Enhancing Resource Management of the World's Largest PCDN System for On-Demand Video Streaming

Rui-Xiao Zhang, *UIUC;* Haiping Wang, Shu Shi, Xiaofei Pang, Yajie Peng, and Zhichen Xue, *ByteDance;* Jiangchuan Liu, *Simon Fraser University*

https://www.usenix.org/conference/atc24/presentation/zhang-rui-xiao

This paper is included in the Proceedings of the 2024 USENIX Annual Technical Conference.

July 10–12, 2024 • Santa Clara, CA, USA

978-1-939133-41-0

Open access to the Proceedings of the 2024 USENIX Annual Technical Conference is sponsored by

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Enhancing Resource Management of the World's Largest PCDN System for On-Demand Video Streaming

Rui-Xiao Zhang* *UIUC*
Haiping Wang† *ByteDance*
Shu Shi *ByteDance*
Xiaofei Pang *ByteDance*
Yajie Peng *ByteDance*
Zhichen Xue *ByteDance*
Jiangchuan Liu† *Simon Fraser University*

Abstract

The rapid growth of video services has led to the significant requirement for efficient content delivery. Traditional approaches mainly rely on Content Delivery Networks (CDNs), which unfortunately incur significant bandwidth cost for video providers. To resolve this problem, the cost-efficient edge resources have emerged as a new solution to replace CDNs. However, their heterogeneous hardware and poor performance still present challenges in their effective utilization. In this paper, we present how ByteDance explores the use of these cost-efficient but less performant resources. Specifically, we first present an extensive overview of PCDN, ByteDance's alternative delivery network for CDNs. Second, as PCDN encounters significant resource imbalances after years of deployment, we further introduce PCDN$^+$, the enhanced iteration of PCDN. Specifically, by integrating a well-designed centralized/decentralized framework, we evolve previous "static" and "uncontrolled" PCDN into a "dynamic" and "controlled" system. The extensive A/B test and real-world deployment have demonstrated that PCDN$^+$ 1) effectively alleviates overloading issues, 2) significantly improves the utilization of low-cost resources, 3) provides higher service speed.

1 Introduction

With the rapid development of video generation software/hardware and networking infrastructure, recent years have witnessed the booming growth of video-based services. As reported by *The 2022 Global Internet Phenomena Report* [35], video traffic has taken over 65.93% of all Internet traffic in the first half of 2022, showing a 24% increase compared to the same period in 2021. Traditionally, to maintain user engagement, the video platforms will utilize Content Delivery Networks (CDNs) to help distribute video content [1, 29, 44]. While this de-facto industry standard solution provides a lot of benefits (i.e., stable bandwidth, reliable service, unrestricted scalability, etc.), it also comes with a high price tag. The cost of building and maintaining data centers/clusters for content delivery has gradually become a hard-to-afford burden for large video streaming providers, especially those who provide free streaming services to users. E.g.,, a publicly listed leading short video streaming platform in China revealed in their financial report that the cost of bandwidth and servers accounts for up to 10% of their total revenue, which is much larger than all other costs except the revenue share [15].

As the largest VoD (video on-demand) content provider in China, ByteDance operates Douyin [17] that attracts over 750 million daily active users [18]. Given the massive viewer number and continuously increasing video quality, it is imperative to effectively reduce bandwidth cost to control expenses. Starting in 2020, ByteDance began exploring the use of edge nodes as a partial replacement for CDNs. These nodes mainly consist of idle computing resources on the Internet (e.g., smart home devices and outdated servers), and they are collected by third-party companies and sold in bundles to us at a heavily discounted price compared to CDNs. While these resources offer sufficient storage and network connectivity, they do not guarantee computation capability, system reliability, or service availability. Therefore, the technical challenge for us is how to make use of these cost-efficient but less-performant resources.

We tackle this challenge by building up a customized delivery system called PCDN. PCDN is composed of a server side and a client side. The server side manages the edge resources and is responsible for answering the user's data requests and returning available edge nodes; while the client side manages the data downloading process after receiving the server-side response. Specifically, PCDN leverages a *multi-source parallel downloading* (MPD) mechanism [16]: i.e., the user will simultaneously download data from multiple edge nodes. By pairing each user with multiple sources, the MPD is able to compensate for the unreliability and low performance of any single node. Through the above designs, PCDN has been continuously running and hosting large quantity of Douyin traffic,

*This work was finished before Rui-Xiao Zhang joined UIUC

†Haiping Wang (wanghaiping.paloma@bytedance.com) and Jiangchuan Liu (jcliu@sfu.ca) are corresponding authors.

and achieves substantial cost savings.

However, with more viewers being served and more heterogeneous edge nodes included, we find our PCDN system is facing a significant resource unbalance problem: i.e., plenty of edge resources are overloaded, while a significant number of resources are still underutilized. This resource utilization problem has greatly deteriorated the efficiency of the system: on the one hand, we see that the overloaded servers will suffer performance degradation, thereby negatively impacting user engagement; on the other hand, we also find that the low-price resources are more likely to be under-utilized, which leaves large room for cost reduction. After large-scale measurement, we have identified the fundamental limitations of the aforementioned issues, which are 1) static allocation on the server side and 2) uncontrolled downloading process on the client side. The server-side resource allocation relies on fixed rules (i.e., preferring same-region and high-price nodes), leading to the frequent selection of specific nodes and their subsequent overloading. Meanwhile, the client-side downloading process operates in a "free-competition" manner, resulting in random resource consumption and further separating PCDN from on-demand resource allocation.

To tackle the above problems, we have been diligently working on optimizing the existing PCDN system for the past two years. Specifically, we propose PCDN$^+$, the next iteration of PCDN. The core idea is to evolve PCDN from being "static" on the server side and "uncontrolled" on the client side to a "dynamic" and "controlled" system. Specifically, PCDN$^+$ is featured with the following designs.

First, to address the issues caused by the static PCDN server side, we propose to enhance it with a dynamic centralized logic. Specifically, this centralized logic facilitates two properties: on the one hand, by well leveraging the information from both users and resources, it can dynamically adapt the resource allocation strategy, and optimizes the overall system from a global view; on the other hand, by modeling the decision-making process as an optimization problem, it also provides control knobs to operators to flexibly balance cost and quality.

Second, to address the issues of the uncontrolled client side, we propose to augment it with a controlled decentralized logic. This is accomplished through a well-designed joint bandwidth allocation algorithm. Specifically, by interacting with the transport layer and actively optimizing the node resources based on network conditions, it enables on-demand resource utilization without negatively impacting performance. The coordination between the centralized logic and decentralized logic finally facilitates effective management of edge resources, leading to improved overall performance. In summary, our contributions are as follows:

- We first present a comprehensive overview of the edge resources in ByteDance, and present their critical features. After that, we provide a detailed introduction to PCDN, Bytedance's edge-based delivery network.
- We find that PCDN system is faced with the problem of unbalanced resource utilization, which is caused by static decision-making on the server side, and the uncontrolled data downloading on the client side.
- We propose PCDN$^+$, the next iteration of PCDN to address the aforementioned issues. Specifically, by integrating a dynamic centralized logic into the PCDN server side and a controlled decentralized logic into the PCDN client side, we achieve flexible and improved resource management.
- We evaluate PCDN$^+$ through large-scale A/B tests and real-world deployment. Through extensive experiment, we demonstrate that PCDN$^+$ 1) effectively addresses overloading issues; 2) significantly improves the utilization of low-cost resources; and 3) obtains higher speed. The fast improvement of shared traffic after fully deployment also demonstrate the effectiveness of PCDN$^+$

All user-related data is anonymous. This work does not raise any ethical issues.

2 Background

Traditional video services are supported by CDNs. The CDN is a large-scale distribution system composed of a large storage server (also known as the origin server) and hundreds of server clusters deployed closer to viewers [8, 31]. In general, all videos are stored in the origin server, while only popular videos are replicated and deployed across the clusters. When a viewer requests a video, it initiates a DNS resolution process to determine the optimal cluster. The DNS resolver responds with the IP address of the closest cluster. The cluster either directly serves the requested video from its cache or fetches it from the origin server if not available. The content is then delivered to the viewer using HTTP, allowing for adaptive streaming based on the viewer's network conditions [26, 40]. Through DNS resolution and widespread clusters, CDNs efficiently serve viewers by minimizing latency and improving delivery performance. However, the cost of employing CDNs services has constituted a significant portion of the overall operation cost. At the same time, with the increase in the user number and the development of video resolution, this cost will consistently increase in the future. As a result, reducing CDNs cost has become a strong incentive for content providers.

