USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

HyperEdge: An Edge CDN Infrastructure for Cost Efficient Video Streaming

Dehui Wei, *National University of Singapore;* Jiao Zhang, *Beijing University of Posts and Telecommunications, and Purple Mountain Laboratories;* Haozhe Li, Rui Han, Zhichen Xue, Yajie Peng, Xiaofei Pang, and Yan Ma, *ByteDance;* Jialin Li, *National University of Singapore*

https://www.usenix.org/conference/nsdi26/presentation/wei

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

HyperEdge: An Edge CDN Infrastructure for Cost Efficient Video Streaming

Dehui Wei[‡,§] Jiao Zhang[‡,¶,*] Haozhe Li[†] Rui Han[†] Zhichen Xue[†]
Yajie Peng[†] Xiaofei Pang[†] Yan Ma[†] Jialin Li[§]

[§]*School of Computing, National University of Singapore* [‡]*Beijing University of Posts and Telecommunications*
[¶]*Purple Mountain Laboratories* [†]*ByteDance*

Abstract

As ByteDance's business expands, the cost of video streaming using content delivery networks (CDN) has become prohibitively high. We have discovered a sea of under-utilized edge devices with the potential to reduce content distribution cost. The unreliable performance of an edge network, however, presents deep challenges to video streaming services. In this work, we introduce HyperEdge, an edge-assisted content delivery system for video streaming. HyperEdge seamlessly integrates the robustness of a conventional CDN with the cost-efficiency of an edge network. It offers dependable streaming quality to users while minimizing traffic expenses. HyperEdge employs a centralized tracker cluster to optimize content distribution to a pool of edge devices, based on real-time monitoring. To ensure satisfactory video playback quality, we develop a novel multi-path protocol for client-edge video transmission. Having been in stable operation for six years, HyperEdge manages over a hundred thousand edge devices, serving about a hundred million users daily, and saving hundreds of millions of dollars in content delivery cost annually.

1 Introduction

At ByteDance, we rely heavily on content delivery networks (CDNs) [42] to serve billions of global users across our wide range of products and services. As business continues to grow, we are forced to increase the capacity of our CDN provisioning from third-party providers to meet the surging demand. This continuous expansion comes with a steep cost increase associated with content delivery. Relying on central CDN infrastructure also leads to poor agility and flexibility [44, 65]. Even though our user traffic exhibits high temporal variance, we have to over provision CDN bandwidth for peak traffic. These compounding factors have led to an unsustainable increase in ByteDance's expenses on CDN.

How to reduce content delivery cost? Our investigation discovered a sea of *underutilized resources* scattered across global devices, including set-top boxes, smart home gadgets, dormant servers, and more. These *edge* devices have significantly lower per-unit hardware, bandwidth, electricity, and management cost than their data center counterparts [31, 66]. More importantly, harnessing edge devices enables better *resource elasticity* [68]. The network can scale up or down dynamically based on demand, and only pays for the resources required to serve the traffic.

Leveraging edge networks for content delivery, however, comes at a performance trade-off [36]. Edge devices are highly heterogeneous: They exhibit large variance in their network bandwidth, storage capacity, and computing capabilities; they are dispersed across large geographic locations, regional networks, and ISPs, without any central management; most of these devices are behind a private network, requiring lengthy Network Address Translation (NAT) penetration [15, 16] during connection setup. Such factors present formidable challenges to provide the strict real-time content delivery requirements of our services [17].

In this paper, we present HyperEdge, ByteDance's content delivery system for video streaming. HyperEdge employs a pool of underutilized low-cost devices outside our managed data centers and networks. The system delivers video streaming quality *on par* with dedicated CDN infrastructure, despite all the shortcomings of an edge network. HyperEdge carefully incorporates *centralized management* to address the performance challenges of edge devices. We divide the content distribution responsibility between a fast but simple data transmission plane, and a centrally managed decision-making engine. This centralized control plane consists of a cluster of *trackers* running in our data centers. Centralization enables the trackers to have a global view of edge device status, content location, network traffic, user demand, and service requirements. Trackers can therefore make fine-grained and optimized resource allocation, content distribution, and client-to-device mapping decisions.

