI. Related Work

A. Multipath Transmission Protocols

Multipath transmission protocols have evolved as the number of device network interfaces increases [13], [15], [44]. MPTCP [15] and MPQUIC [16] respectively expand the single-path transmission protocol based on TCP [45], UDP to an end-to-end multipath transmission protocol to increase the transmission rate and robustness. PCDN is a multiple-server-to-one-client transmission protocol that makes up for the lack of single-path transmission performance.

B. Multipath Data Forwarding Algorithms

There are currently scheduling algorithms designed for MPTCP. MPTCP initially uses the RoundRobin [46] algorithm, which traverses all subflows and sends data as long as the sub-flow has a free CWND. Currently, in the Linux kernel, the default scheduling algorithm of MPTCP is the minRTT algorithm, which prioritizes data scheduling to paths with small RTTs so that subflows with good path quality can send more data. Otias [47], ECF [19], and BLEST [18] will maximize the utilization on the fast path and block the slow path to prevent HoL. In order to improve the utilization of the fast path, they will cause the slow path to idle, reducing the aggregation throughput to a certain extent. In addition, some works ensure the orderly transmission of data through



(a) The Size of Out-of-order Data. (b) Waiting Time. (c) Flow Complete Time.

Fig. 18. [Simu] The different number of parallel transmission paths. The path bottleneck bandwidth is 5Mbps without loss rate, the minimum path delay is 30ms, and the delay of each additional path increases by 10ms.

redundancy [48]. Redundant selects a path with the smallest RTT as the main path to transmit data and transmits repeated data packets on other subflows. TWC [49] restricts the sending of redundant packets by estimating the arrival time range of data packets. Nevertheless, TWC's bandwidth waste is still serious. XLINK [29] further limits the number of redundant packets through double thresholds. However, the above algorithms all work on MPTCP and MPQUIC, which have only two paths in most cases, and focus on solving the HoL problem. Pscheduler takes into account the unique challenges of streaming multipath parallel transmission in PCDN, ensures smooth video playback through precise compute-aware scheduling at the packet level and meets the sophisticated demands on the network for better quality of application users.

C. Network Architecture

CDN, content distribution network, caches video data on servers closer to users. With the increase in bandwidth demand, the construction cost of servers, switches, etc. has also risen sharply. P2P [50], [51] is a peer-to-peer transmission network architecture, which can be divided into tree-based, grid-based, and hybrid tree-grid-based data-forwarding systems. Although P2P networks have higher scalability and lower deployment costs, dynamic nodes also cause instability and low performance due to insufficient participating nodes. PCDN combines the two to build a high-quality and low-cost video distribution network.

VII. Conclusion

With the increasing costs of traffic, the demand for more cost-efficient video distribution has led to the transition from traditional CCDNs to PCDNs. In this paper, we present Pscheduler, a novel MS2OC scheduling mechanism tailored specifically for PCDNs. Pscheduler adopts a fine-grained packet-level scheduling approach, with the primary goal of optimizing the aggregate bandwidth while simultaneously minimizing end-to-end latency, thus enhancing the overall video QoE. Our evaluation demonstrate that Pscheduler effectively reduces data disorder in the MS2OC scenario and significantly improves video goodput.

References

[1] *Cisco Annual Internet Report (2018–2023) White Paper*, Cisco, San Jose, CA, USA, 2020.

[2] S. Nacakli and A. M. Tekalp, "Controlling P2P-CDN live streaming services at SDN-enabled multi-access edge datacenters," *IEEE Trans. Multimedia*, vol. 23, pp. 3805–3816, 2021.

[3] *Douyin Web Version*. Accessed: May 30, 2024. [Online]. Available: https://www.douyin.com/

[4] *Xigua Web Version*. Accessed: May 30, 2024. [Online]. Available: https://www.ixigua.com/

[5] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution," *Multimedia Syst.*, vol. 11, no. 4, pp. 383–399, Apr. 2006.

[6] K. Li, W. Zhou, S. Yu, and Y. Zhang, "A resource-search and routing algorithm within PCDN autonomy area," in *Proc. 8th Int. Conf. Parallel Distrib. Comput., Appl. Technol. (PDCAT)*, Dec. 2007, pp. 509–514.

[7] H. Yin et al., "LiveSky: Enhancing CDN with P2P," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 6, no. 3, pp. 1–19, Aug. 2010.

[8] C. Hu, M. Chen, C. Xing, and B. Xu, "EUE principle of resource scheduling for live streaming systems underlying CDN-P2P hybrid architecture," *Peer-Peer Netw. Appl.*, vol. 5, no. 4, pp. 312–322, Dec. 2012.

[9] E. Baccaglini, M. Grangetto, E. Quacchio, and S. Zezza, "A study of an hybrid CDN–P2P system over the PlanetLab network," *Signal Process., Image Commun.*, vol. 27, no. 5, pp. 430–437, May 2012.

[10] G. Peng, "CDN: Content distribution network," 2004, *arXiv:0411069*.