Compared to traditional CDN clusters, edge resources are much more cost-efficient as they are generally less performant (such as household routers, or small servers). E.g.,, users may receive free setup boxes and agree to lease them for third-party use when they are not actively using them; or the servers in datacenters, although limited to office hours, perfectly serve the evening rush hour for video streaming

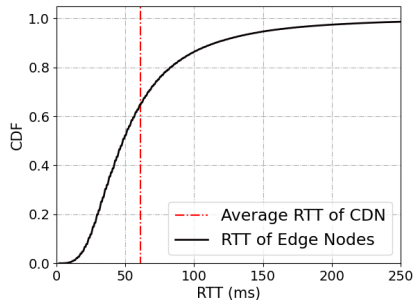

Figure 1: Compare node resources' RTT with CDNs.

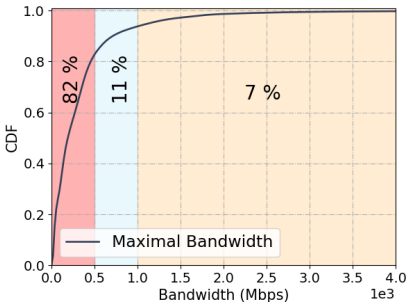

Figure 2: The distribution of bandwidth constraints for the node resources.

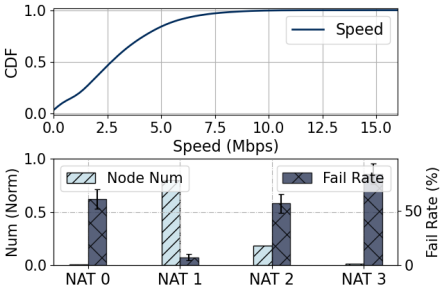

Figure 3: The service speed and accessibility for the node resources.

services. For ByteDance, these resources are collected by third-party companies (denoted as edge resource providers), and leased to ByteDance at a certain price. In contrast to CDN-based services, which mostly manage the overall video distribution process and provide software as a service (i.e., the distribution process is transparent to us), edge resource providers only provide physical servers that must be managed by ourselves. Also, the payment is made only for the incurred bandwidth costs.

2.1 Resource Overview

We provide three critical features of the node resources in our system. First, the edge resources in our system are **crowdsourcing**: i.e., 1) the node number in the PCDN system is incredibly large, and 2) these nodes are geographically distributed across a wide range of locations. As of early 2023, the PCDN system has nearly a million nodes, which are extensively located in over 200 cities across China (in contrast, CDN clusters are usually deployed at the province level). This property is highly promising because it inherently brings the resources closer to users, thereby providing a faster response. For better demonstration, we compare the RTT of edge resources and CDNs, and present the results in the form of CDF (cumulative distribution function) in Figure 1. We can see that over 63% edge nodes are equipped with competitive (or better) RTT with CDNs.

Second, there exists significant **heterogeneity** among their hardware. For demonstration, we measure the maximal bandwidth of all edge nodes (this value is provided by resource providers), and present the results in Figure 2. We can see that for the large edge nodes, their bandwidth constraint can reach up to 1000 Mbps, while for the small ones, the value comes to less than 500 Mbps. At the same time, we also observe that most of the edge nodes are small (i.e., 82% are less than 500 Mbps). Therefore, compared to CDNs with large clusters, the heterogeneity of edge nodes and the presence of numerous small nodes make resource management even more complex.

Third, the edge resources usually suffer from **poor performance**. For illustration, we investigate their service speeds and present the results in Figure 3 (the top sub-figure). we can see that most of the edge nodes exhibit low-speed capabilities. E.g.,, over 80% of the nodes have speeds less than 4.8 Mbps; while only less than 1% nodes have speeds exceeding 8 Mbps (as a comparison, the average speed for CDNs in our system is larger than 24 Mbps). Therefore, it is difficult to satisfy the speed requirement in most cases by solely relying on a single edge node.

In addition, the accessibility of edge resources is also not satisfactory. Different from CDN clusters equipped with public IP addresses, the edge nodes are probably behind NAT (Network Address Translation), and different NAT types directly affect the success rate of P2P (peer-to-peer) connection even using P2P STUN (Peer-to-Peer Session Traversal Utilities for NAT) techniques [32]. Figure 3 presents the distribution of edge resources for different NAT types and their connection failure rates (the bottom sub-figure). We can see that there exist a lot of non-public IP nodes (i.e., NAT0, NAT2, and NAT3), accounting for approximately 23%, and these nodes have significantly higher failure rates than NAT1 nodes (for NAT0 to NAT3, the fail rates are 61%, 7%, 57%, and 84%).

To leverage the edge resources' potential and overcome their disadvantages, ByteDance has developed a customized edge-based delivery system called PCDN.

2.2 PCDN in ByteDance

2.2.1 PCDN, CDN and Video Player

We start with describing the relationship between PCDN, CDN and video player in our system. As illustrated in Figure 4, videos are segmented into several chunks, and the downloaded chunks are stored in the data buffer managed by the video player. CDN and PCDN all act as the data module (i.e., ② in Figure 4). At any time, the video player assigns chunk-level tasks to CDN/PCDN (i.e., ① in Figure 4). The specific assignment, however, is determined by the player based on the current task status: e.g., CDN is usually utilized for downloading first chunk (denoted as preload in Figure 4), and serving as the backup in cases of urgent data requests (e.g., the PCDN system faces performance issues, or there is inefficient data in

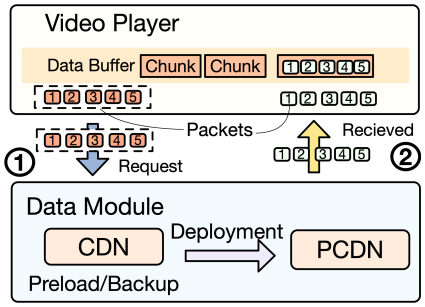

Figure 4: The relationship of CDN, PCDN and video player.

player's buffer). Meanwhile, CDN is also utilized to deploy video files to edge nodes (e.g., replace unpopular videos with popular ones), and we will not describe here as the details are beyond this paper's scope.

The above processes ensure that the traffic offloading does not affect user QoE (quality of experience). Moreover, as the service speed of PCDN increases, a larger portion of the traffic can be offloaded, thereby yielding substantial cost savings. In the following part, we will give more details of PCDN.

2.2.2 PCDN Server Side

The core idea of PCDN is to serve viewers with multiple nodes to compensate for the performance loss of the single node. PCDN is composed of both *server side* and *client side*. The server side has two functionalities: the tracker service, and the log service. The tracker service records the index between video content and edge nodes. The log service consistently collects some statistical information for each download task, including user information (e.g., filename) and node information (e.g., workload level and geographic location). We have also presented the workflow of PCDN in Figure 5.

At any time when the video player initiates a downloading task (❶ in Figure 5), the tracker service will be firstly invoked, and then determines L nodes to be returned back to the client (❷ in Figure 5). L is the maximal node number that the user can connect. In detail, the tracker selects nodes based on two criteria: first, it gives priority to returning nodes located in the same region as the user. Second, among nodes within the same region, it then prioritizes "high-quality" nodes, which are labeled by the resource providers. These nodes offer larger bandwidth, but also come at a higher cost. In Figure 6, we also present the download speed of different priced resources (i.e., high-price, middle-price, and low-price), and we can see that high-price resources indeed achieve better performance.

After receiving the response indicating which edge nodes are available, the client side will establish connections to all the nodes. It is notable that although the client will connect all the returned L nodes simultaneously, only the first K out of L nodes that respond will be utilized to download data. K is a carefully selected parameter. Determining an appropriate K is non-trivial: as too large K can lead to competition in downloads among nodes, while a too small K may not fully

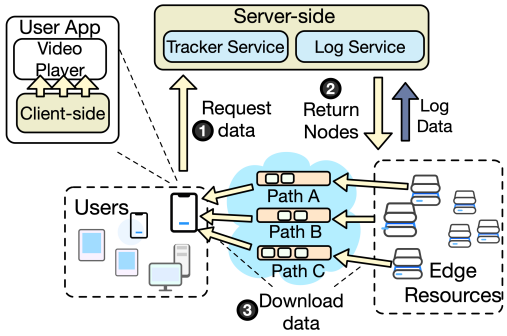

Figure 5: The system and workflows of ByteDance's PCDN.