The vast amount of edge devices power the HyperEdge data plane. As any single device may not have the computational power or network bandwidth to deliver the required

*Jiao Zhang is the corresponding author.

video streaming speed, we design a novel multi-site parallel downloading scheme. Specifically, clients establish connections to multiple edge devices caching the requested video and schedule concurrent video transmission from all connected devices. Our transmission scheme aggregates computation and network capacity from a pool of devices while meeting playback deadlines, ensuring video downloading performance that complies with our business requirements.

Even with careful designs, edge networks alone are insufficient to ensure our content delivery Service Level Objectives (SLOs): Long latency of NAT punching lengthens the video loading time; failures of edge devices and network issues are common, leading to unreliable transmission performance. To address such issues, we deploy a hybrid architecture that combines HyperEdge with a traditional CDN. The CDN serves three purposes: It provides a video content "backend storage" for HyperEdge; its more predictable performance is leveraged to reduce video loading time; it serves as a fallback option when the quality of service of HyperEdge drops below the SLO. However, we do not provision the CDN to serve peak user traffic. Instead, we rely on HyperEdge for cost-effective and elastic capacity provisioning. By carefully scheduling CDN traffic, this hybrid approach only adds marginal cost to our content delivery infrastructure.

HyperEdge has served as ByteDance's main content distribution engine for over six years. It has grown to 100,000 edge devices distributed across our deployed regions mainly covers Asia-Pacific and South America regions currently. Serving close to a hundred million daily users, HyperEdge processes more than ten billion daily playbacks and handles tens of terabytes of video traffic per day. Our strategy of embracing external edge devices has led to an annual cost saving of hundreds of millions of dollars. Critically, our large-scale A/B testing results demonstrate that the cost-efficiency of our HyperEdge design does not lead to any compromises in user experience. We provide more detailed experiences of deploying HyperEdge in Appendix A.

2 Background and Motivation

Videos have emerged as the dominant content in today's internet landscape. Our video hosting services, in particular, have demonstrated an expedited growth pattern. The growth in video popularity has resulted in rapid increase in video streaming network traffic. This massive traffic surge has brought challenges to the content delivery network infrastructure to continue delivering high-quality video services.

2.1 Content Distribution Networks

Traditional Content Delivery Networks (CDNs) are built as highly centralized and structured server-to-client systems. CDNs deploy dedicated server clusters across vast geographical locations. They scale content delivery capacity and enable users to fetch data from servers in proximity to reduce access latency [4]. A CDN integrates its own DNS servers to redirect user requests to edge servers [41]. On a cache miss, an edge retrieves the file from a server higher following a hierarchical model. To maintain fast and reliable video transmission, CDN clusters deploy powerful rack servers and use dedicated commercial networks from large ISPs [63]. Content providers rely heavily on CDNs for reliable video services to ensure low-latency and high-performance video streaming. Traditional CDN operators handle higher demand by scaling out their server infrastructure to expand the cache hierarchy. This expansion comes with substantial scaling costs [2, 25]. Our short video business (Douyin) has reported yearly traffic expenses exceeding billions of RMB.

2.2 Opportunity: Edge Devices For Cost Reduction

In the markets in which ByteDance operates, there exists an abundance of under-utilized resources nestled within household devices such as set-top boxes, smart home appliances, and occasionally idle servers. We refer to them as *hyperedge devices*, or **Hdevices**, to differentiate them from the narrower class of edge servers used by CDNs. Hdevices, often geographically closer to users, represent an untapped potential for content storage and distribution. They also offer cost-saving opportunities compared to CDN infrastructure. This cost-saving comes from two main factors: *1) Inexpensive resources*: Compared to data center rack servers, edge devices have lower hardware and utility cost. Moreover, by sharing the hardware with the user, we can tap into the under-utilized portion of their resources at a fraction of the overall cost. *2) Avoiding expensive dedicated networks*: Traditional CDNs spend a major fraction of their infrastructure budget on dedicated commercial networks [7] . Hdevices, however, are connected to cheaper public networks. In our operating regions, the cost difference between the two network types is substantial [*]. By exploiting the under-utilized network bandwidth, particularly for uploading, the cost of using Hdevices is further reduced. In §5, we provide a detailed cost analysis of leveraging edge devices for content distribution.