[11] V. K. Adhikari et al., "Unreeling netflix: Understanding and improving multi-CDN movie delivery," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1620–1628.

[12] Y.-C. Chen, Y.-S. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of MultiPath TCP performance over wireless networks," in *Proc. Conf. Internet Meas. Conf.*, Oct. 2013, pp. 455–468.

[13] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian, and S. Sen, "Accelerating multipath transport through balanced subflow completion," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2017, pp. 141–153.

[14] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, Oct. 2006.

[15] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," Internet Eng. Task Force, USA, Tech. Rep. RFC 8684, 2013.

[16] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and evaluation," in *Proc. 13th Int. Conf. Emerg. Netw. Experiments Technol.*, Nov. 2017, pp. 160–166.

[17] J. Hwang and J. Yoo, "Packet scheduling for multipath TCP," in *Proc. 7th Int. Conf. Ubiquitous Future Netw.*, Jul. 2015, pp. 177–179.

[18] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proc. IFIP Netw. Conf.*, May 2016, pp. 431–439.

[19] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proc. 13th Int. Conf. Emerg. Netw. EXperiments Technol.*, Nov. 2017, pp. 147–159.

[20] M. Scharf and S. Kiesel, "NXG03–5: Head-of-line blocking in TCP and SCTP: Analysis and measurements," in *Proc. IEEE Globecom*, Nov. 2006, pp. 1–5.

[21] S. Ferlin, T. Dreibholz, and Ö. Alay, "Multi-path transport over heterogeneous wireless networks: Does it really pay off?" in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 4807–4813.

[22] L. De Vito, S. Rapuano, and L. Tomaciello, "One-way delay measurement: State of the art," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 12, pp. 2742–2750, Dec. 2008.

[23] D. Wei et al., "Pscheduler: QoE-enhanced MultiPath scheduler for video services in large-scale peer-to-peer CDNs," in *Proc. IEEE Conf. Comput. Commun.*, May 2024, pp. 2508–2517.

[24] T.-M. Pham, M. Minoux, S. Fdida, and M. Pilarski, "Optimization of content caching in content-centric network," Sorbonne Universities, Paris, France, Tech. Rep., 2017.

[25] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *ACM SIGOPS Operating Syst. Rev.*, vol. 44, no. 3, pp. 2–19, Aug. 2010.

[26] F. Chen, K. Guo, J. Lin, and T. La Porta, "Intra-cloud lightning: Building CDNs in the cloud," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 433–441.

[27] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, Ö. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 302–315, Feb. 2019.

[28] G. Zhang, W. Liu, X. Hei, and W. Cheng, "Unreeling xunlei kankan: Understanding hybrid CDN-P2P video-on-demand streaming," *IEEE Trans. Multimedia*, vol. 17, no. 2, pp. 229–242, Feb. 2015.

[29] Z. Zheng et al., "XLINK: QoE-driven multi-path QUIC transport in large-scale video services," in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 418–432.

[30] C. Raiciu et al., "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proc. NSDI*, Apr. 2012, p. 29.

[31] H. Shi et al., "STMS: Improving MPTCP throughput under heterogeneous networks," in *Proc. ATC*, Jul. 2018, pp. 719–730.

[32] *Mininet*. Accessed: May 30, 2024. [Online]. Available: http://mininet.org/

[33] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[34] T. Gilad, N. Rozen-Schiff, P. B. Godfrey, C. Raiciu, and M. Schapira, "MPCC: Online learning multipath transport," in *Proc. 16th Int. Conf. Emerg. Netw. Experiments Technol.*, Nov. 2020, pp. 121–135.

[35] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," Internet Eng. Task Force, USA, Tech. Rep. RFC6356, 2011.

[36] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *Proc. Appl. Netw. Res. Workshop*, Jul. 2018, pp. 329–342.

[37] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[38] D. Mo, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. NSDI*, 2015, pp. 395–408.

[39] N. Cardwell et al., "BBRv2: A model-based congestion control performance optimization," in *Proc. IETF Meeting*, 2019, pp. 1–32.

[40] A. Nikravesh, Y. Guo, X. Zhu, F. Qian, and Z. M. Mao, "MP-h2: A client-only multipath solution for HTTP/2," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2019, pp. 1–16.

[41] Q. De Coninck and O. Bonaventure, "Observing network handovers with multipath TCP," in *Proc. ACM SIGCOMM Conf. Posters Demos*, Aug. 2018, pp. 54–56.

[42] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," Internet Eng. Task Force, USA, Tech. Rep. RFC3448, 2003.

[43] D. Mo et al., "PCC vivace: Online-learning congestion control," in *Proc. NSDI*, Jan. 2018, pp. 343–356.

[44] Q. Coninck and O. Bonaventure, *Multipath Extensions for QUIC (MP-QUIC)*, document Internet Draft draftdeconinck-quic-multipath-04, IETF, Individual Submission, 2020.

[45] B. A. Forouzan, *TCP/IP Protocol Suite*. USA: McGraw-Hill, 2002.