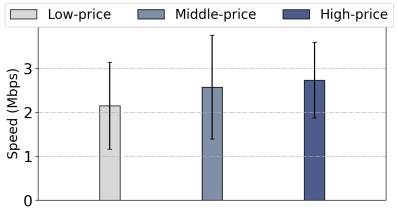

Figure 6: The download speed of different priced resources.

exploit the advantages of multiple nodes. We select K through multiple A/B tests. In case of failure in K nodes, replacement involves randomly selecting a node from the remaining $L-K$ nodes.

2.2.3 PCDN Client Side

The PCDN client side is responsible for the management of the data downloading process (❸ in Figure 5). It is integrated into the user's app and collaborates with CDNs to provide data for the video player (see Figure 4). The PCDN client side leverages a UDP-based multi-source parallel downloading (MPD) mechanism (i.e., simultaneously download from multiple edge nodes). The core idea behind this design is straightforward: on the one hand, multiple edge nodes can well compensate for the slow speed of any individual edge node; on the other hand, connecting to multiple nodes concurrently greatly improves the success rate of connections by leveraging the node redundancy.

Figure 7 presents the details of PCDN client side. The transport protocol follows a receiver-driven approach for data downloading, where the client side sends request packets to the edge nodes, and the nodes respond with the required data packets. This approach allows the client side to elastically determine how to request data from multiple nodes and ensures high flexibility. As illustrated in Figure 7, when the client side receives the data-downloading task, all the data-request packets will be first stored in a packet queue, and then allocated among multiple connections through a packet scheduler. In our real-world production, the scheduler employs a "best-effort" approach, i.e., iterating through all paths, and as long as a path has free congestion windows, the data requests

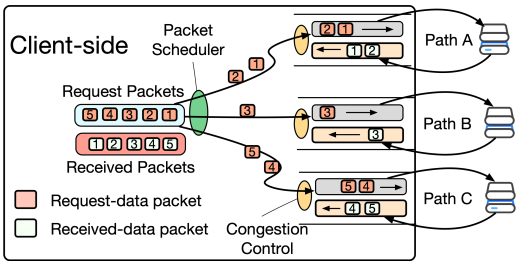

Figure 7: The details of PCDN client side. Specifically, it leverages a customized receiver-driven MPD mechanism.

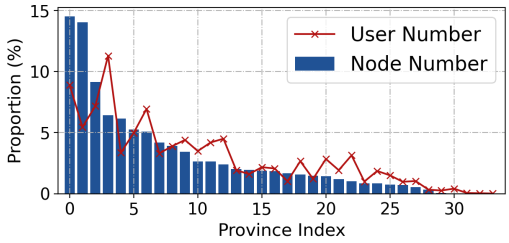

Figure 8: The current geographical distribution of resources and viewers.

will be immediately sent through that connection. Moreover, the transport protocol also follows a decoupled multi-path congestion control method [9, 10], wherein each connection independently maintains its congestion window based on its network information. Designing a good congestion control algorithm is a significant challenge in our unique MPD scenario [16], but it is beyond the scope of this paper and is only simply mentioned here.

It is notable that our system differs significantly from traditional P2P-based file-sharing systems (e.g., BitTorrent [43]): as prevalent P2P systems are fully distributed and do not guarantee service performance; in contrast, PCDN requires proactive management of node resources to ensure service quality while simultaneously meeting the demands of a growing user base.

For better illustration in the following text, we use **task** to denote the entire video downloading process, while use **connection** to denote the node-level downloading process. Therefore, each task can have multiple connections.

3 Improvement is Still Needed

3.1 Problem

With more users included in PCDN, the system also consistently augments more node resources. However, we find that to ensure the service quality remains uncompromised, the increment of resources should be much larger than the increment of users. E.g.,, from Nov. 2021 to Jan. 2022, the workload in PCDN has increased by about 15%, but the node number has increased by over 25% (we do not report detailed numbers due to commercial concerns). Therefore, this mismatch between the growth rates of resources and users has led to a gradual reduction in the marginal cost-effectiveness of PCDN.

To solve this problem, we have conducted a series of actions. E.g., when deploying new edge nodes, we consider the variations in user distribution across different geographical locations and ISPs. Taking geographical location as an example, Figure 8 presents the geographical distribution of resources and users across 34 provinces. We observe a consistent distribution between them. For instance, among the top three provinces, two appear in both the node count and user count rankings. Similarly, within the top ten, eight provinces overlap between the two rankings. However, through all these attempts, the problem of decreasing cost-effectiveness still exists.

After conducting a thorough analysis of the online data, we have identified that the primary cause for the decline of cost-effectiveness is the **unbalanced resource utilization problem**. Figure 9 illustrates the bandwidth utilization of all nodes. It can be seen that over 22% of nodes are already overloaded (i.e., their bandwidth utilization exceeds 100%), while a significant number of nodes are still underutilized (e.g., about 30% of nodes having a bandwidth utilization of less than 40%). Actually, this unbalanced resource utilization can introduce two critical problems. First, overloading can deteriorate service performance. Figure 10 shows an example (see the top sub-figure), in which we select a certain node and present how its service speed varies with workload. We can see significant performance degradation when the node is overloaded (i.e., the workload exceeds a certain threshold, as circled in Figure 10). Second, the under-utilized nodes actually result in the wastage of resources. Specifically, we find that the low-price resources are more likely to be under-utilized. For illustration, we present the bandwidth utilization of different priced resources in Figure 10 (see the bottom sub-figure). We can see that the utilization for low-price resources is 53%, while for middle-price and high-price, the utilization comes to 81% and 72% respectively. To this end, we can conclude that the overall PCDN system is still far from its satisfactory utilization, and the improvement is still needed.

3.2 Limitations

To alleviate the above problems, we need to first investigate the underlying limitations that contribute to the above problems. Specifically, we conduct large-scale measurement and highlight the following two main limitations in our first-version PCDN system.

▷ **Limitation 1: Server-side decisions are static and lack of global view.** PCDN server side selects nodes based on fixed rules: i.e., 1) prefer same-location and 2) high-price nodes. However, making decisions through these static settings has two main problems. First, it tends to return a small number of resources, which makes them more likely to be overloaded.

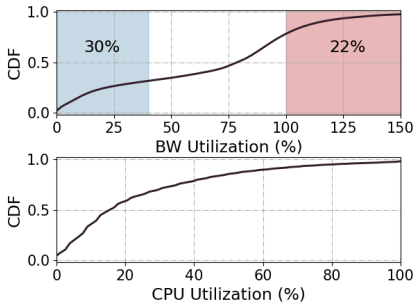

Figure 9: PCDN is resource-unbalanced.

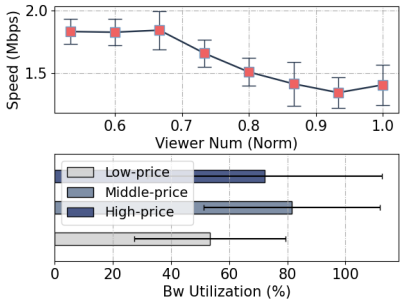

Figure 10: Systematic problems of PCDN.

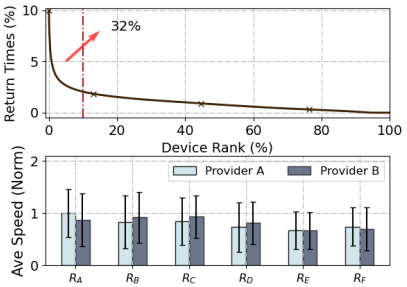

Figure 11: Server-side static decisions

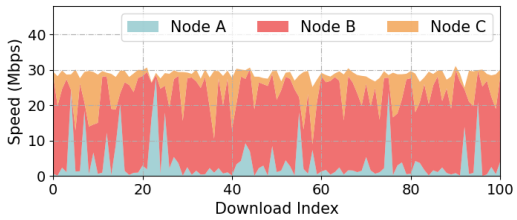

Figure 12: The PCDN client side is uncontrolled.