Performance limitations. While using edge devices has the potential to reduce content delivery cost, they present performance challenges to meet our video business requirements. Hdevices in our deployment typically offer limited storage capacity and computational power. Figure 1 shows the CDF of network bandwidth (top) and available storage capacity (bottom) based on 250,000 Hdevice samples. We observe that 90% of the devices have an upload bandwidth less than 54 Mbps, and more than 90% has less than 67 GB of storage.

[*]The degree of cost differences is region-specific. We are aware of other geographical regions where the difference is significantly smaller.

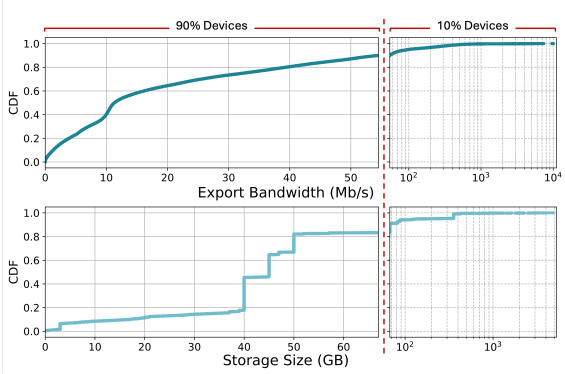


Figure 1: Resource distribution of Hdevices, including network bandwidth (top) and storage capacity (bottom). Hdevices are typically weaker than dedicated CDN servers and exhibit substantial resource heterogeneity.

Notably, 23% of the devices have bandwidth below 5 Mbps and 8.5% with less than 10 GB of storage, indicating a considerable proportion of highly constrained nodes. In comparison, CDN servers typically offer 1–10 Gbps uplink and several terabytes of storage; they are also configured with consistent and reliable hardware [39]. Moreover, Hdevices has a churn rate of 5%–10%, further complicating resource availability.

Furthermore, Hdevices exhibit high resource heterogeneity. In our dataset, device bandwidth ranges from below 1 Mbps to over 10 Gbps, and storage capacity spans from a few gigabytes to 4.8 TB. These wide disparities complicate system scheduling, making the design of a one-size-fits-all delivery strategy challenging. Other co-located services on the devices also impose performance interference. Collectively, these factors lead to highly unpredictable device performance.

Hole punching overhead. Most Hdevices operate within Local Area Networks (LAN) [71], behind Network Address Translation (NAT) barriers. Connecting to such devices introduces long delays in the video playback startup time [32]. Moreover, the complexities associated with NAT punching increase the likelihood of connection failures, further complicating user experiences.

2.3 The HyperEdge Approach

How should we organize the vast and heterogeneous pool of Hdevices to deliver high-quality video services that meet the strict SLO requirements? A natural design is to replicate the hierarchical architecture used in traditional CDNs. However, edge devices experience more frequent cache misses due to limited available storage space and dynamic content popularity patterns. They thus require more frequent data retrieval from higher-level caches or a CDN, generating substantial bandwidth cost. Furthermore, their longer RTTs impose higher latency penalties in each hierarchy traversal. Therefore, these resource-constrained devices do not match a traditional hierarchical approach.

Instead, HyperEdge adopts a *flat architecture* among Hdevices. All devices are egalitarian with no specific hierarchy. However, individual devices lack a global view of both video request demand and hardware resource distributions. Making device-local caching and content delivery decisions would not respond promptly to rapid changes in content popularity [35, 45], which can occur within minutes in real-world workloads [8, 68]. A decentralized design is also prone to load problems, particularly given the uneven resource distribution of the heterogeneous Hdevices. To address these challenges, HyperEdge heavily incorporates centralization in its core design. Specifically, a centralized tracker system collects global traffic demand information and Hdevice metrics. The system then leverages its vantage point to make globally optimal video distribution decisions; it also mediates all client video requests, ensuring sufficient transmission resources to satisfy SLOs while balancing Hdevice resource utilization.