[46] M. F. Imaduddin, A. G. Putrada, and S. A. Karimah, "Multipath TCP scheduling performance analysis and congestion control on video streaming on the MPTCP network," in *Proc. Int. Conf. Softw. Eng. Comput. Syst. 4th Int. Conf. Comput. Sci. Inf. Manage. (ICSECS-ICOCSIM)*, Aug. 2021, pp. 562–567.

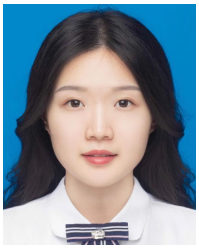
[47] F. Yang, Q. Wang, and P. D. Amer, "Out-of-Order transmission for in-order arrival scheduling for multipath TCP," in *Proc. 28th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, May 2014, pp. 749–752.

[48] A. Frommgen, T. Erbshäußer, A. Buchmann, T. Zimmermann, and K. Wehrle, "ReMP TCP: Low latency multipath TCP," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.

[49] Y. Xing et al., "A low-latency MPTCP scheduler for live video streaming in mobile networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7230–7242, Nov. 2021.

[50] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-VoD system," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 375–388, Oct. 2008.

[51] J. Lei, L. Shi, and X. Fu, "An experimental analysis of Joost peer-to-peer VoD service," *Peer-Peer Netw. Appl.*, vol. 3, no. 4, pp. 351–362, Oct. 2010.



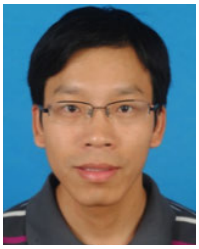
Dehui Wei (Member, IEEE) received the B.E. degree in computer science and technology from Hunan University, Changsha, China, in 2019. She is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT). From June 2023 to June 2024, she was a Visiting Student with the School of Computing, National University of Singapore. Her research interests are in the areas of network transmission control and cloud computing. She received the Outstanding Graduate Award for the B.E. degree.

Jiao Zhang (Senior Member, IEEE) received the bachelor's degree from the School of Computer Science and Technology, Beijing University of Posts and Telecommunications (BUPT), China, in July 2008, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, China, in July 2014. She is currently a Professor with the School of Information and Communication Engineering, BUPT. From August 2012 to August 2013, she was a Visiting Student with the Networking Group of ICSI, UC Berkeley. Her research interests include data center networking, network function virtualization, and future internet architecture.

Xiang Liu (Graduate Student Member, IEEE) was born in Shandong, China, in 1995. He received the Bachelor of Engineering degree in communication engineering from Beijing University of Posts and Telecommunications in 2018. He is currently pursuing the Ph.D. degree with the School of Computing, National University of Singapore. His major field of study has been focused on artificial intelligence systems, data science, and computer systems. In terms of accolades, he has received the National Scholarship twice and has also been awarded a special scholarship. He has published papers in top-tier conferences and journals, such as PSB, NeurIPS, BIBM, CVPR, VTC, and Methods.

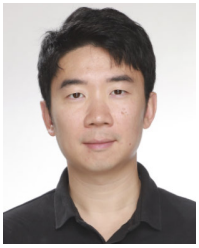
Haozhe Li received the B.E. degree in communication engineering from Wuhan University of Technology, Wuhan, China, in 2020, and the M.S. degree from the State Key Laboratory of Networking and Switching Technology, Beijing University of Post and Telecommunications (BUPT), in 2023. He is currently a Staff Engineer with ByteDance Inc. His research interests are in areas of data center networks and network congestion control.

Zhichen Xue received the M.S. degree in computer science from Illinois Institute of Technology, Chicago, IL, USA, in 2021. After graduation, he joined Bytedance Ltd., Beijing, China, as a Software Development Engineer. His current research interests include high-performance network programming, multi-path transport, and network simulation.



Tao Huang (Senior Member, IEEE) received the B.S. degree in communication engineering from Nankai University, Tianjin, China, in 2002, and the M.S. and Ph.D. degrees in communication and information system from Beijing University of Posts and Telecommunications, Beijing, China, in 2004 and 2007, respectively. He is currently a Professor with Beijing University of Posts and Telecommunications. His current research interests include network architecture, routing and forwarding, and network virtualization.

Linshan Jiang received the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2022. He is currently a Research Fellow with the Institute of Data Science, National University of Singapore. His research interests focus on the privacy and security in the CPS-IoT and distributed AI systems, including federated/collaborative learning, blockchain-enabled AI, and resilient AIoT systems. He has published papers in top-tier conferences and journals, such as IEEE INTERNET OF THINGS JOURNAL, EWSN, SenSys, ICDCS, TOSN, TIOT, ICC, ICCS, ICPDS, and IJCAI.



Jialin Li received the bachelor's degree from the University of Michigan and the Ph.D. degree from the University of Washington. He is currently the Sung Kah Kay Assistant Professor with the School of Computing, National University of Singapore. His research interests are in distributed systems and operating systems. His research has been awarded with the OSDI '14 Jay Lepreau Best Paper Award and the NSDI '15 Best Paper Award.