For illustration, we sort all nodes in descending order based on their return frequency, and present the results in Figure 11 (see the top sub-figure). We can see that the frequency is highly skewed: the top 10% nodes have been returned in 32% requests, while the bottom 40% of nodes only cover 10% requests. Second, fixed rules also restrict PCDN from capturing the characteristics of users/resources for global-view resource allocation. Actually, service speed can be influenced by a lot of factors, and merely using geographical location and price is inadequate. E.g.,, in addition to region and price, we further consider another feature: the resource provider, and present its influence in the bottom sub-figure of Figure 11 (denoted as Provider A and Provider B). We observe that even with the same geographical/high-price resources, the download speeds of these two providers are significantly different. Furthermore, their difference is inconsistent across regions. E.g.,, provider A is better than B in region A (denoted as R_A), but worse in region B (i.e., R_B). Actually, through better describing the performance when aligning different-featured users/resources, we can further optimize the system (take Figure 11 as an example, we can prefer provider A when serving users in R_A, while provider B for users in R_B.).

▷ **Limitation 2: Client-side downloading is uncontrolled and lacks coordination with the server side.** PCDN client side downloads data in a free-competition manner: i.e., 1) it initiates data downloading as long as a connection is established, and 2) sends data request whenever there is a free congestion window. However, in an MPD scenario, such an approach may introduce randomness in resource consumption. For demonstration, we repeatedly download the same video file from three identical nodes, and we then analyze the speed contribution for each of them (it is notable that to make the results more convincing, we select three large nodes that have bandwidth larger than 1000 Mbps). The results are presented in Figure 12. We can see that the speed contribution of each node varies significantly in each download. Take **Node A** as an example, the highest speed contribution is over 24 Mbps (i.e., in the 23rd download), but the lowest speed contribution is even less than 0.4 Mbps (i.e., in the 1st and 2nd download). This lack of control on the client side leads to two key issues. First, the client side will not respond to node status (e.g., actively reduce data requests when the node is overloaded). Second, it also prevents the client side from achieving on-demand resource allocation, therefore cannot coordinate with the centralized logic.

4 PCDN$^+$ Overview

Based on the above analysis, we are determined to evolve the previous PCDN to its next iteration, which is represented as PCDN$^+$. The basic idea of PCDN$^+$ is to enhance the previous static & uncontrolled system into a dynamic & controlled one. Specifically, we augment the PCDN server side with a centralized logic, which dynamically adapts the resource allocation strategy through global-view information; while we also deploy a controlled decentralized logic in the PCDN client side to achieve on-demand resource allocation. Figure 13 presents the system overview.

4.1 Centralized Logic

Centralized logic is designed with a **Profiling Module** and an **Allocation Stage**, which serves the following purposes: 1) outputs better resource allocation decisions; 2) provides the control knobs to the operators to balance quality and cost.

4.1.1 User & Edge Profiling

The profiling module is used to evaluate the download speed when assigning different featured viewers and edge nodes. Specifically, it will classify viewers and nodes into several groups based on some pre-selected features. The features are

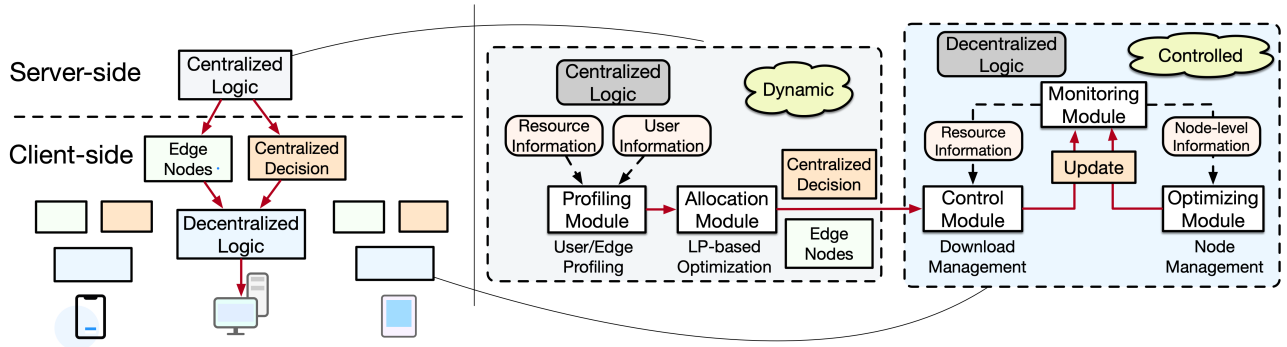

Figure 13: The system overview of PCDN$^+$, which consists of a dynamic centralized logic and a controlled decentralized logic.

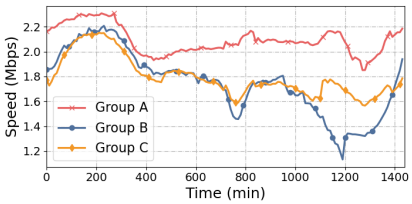

Figure 14: The temporal dynamics of different groups.

selected based on two criteria: 1) the users/resources within the same group should be "similar" (i.e., the variation of their speeds is small); 2) the feature dimension cannot be too large. The first criterion mainly considers from the algorithmic perspective: if there is a significant speed disparity within the same group, the resources will be mistakenly allocated. The second criterion mainly considers from the systematic perspective: on the one hand, a large feature dimension will result in a large group number, leading to a significant increase in computational complexity; on the other hand, to ensure fast connection establishment between nodes and viewers, we have to transmit all necessary information in a single MTU packet. Therefore, considering that some meta-data information already occupies a significant portion of the limited resources in the packet payload, the feature dimension should be carefully controlled.

After extensive feature engineering, we finally select the following features to classify viewers and nodes, which are *ISP*, *location*, *NAT type*, *connection type* (i.e., WI-FI/Cellular), *bitrate*, *resource provider*, and *resource price*. After dividing the viewers and resources through the above features, the profiling module will consistently collect data from users and nodes, and use the average speed over the past several timesteps as the prediction value. It is notable that although the selected features are fixed and we only use moving average as the prediction, we still need to periodically update the module's outputs. This is because the speed can show some temporal dynamics. Figure 14 shows an example: we can see that between [200, 400] min, the speed gap of Group B to Group C is fairly small; while when it comes to 1200 min, the gap has been significantly enlarged.

4.1.2 Dynamic Resource Allocation

Based on the outputs of the profiling module, we then step into the allocation module, which outputs the resource allocation decisions. Specifically, we model the resource management process as an optimization problem, which minimizes cost while satisfying performance requirements.

Assume that the users and nodes are divided into M and E groups through our profiling module. R_m denotes the number for m-th group of viewers. C_e and B_e are the price and total bandwidth for e-th node group, respectively. $s_{m,e}$ represents the average connection speed when m-th group of viewers are served by e-th group of nodes (i.e., the output of the profiling module). K is the node number used in MPD. Therefore, we need to determine the ratio $x_{m,e}$ of viewers in m served by node group e. To this end, the optimization problem can formulated as follows.

$$\min \sum_{e=1}^{E} C_e \sum_{m=1}^{M} R_m * x_{m,e} * s_{m,e} * K \tag{1}$$

$$\textbf{s.t.} \quad x_{m,e} \geq 0 \quad (m = 1, \cdots, M, e = 1, 2, 3, \cdots, E). \tag{2a}$$

$$\sum_{e=1}^{E} x_{m,e} = 1 \quad (m = 1, \cdots, M). \tag{2b}$$

$$\sum_{m=1}^{M} x_{m,e} * K * R_m * s_{m,e} \leq B_e \quad (e = 1, \cdots, E). \tag{2c}$$

$$\sum_{e=1}^{E} x_{m,e} * K * s_{m,e} \geq S_{m,\text{target}} \quad (m = 1, \cdots, M). \tag{2d}$$

The objective function is written as Eq.(1), which means that we aim to find an allocation policy for each group of viewers, through which the cost can be minimized; Eq.(2a-2b) ensure that all viewers should be served; Eq.(2c) indicates that the bandwidth usage should not exceed resource constraint. $S_{m,\text{target}}$ in Eq.(2d) is a hyper-parameter, which is the minimum speed for m-th group of users (it is mainly related to the *bitrate* feature). Therefore, Eq.(2d) indicates that the task speed (represented as the sum of K node speed) should satisfy the performance requirement. We accept that using the sum of connection speed as the task speed may be imprecise, but

it is sufficient from the perspective of global-view resource allocation. It is notable that through Eq.(2c) and Eq.(2d), we can control the trade-off between cost and performance. E.g.,, if we prefer cost reduction, we can increase the quantity of low-price resources and decrease the quantity of high-price resources.