Comparison to peer-assisted CDNs. Another line of work proposes to use P2P for content distribution [10, 21, 43, 55, 62, 65, 69]. These peers are fully voluntary and autonomous; they can join and leave the network at will and are not under the control of the CDN provider. The distribution of content to peer devices is also decentralized. In contrast, HyperEdge operates on a fundamentally different deployment model. We *rent* edge devices from third-party vendors, which grants us full control over the number, identities, and operational behavior of the Hdevices. HyperEdge thus remains a server-to-client system with centralized control, albeit composed of highly heterogeneous, low-power edge devices rather than traditional, rack-mounted CDN servers. We have deployed Hdevices across multiple continents; the deployment model also allows us to provision these edge devices based on traffic demand. This model better aligns with our business requirements, where predictability, accountability, and service-level guarantees are essential. We include more detailed comparisons to P2P-assisted CDNs in §7.

HyperEdge optimizations. Several concurrent efforts focus on individual components and optimizations of HyperEdge. Zhang et al. [67] augment our multi-path transmission protocol with a resource-aware biasing mechanism. The mechanism directs more video traffic towards cheaper or higher-quality devices, alleviating resource imbalances across the edge network. Twist [56] implements a novel joint flow control algorithm to further improve the throughput of our multi-path transmission protocol. Wei et al. [57] propose a preloading algorithm to balance the cost and transmission quality in the multi-path design, and optimize transmission scheduling [58] to further enhance its delivery efficiency. All these solutions build upon the foundation of HyperEdge. We are the first to present a complete and systematic description of the design and implementation of HyperEdge, covering its video transmission protocol, centralized tracker design, content distribution strategy, and Hdevice management. Our work also

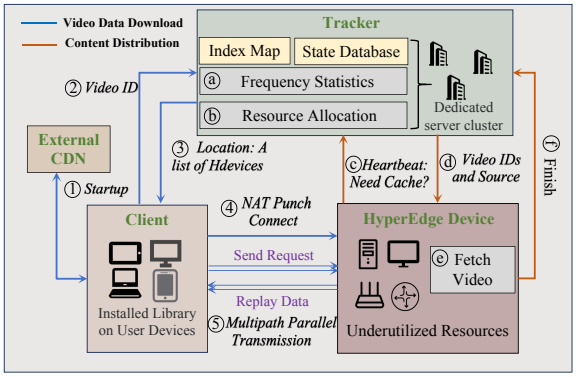


Figure 2: HyperEdge architecture overview. The figure shows the main components of HyperEdge and its two primary workflows - video downloading and content distribution.

quantifies the concrete cost savings of edge-assisted content delivery in a globally deployed system, while demonstrating its video delivery quality on par with centralized CDNs.

3 HyperEdge Overview

Figure 2 shows the overall system architecture of HyperEdge. The HyperEdge system consists of a client-side library that runs on end-user devices, a tracker cluster deployed in ByteDance data centers, and a set of Hdevices with underutilized resources running in dispersed locations. An external CDN offers auxiliary support for content delivery.

Each video is uniquely identified by an *ID*. Due to the unique characteristics of short video streaming, we partition each video into equal-length (in duration) *segments*. The default segment length is 10 seconds. The external CDN stores *all* videos. As this paper focuses on the design of HyperEdge, we treat the external CDN as a black box; readers interested in their designs may refer to prior literature [40].

Hdevices are provisioned by *third-party vendors*. ByteDance does not have direct ownership of their physical infrastructure. However, by *renting* from the vendors, ByteDance is authorized to use the compute, storage, and network resources on each Hdevice. Each Hdevice runs an authorized HyperEdge runtime that connects to our tracker system. This allows the tracker to apply centralized content distribution policies on the distributed Hdevices.

Each Hdevice stores a set of video files in its local memory and storage devices. Hdevices typically store whole videos, instead of individual segments, so the tracker only needs to map video IDs instead of segment IDs. These Hdevices are purely *passive*: Clients pull videos from them, and the tracker instructs them to store specific videos. Such a design reduces the computational requirement on Hdevices, a good match for their weaker and heterogeneous computational power.