To solve this problem, the viewer number R_m, the bandwidth B_e, and also the average speed $s_{m,e}$ should be prepared in advance. $s_{m,e}$ is obtained through the profiling module. R_m is computed based on the average viewer count of each group in previous time steps, which is similar to the calculation of $s_{m,e}$. Bandwidth B_e is collected through the periodic reporting from server-side log service (see Figure 5). The optimization problem itself is a linear programming problem, which has been well-studied and can be solved by many methods, such as interior-point [33] and simplex [30]. The computational complexity of these methods depends on the dimension of the decision variables (i.e., E*M) [7]. After solving the aforementioned optimization problem, the server side will sort all the nodes according to $x_{m,e}$, and return both $x_{m,e}$ and the top-ranked nodes to the client side. The centralized logic outputs dynamic resource allocation, and we then need the decentralized logic to coordinate with the centralized logic.

4.2 Decentralized Logic

The decentralized logic aims to facilitate a controlled data downloading process on the PCDN client side, therefore it should support the following two functionalities, i.e., 1) achieve on-demand resource utilization (i.e., $x_{m,e}$), and 2) satisfy the speed requirement (i.e., $S_{m,\text{target}}$ in Eq.(2d)). We design a *multi-source joint bandwidth allocation algorithm* to achieve these. The "multi-source joint" means PCDN$^+$ employs the information of all connected edge nodes; while "bandwidth allocation" means that PCDN$^+$ will make bandwidth allocation to satisfy centralized requirements. Details are as follows:

On-demand Allocation: In a typical MPD scenario, multiple connections share the last mile of the network. Therefore, the available bandwidth provided to the user is the sum of the download bandwidth of multiple connections. In order to control the download bandwidth of each connection, we introduce a maximum window mechanism to give an upper limit on the connection's in-flight packets. Specifically, each connection estimates the congestion window based on its own congestion control algorithm, but the window cannot exceed the assigned maximum window value. If the bandwidth share of a certain connection exceeds the target, we should lower its maximum window to reduce the number of packets sent; conversely, the maximum window can be increased to increase the number of packets sent. By setting a proper maximum window for each connection, we can adjust bandwidth contribution as needed. It is notable that this mechanism is decoupled from the underlying congestion control algorithm, as it only periodically controls the upper bound of the congestion window.

Next, we describe how to calculate the maximum window for connection k (denoted as $CWND_k^{\max}$). Suppose the user is featured with m. First, the algorithm monitors the aggregated bandwidth of multiple connections (denoted as $BW = \sum_{k=1}^{K} BW_k$) and the contribution of each connection (i.e., $\hat{x}_{m,e}^k = BW_k/BW$). After that, it calculates the achieved contribution of each resource group by aggregating the contribution of connections belonging to the same group (i.e., $\hat{x}_{m,e} = \sum_{k=1}^{K} \hat{x}_{m,e}^k$, if k in group e). Then for each resource group e, we will compare the achieved contribution $\hat{x}_{m,e}$ and the target contribution $x_{m,e}$, and compute a scale factor α_e shared by all nodes in group e. That is if $\hat{x}_{m,e} > x_{m,e}$ (this indicates that this group of resources is too much used by this user), the scale factor α_e will be set with a small value (usually $\alpha_e < 1$); while if $\hat{x}_{m,e} > x_{m,e}$, α_e will be set with a large value. After that, the shared bandwidth of node k in the next round will be updated as $BW_k = BW * \frac{\alpha_k}{\sum_{k=1}^{K} \alpha_k}$. Finally, the maximum window of connection k can be calculated using a simple window-based model, i.e., $CWND_k^{\max} = BW_k \times RTT_k$, where RTT_k is the average RTT.

Satisfy Speed Requirement: To achieve the second functionality, our design is to enable dynamic resource optimization. The core idea is: in our MPD scenarios, the bottleneck can occur either on the user side or on the resource side. As a result, to ensure that our allocation algorithm does not deteriorate the task speed, we need to prevent the resource side from becoming a bottleneck. For illustration, we present the idea in Figure 15 (the solid blue box indicates the requirement from users, and the dash yellow box indicates the available resource in edge nodes), in which we care about two cases:

- **Case #1** (the left sub-figure): the bottleneck is located on the user side. In this case, allocating the resource will not introduce any speed degradation: any group of node resources can solely cover the user-side requirement, therefore, no matter how the data requests of this user are allocated, the speed will not decrease.

- **Case #2** (the right sub-figure): the bottleneck is located on the resource side (i.g., *Resource A*). In this case, if too many data requests are allocated to *Resource A*, the task speed will suffer degradation. Meanwhile, if we can replace it with a more performant one (i.e., *Resource A'*), we then transfer case #2 to case #1, which enables us to allocate resources freely without compromising speed.

In practice, we distinguish the above cases by monitoring the task speed: i.e., if there is a significant decrease in speed when attempting to increase the share of certain resource group, it corresponds to Case#2; otherwise, it is Case#1.

We implement the above designs into the transport protocol. As illustrated in Figure 16, decentralized logic mainly consists of three parts, i.e., *Monitoring Module*, *Control Module*, and *Optimizing Module*. The monitoring module manages

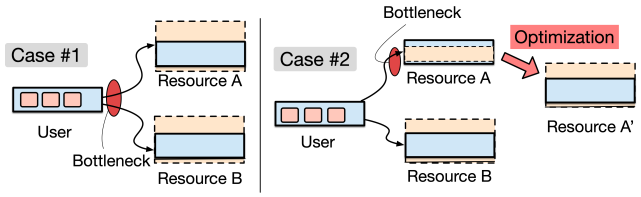

Figure 15: Resource optimization to satisfy the speed requirement.

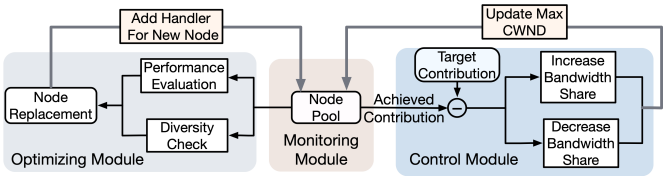

Figure 16: The decentralized logic overview of PCDN$^+$.

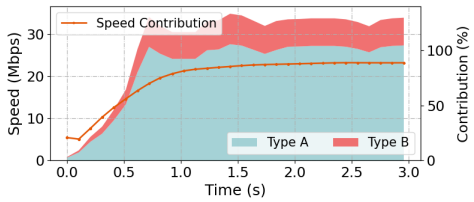

Figure 17: The case study of how decentralized logic works.

all connected nodes, and collects the network information of each connection. The control module refers to our allocation design, which uses the collected information to calculate $CWND_k^{\max}$. The optimizing module refers to our resource optimization design, which checks whether there is a bottleneck on the resource side. Notably, it performs two primary functionalities. First, it evaluates the performance of each node based on transport layer information. Secondly, it also verifies whether the resources are equipped with enough feature types (denoted as *Diversity Check* in Figure 16). Any newly added nodes must be registered into the monitoring module.

We demonstrate the entire download process of a file and showcase how the decentralized logic works. For clarity, we only select four nodes, and we randomly divide them into two groups, denoted as **Group A** and **Group B**. Specifically, we consider an extreme case to demonstrate the effectiveness: we hope that group A can contribute 90% of the task speed (i.e., $x_A = 0.9$ while $x_B = 0.1$). We present both session speeds (see the stacked plot) and **Group A**'s contribution (see the red line) in Figure 17. We can see that at the first 100 ms, the contribution is fairly small (about 25%); while as the downloading process goes on, the contribution starts to quickly increase, and finally stabilizes at the target contribution (i.e., 90%). Moreover, we also see that the contribution is stable even when the task speed fluctuates.

Another straightforward idea for on-demand allocation, which is using the packet scheduler in Figure 7. However, this method may potentially limit the full utilization of certain connections (i.e., the connection has free congestion windows, but no data is allocated to it). In contrast, by periodically setting the upper limit of the congestion window, and avoiding directly changing the congestion control algorithm or packet scheduler, our method is more protocol-friendly and ensures the full utilization of each connection.

5 Evaluation

In this section, we evaluate the performance of PCDN$^+$ through Bytedance's real production. We have rewritten the PCDN server side and client side to support PCDN$^+$. The server-side logic runs in minute-level, while client-side logic runs per task. Specifically, we first evaluate the decentralized logic; then we evaluate full PCDN$^+$[1]. It is notable that we do not compare the performance with CDNs or other systems, because our downloading process automatically switches to CDNs if PCDN cannot satisfy performance criteria (see §2.2).

5.1 Decentralized Logic Evaluation

We first demonstrate the effectiveness of decentralized logic by answering three questions: 1) Can the resources be controlled as expected? 2) Will the control logic have a negative impact on task speed? 3) Is the optimizing module necessary? The experiment is conducted through ByteDance's internally developed A/B test platform which has isolated nodes and simulated traffics. The experiment runs for a week, during which we collect data from millions of tasks.

Controllability: To better illustrate our results, we initially quantify the *control error* (denoted as d_c) by calculating the Euclidean distance between target resource allocation ratio (denoted as $x_{m,e}$) and the achieved ratio (denoted as $\hat{x}_{m,e}$), and the lower d_c indicates a higher controllability. Moreover, to make the results across different target ratios comparable, we normalize the results by using the maximum distance $d_{\max}$, i.e., $d_c = \frac{||x_{m,e} - \hat{x}_{m,e}||_2}{d_{\max}}$ (notably, different $x_{m,e}$ may have different $d_{\max}$, and it can be easily calculated by iterating through the vertices of the hyperplane defined by $\sum_{e=1}^{E} x_{m,e} = 1$). Figure 18 presents the results (it is notable that although we only show the results on a daily basis, we consistently change the $x_{m,e}$ in our one-week experiment). We can obtain the following observations. First, we find that PCDN is unable to achieve resource controllability: we can see that its control error ranges widely from 0 to 1, and more than half of the tasks experience a control error larger than 0.47. This result is expected, since PCDN downloads data in a free-competition manner and lacks resource control. In contrast, PCDN$^+$ well controls the node resources: hroughout the entire one-week

[1]We do not conduct separate tests for centralized logic, as it only works in conjunction with decentralized logic

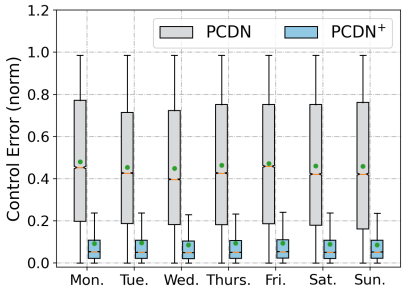

Figure 18: The controllability of decentralized logic.

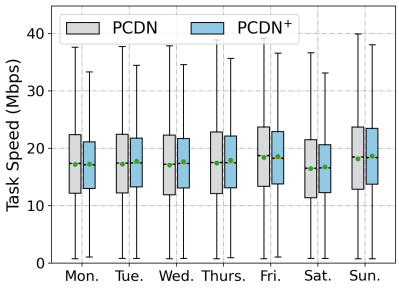

Figure 19: The performance impact of decentralized logic.

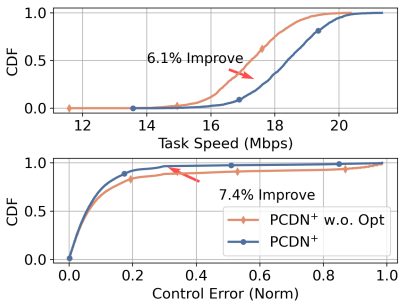

Figure 20: The necessity of optimizing module

experiment, PCDN$^+$ consistently exhibits very low control errors (50th percentile consistently below 0.04).

The Influence on Performance: We then investigate how PCDN$^+$'s decentralized logic influences task speed. The results are presented in Figure 19. We find that compared with PCDN, PCDN$^+$ will not have a notable negative impact. E.g.,, we can see that during one-week experiment, PCDN$^+$ achieves an average speed improvement ranging from 1.2% to 2.7% compared to PCDN (the 50th percentile speed has a similar observation). These results align with our expectations: the optimizing module in decentralized logic enables us to shift the bottleneck from the possible node-side to the user-side, allowing us to safely allocate resources. At the same time, we also notice that PCDN performs better than PCDN$^+$'s in maximum speed (see the top error bar shown in each boxplot). This is because PCDN downloads data in the free-competition way. This is also acceptable as the average & 50-percentile speed is more meaningful at systematic level.

The Necessity of the Optimizing Module: We further conduct an ablation study to demonstrate the effectiveness of the optimizing module in PCDN$^+$'s decentralized logic. Specifically, we compare the performance of PCDN$^+$ with/without optimizing module (denoted as **PCDN$^+$ w.o. Opt**). We first focus on the service performance, and we compare the speed distribution in Figure 20 (see the top sub-figure). We can see that without optimizing module, the speed will suffer significant degradation (the average speed degrades about 6.1%). This is because the bottleneck may be located on the server side in certain cases, in which the servers cannot provide required performance (see Figure 15). Moreover, we also focus on controllability, and present the results in the bottom sub-figure of Figure 20. We find that compared with PCDN$^+$, the control error of **PCDN$^+$ w.o. Opt** is also degraded (the average control error increases about 7.4%). This is because the optimizing module also provides diversity checks to guarantee that there are enough node groups to use.

5.2 PCDN$^+$ Evaluation

The A/B tests of PCDN$^+$ are based on all PCDN nodes, and cover 20% of Douyin users. Our specific goal is to optimize the utilization of low-price edge resources with PCDN$^+$ for further cost reduction. We evaluate PCDN$^+$ through the perspective of resource utilization, cost savings, and also service performance.

Resource Imbalance: We first investigate the bandwidth utilization. In Figure 21, we compare the distribution of bandwidth utilization in the form of CDF. We can obtain the following observation. First, the overloading problem has been significantly improved. E.g.,, we can see that for PCDN there are over 20% nodes are overloaded, while for PCDN$^+$, almost all nodes are below the bandwidth constraint. Second, the issue of under-utilization is also well alleviated (as pointed out in the black dashed circle). Taking 40% as the threshold, we see that only 20% nodes in PCDN$^+$ are under-utilized, while the value is 30% for PCDN.

Cost Savings: We then analyze the utilized bandwidth of the low-price nodes. The results are presented in Figure 22 (we also present the relative improvement). We can find that compared with PCDN, the workload for the low-priced nodes has been significantly improved. E.g.,, we see that throughout the one-week experiment, the workload improvement ranges from 31.0% to 95.8%. It's notable that we cannot disclose the actual cost savings due to the commercial confidentiality of the relative pricing of different resources. However, to better illustrate the cost-saving benefits of PCDN+, we present the changing proportion of low-cost resources within the entire system over the past three months. The results are depicted in Figure 22 (refer to the bottom sub-figure). We can see that low-price resources have consistently increased and have already accounted for over 50%. Therefore, even with a conservative assumption of a 20% discount on these resources, the corresponding estimated cost savings compared to PCDN range from 8% to 10%.

Service Performance: Considering that the cost savings are obtained through more utilization of low-price resources, which are usually equipped with unsatisfactory speed (see Figure 6), we further evaluate the service performance of PCDN$^+$. Specifically, we consider the task speed, and the results are presented in Figure 23. We can see that PCDN$^+$

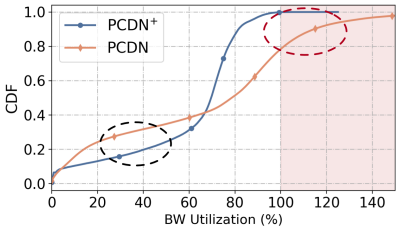

Figure 21: PCDN$^+$ mitigates resource imbalance issues.

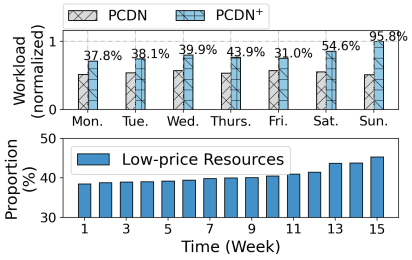

Figure 22: PCDN$^+$ reduces bandwidth costs.

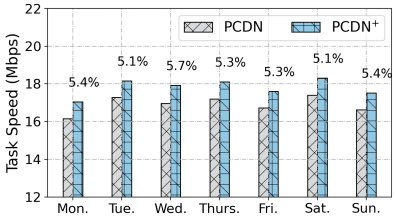

Figure 23: PCDN$^+$ improves the task speed.

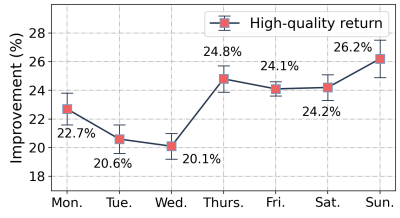

Figure 24: The improvement of server-side high-quality return.

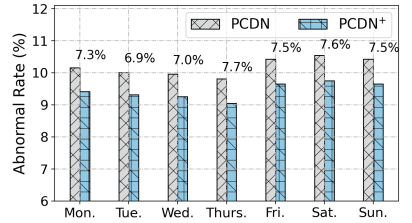

Figure 25: The comparison of abnormal rate.

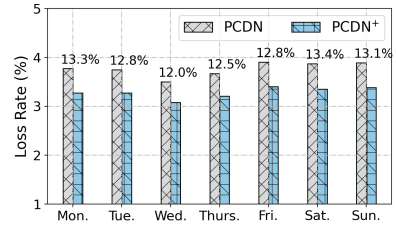

Figure 26: The comparison of loss rate.

also achieves higher speed and outperforms PCDN. E.g.,, we see that on Monday, PCDN$^+$'s task speed is 17.04 Mbps, about 5.4% higher than PCDN.

5.3 Benefit Deep Dive

We first demonstrate the effectiveness of PCDN$^+$'s centralized logic through the metric called *high-quality return*. This metric is defined from the perspective of the server side, and mainly depends on three connection-level criteria: speed, packet loss, and RTT (recall that connection speed is different from task speed). If and only if all three metrics satisfy some pre-defined criteria, can this node be regarded as a high-quality return node. Since the thresholds for different video types are different and may vary at each day, here we only report the improvement. The results are presented in Figure 24. We can see that PCDN$^+$ significantly improves the performance, and the improvement of the week is consistently higher than 20% (the highest improvement is 26.2%). This is expected as the centralized logic has the global view of the resources, and better characterizes the performance when aligning different users and nodes.

We then focus on abnormal connections, which refer to the scenario when a user fails to download data from the node. This scenario commonly arises from inherent deficiencies within the node itself, such as excessive overloading, resulting in connection disruptions. Therefore, it can also reflect the effectiveness of the centralized logic. The abnormal state is recorded at the connection level and is reported once the task is completed. To this end, we calculate the abnormal rate (i.e., the proportion of abnormal connections relative to the total connections), and present the results in Figure 25.

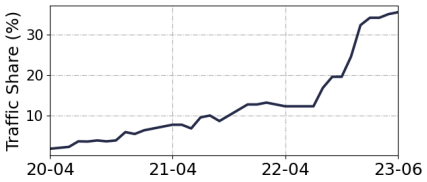

Figure 27: The traffic share of PCDN$^+$.

We can see that the abnormal rates are significantly reduced: throughout the week, the reduction ranges from 6.9% to 7.6%. This is mainly because PCDN$^+$ facilitates a dynamic & controlled framework which enables it to better manage resources, especially avoid returning low-quality nodes. Finally, we investigate the loss rate. The results are presented in Figure 26. We can see that the PCDN$^+$ also achieves better performance: the loss rate reduction ranges from 12.0% to 13.3%.

After extensive A/B test, PCDN has been evolved to PCDN$^+$ and fully deployed since 2023, In Figure 27, we also present the workload that has been shared by our system since April 2020. It can be observed that until 2023, our system accounted for approximately 20% of the overall traffic, but with a relatively slow growth rate. With the deployment of PCDN$^+$ in 2023, the traffic of our system has rapidly increased and as of June 2023, it has already taken over 35% of the total traffic. We also evaluate PCDN$^+$ by comparing with CDN for some QoE metrics: the stall ratio and stall frequency have decreased 0.86% and 3.45%, respectively.

6 Related Work

Video Delivery Optimization: Some work optimizes from the perspective of content providers and focuses on the request

scheduling. E.g., [3,20,25,46] propose to use some prediction models to predict the quality of experience (QoS), and select the best CDN providers. [21] utilizes some learning-based methods to make end-to-end server selection. This kind of work usually focuses on the last mile, and treats CDN as the black-box. At the same time, there is also work focusing on intermediate link optimization, like network relay and routing problems. E.g., VDN [28] and Livenet [23] present some centralized control for the live streaming routing optimization; VIA [19] cares about the VoIP calls and presents some dynamic relay selection algorithms. Some other work [38,47] also optimize for cloud service providers. Our work shares some similarities with them at a high level, such as the centralized control plan and global-view decision-making. However, the centralized decisions in our case require decentralized logic to assist in their execution. Moreover, compared to CDN, our utilization of crowdsourcing resources and the adoption of MPD also introduce unique challenges and considerations.

P2P System: PCDN is closely related to the P2P technique. Traditional P2P network (such as BitTorrent [6,43], PeerTube [14], and IPFS [42]) is fully decentralized, and users can join or leave whenever they want. However, this full decentralization also prevents it from directly utilizing in commercial video delivery, such as participant availability [41], and data security [36]. To tackle this problem, the combination of CDNs and P2P networks (also denoted as hybrid CDN-P2P) has emerged as a fascinating framework. There have been a lot of work addressing various aspects, including the large-scale measurement studies [11,27], the impact of heterogeneous bandwidth constraints [45], resource allocation optimization, and also caching strategies [34]. Compared with existing work, since the edge resources in our system are collected and provided by specialized companies, they are more stable (i.e., we do not need to consider the impact of abrupt joining or leaving), and there is no need to worry about security risks.

Resource Allocation: Our work also relates to resource allocation field. Traditionally, the resource allocation aims to maximize systematic utilization, such as achieving efficient operation cost [5,22], and specifying different service priorities [12,13]. E.g.,, Some of them optimize intra-WAN scheduling and aim to maximize bandwidth utilization or prevent bottleneck congestion [24]. Some work aims to quantify the value of certain flows and reduce the cost of WAN egress based on Shapley Values [39], Big-M method [38] or relaxed convex optimization [37]. They are similar to PCDN$^+$, as they all model the scheduling problem as an optimization process. However, PCDN$^+$ still needs a well-designed decentralized logic to coordinate with the centralized logic.

7 Lessons Learned and Discussion

From our experience of developing and optimizing PCDN system, we have obtained the following lessons. First, we find that using cheaper edge devices to replace CDN for content delivery is promising: on the one hand, although PCDN was initially designed only as a cost-saving supplement of CDNs, after all the work we have put in the system, we believe it has the full potential to actually replace CDN as a more cost-effective solution for future VoD streaming services; on the other hand, connecting to and downloading from multiple nodes enables the optimization at video chunk level, providing a much finer granularity than CDNs which typically schedule at task level. At the same time, we also believe that the optimal solution still requires optimization in several aspects as follows:

Better Prediction Model: There are several open questions we wish to highlight to further improve the PCDN system. First, the feature selection during the profiling stage of centralized logic is static and heuristic, and the predicted speed/viewer number are all based on the statistics of previous timesteps. However, recent advances in data-driven prediction models are promising [2,4] and worth future investigation.

Transport Protocol: PCDN leverages a receiver-driven multi-source transmission protocol, which presents an intriguing area for further investigation. Our observation reveals that congestion can arise due to competition when multiple connections share the last mile. Hence, it becomes imperative to explore more efficient congestion control algorithms tailored for MPD scenarios. Moreover, PCDN nodes are susceptible to disconnections and quality instability, often resulting in head-of-line blocking. Currently, we are exploring more efficient data scheduling and recovery algorithms.

Cooperation with CDNs: Currently, PCDN and CDNs are regarded as separate systems, with CDNs functioning as a backup for PCDN. However, this approach lacks support for more precise resource control and management, such as simultaneously downloading from both PCDN and CDNs. In our future work, we aim to consolidate the management of CDNs and PCDN nodes to enable the development of a more robust, efficient, and heterogeneous content delivery network.

8 Conclusion

In this paper, we have reported PCDN, ByteDance's alternative delivery system for CDNs. PCDN is designed with a server side and a client side, and through a customized multi-source parallel downloading mechanism. Moreover, to further address its resource inefficiency problem, we have also presented PCDN$^+$. In particular, through enhancing the PCDN server side with a centralized logic, and augmenting the PCDN client side with a decentralized logic, PCDN$^+$ has achieved both dynamic adaptability and controllability. We have also demonstrated the effectiveness of PCDN$^+$ through extensive A/B tests and discussed some open questions in our system. PCDN$^+$ now has served over 35% VoD traffic for ByteDance.

References

[1] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *2012 Proceedings IEEE Infocom*, pages 1620–1628. IEEE, 2012.

[2] Abdullah Alomar, Pouya Hamadanian, Arash Nasr-Esfahany, Anish Agarwal, Mohammad Alizadeh, and Devavrat Shah. Causalsim: A causal framework for unbiased trace-driven simulation. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1115–1147, 2023.

[3] Timm Böttger, Felix Cuadrado, Gareth Tyson, Ignacio Castro, and Steve Uhlig. Open connect everywhere: A glimpse at the internet ecosystem through the lens of the netflix cdn. *ACM SIGCOMM Computer Communication Review*, 48(1):28–34, 2018.

[4] Soumi Chattopadhyay, Richik Chanda, Suraj Kumar, and Chandranath Adak. Offdq: an offline deep learning framework for qos prediction. In *Proceedings of the ACM Web Conference 2022*, pages 1987–1996, 2022.

[5] Bo Chen, Zhisheng Yan, and Klara Nahrstedt. Context-aware optimization for bandwidth-efficient image analytics offloading. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2023.

[6] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72. Berkeley, CA, USA, 2003.

[7] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

[8] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[9] Bo Han, Feng Qian, Shuai Hao, and Lusheng Ji. An anatomy of mobile web performance over multipath tcp. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. CoNEXT '16, page 129–143, New York, NY, USA, 2016. Association for Computing Machinery.

[11] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W Ross. A measurement study of a large-scale p2p iptv system. *IEEE transactions on multimedia*, 9(8):1672–1687, 2007.

[12] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pages 15–26, 2013.

[13] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, et al. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 74–87, 2018.

[14] https://en.wikipedia.org/wiki/PeerTube.

[15] https://ir.kuaishou.com/corporate-filings/annual-interim reports/. Annual report of kuaishou. Technical report, 2022.

[16] https://mmsysgc23.github.io/. Multi-source parallel downloading challenge, 2023.

[17] https://www.douyin.com/. Douyin website.

[18] MANSOOR IQBAL. Tiktok revenue and usage statistics. In *https://www.businessofapps.com/data/tik-tok-statistics/*, 2023.

[19] Junchen Jiang, Rajdeep Das, Ganesh Ananthanarayanan, Philip A Chou, Venkata Padmanabhan, Vyas Sekar, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, Renat Vafin, et al. Via: Improving internet telephony call quality using predictive relay selection. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 286–299, 2016.

[20] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. Cfa: A practical prediction system for video qoe optimization. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 137–150, 2016.

[21] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation. In *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, pages 393–406, 2017.

[22] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. Decentralized cloud wide-area network traffic engineering with blastshield. In *19th*

USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pages 325–338, 2022.

[23] Jinyang Li, Zhenyu Li, Ri Lu, Kai Xiao, Songlin Li, Jufeng Chen, Jingyu Yang, Chunli Zong, Aiyun Chen, Qinghua Wu, et al. Livenet: a low-latency video transport network for large-scale live streaming. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 812–825, 2022.

[24] Wenxin Li, Xiaobo Zhou, Keqiu Li, Heng Qi, and Deke Guo. Trafficshaper: Shaping inter-datacenter traffic to reduce the transmission cost. *IEEE/ACM Transactions on Networking*, 26(3):1193–1206, 2018.

[25] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 359–370, 2012.

[26] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 197–210, 2017.

[27] Michel Meulpolder, Lucia D'Acunto, Mihai Capota, Maciej Wojciechowski, Johan A Pouwelse, Dick HJ Epema, and Henk J Sips. Public and private bittorrent communities: a measurement study. In *IPTPS*, volume 4, pages 1–5, 2010.

[28] Matthew K Mukerjee, David Naylor, Junchen Jiang, Dongsu Han, Srinivasan Seshan, and Hui Zhang. Practical, real-time centralized control for cdn-based live video delivery. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM 15)*, pages 311–324, 2015.

[29] Usama Naseer and Theophilus A Benson. Configanator: A data-driven approach to improving cdn performance. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1135–1158, 2022.

[30] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[31] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[32] Marc Petit-Huguenin, Gonzalo Salgueiro, Jonathan Rosenberg, Dan Wing, Rohan Mahy, and Philip Matthews. Session Traversal Utilities for NAT (STUN). RFC 8489, February 2020.

[33] Florian A Potra and Stephen J Wright. Interior-point methods. *Journal of computational and applied mathematics*, 124(1-2):281–302, 2000.

[34] Osama Saleh and Mohamed Hefeeda. Modeling and caching of peer-to-peer traffic. In *Proceedings of the 2006 IEEE international conference on network protocols*, pages 249–258. IEEE, 2006.

[35] SANDVINE. The global internet phenomena report. In *https://www.sandvine.com/phenomena*, 2023.

[36] Benjamin Schleinzer and Nobukazu Yoshioka. A security pattern for data integrity in p2p systems. In *Proceedings of the 17th Conference on Pattern Languages of Programs*, pages 1–5, 2010.

[37] Li Shi, Yang Liu, Junwei Zhang, and Thomas Robertazzi. Coflow scheduling in data centers: routing and bandwidth allocation. *IEEE Transactions on Parallel and Distributed Systems*, 32(11):2661–2675, 2021.

[38] Rachee Singh, Sharad Agarwal, Matt Calder, and Paramvir Bahl. Cost-effective cloud edge traffic engineering with cascara. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 201–216, 2021.

[39] Rade Stanojevic, Nikolaos Laoutaris, and Pablo Rodriguez. On economic heavy hitters: Shapley value analysis of 95th-percentile pricing. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 75–80, 2010.

[40] Thomas Stockhammer. Dynamic adaptive streaming over http– standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, 2011.

[41] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202, 2006.

[42] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. Design and evaluation of ipfs: a storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 739–752, 2022.

[43] Raymond Lei Xia and Jogesh K Muppala. A survey of bittorrent performance. *IEEE Communications surveys & tutorials*, 12(2):140–158, 2010.

[44] Juncheng Yang, Anirudh Sabnis, Daniel S Berger, KV Rashmi, and Ramesh K Sitaraman. C2dn: How to harness erasure codes at the edge for efficient content

delivery. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1159–1177, 2022.

[45] Zhongmei Yao, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. Modeling heterogeneous user churn and local resilience of unstructured p2p networks. In *Proceedings of the 2006 ieee international conference on network protocols*, pages 32–41. IEEE, 2006.

[46] Rui-Xiao Zhang, Ming Ma, Tianchi Huang, Hanyu Li, Jiangchuan Liu, and Lifeng Sun. Leveraging qoe heterogenity for large-scale livecaset scheduling. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 3678–3686, 2020.

[47] Rui-Xiao Zhang, Changpeng Yang, Xiaochan Wang, Tianchi Huang, Chenglei Wu, Jiangchuan Liu, and Lifeng Sun. Aggcast: Practical cost-effective scheduling for large-scale cloud-edge crowdsourced live streaming. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 3026–3034, 2022.