The tracker system runs geo-distributedly over multiple data centers. We use standard replication and partitioning techniques for fault tolerance and scaling. Specifically, we deploy multiple Redis [47] servers across data centers, and apply asynchronous primary-backup replication. The tracker system acts as the main control plane for HyperEdge. It contains two main subcomponents. The first component serves as a "frontend" metadata store for end users. It maintains a table that maps video ID to a list of Hdevices that cache the video. User devices query this tracker component when downloading video segments from HyperEdge. The component also tracks user query statistics for analyzing content popularity distribution. The second tracker component manages Hdevices. It operates a database that maintains the state of Hdevices, such as their locations, bandwidth consumption, and disk utilization. The component leverages detailed Hdevice information for making fine-grained resource allocation and video distribution decisions.

The client-side library is linked into each video streaming application. It manages video transmission scheduling, Hdevice filtering, pre-loading, and transport-level flow and congestion control. The library is responsible for ensuring the quality of video playback experience to users.

Video downloading. Figure 2 illustrates the video downloading process. ① The client library requests the external CDN to download the first few segments of the target video. Using the faster CDN minimizes the initial video loading time. The library stops downloading from the CDN once it establishes connections to the Hdevices. ② In parallel to the first step, the library sends a request to the HyperEdge tracker cluster. The request contains the ID of the target video. ③ The tracker queries the content distribution map and returns a list of Hdevices storing the video segments to the client. ④ The client establishes connections to the returned Hdevices, potentially applying NAT punching for Hdevices behind private networks. ⑤ The client library starts downloading the remaining video segments. To mask the low and unreliable transmission performance of Hdevices, the client receives video data concurrently from all connected Hdevices using a multipath transmission scheme.

Content distribution. Figure 2 also shows the content distribution process. ⓐ The tracker records per-video download frequency over five-minute intervals, using it as feedback metric to gauge video popularity. ⓑ The tracker tallies the available Hdevice resources in each region. This information is combined with video popularity to derive the optimal video caching distribution, both regionally and per-server. ⓒ Periodically, each Hdevice sends a keepalive message to the tracker, reporting its status. ⓓ When replying to a keepalive, the tracker piggybacks a list of new video IDs that the Hdevice should cache. ⓔ The Hdevice fetches each new video using the download protocol analogous to the client library and stores the videos locally. ⓕ After the download completes,

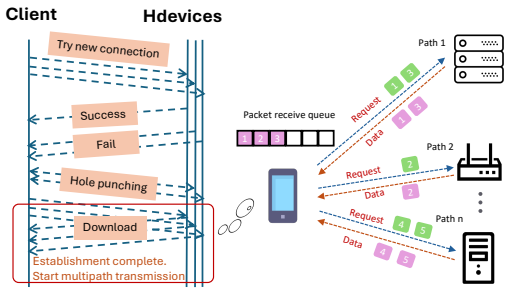


Figure 3: The process of client establishing connections (left) and multi-path parallel transmission (right) with Hdevices.

the Hdevice sends a confirmation back to the tracker. The tracker updates the video ID to Hdevice map accordingly.

4 Design Details

In this section, we detail the design of HyperEdge as overviewed in §3. We first describe the communication library loaded on each client (§4.1); next, we discuss how trackers distribute video files and manage Hdevices (§4.2); lastly, we detail the design of our Hdevice system (§4.3).

4.1 Client-side Transmission

The client-side library plays a pivotal role in orchestrating network transmission to download data from heterogeneous video sources, including Hdevices and the CDN. The library consists of three main modules: ① a connection manager that queries the Tracker for appropriate Hdevices and establishes connections (§4.1.1); ② a transmission scheduler that provides rapid and orderly data arrival in the presence of diverse path qualities (§4.1.2); ③ a segment selector that switches downloading between external CDN and HyperEdge to speed up playback initiation and handle failures graciously (§4.1.3).

4.1.1 Connection Management

The client library maintains connections to both CDN and Hdevices. We use standard approaches to establish connections to the external CDN, i.e., through content-agnostic DNS queries to nearby edge servers via HTTP [6]. The library caches CDN connections across video downloads to amortize the overhead of establishing connections.

For each requested video, a client receives a list of Hdevices that store the video data from the tracker (§4.2). By default, the list contains up to 20 Hdevices and their addresses; the client attempts to establish connections to all of them. The left of Figure 3 shows the procedure for establishing connections. For each Hdevice, the client first sends a UDP packet directly to the listed address. A response from the device signals a successful connection. Timeout in receiving responses typically indicates that the device is behind a private networks. The library applies standard hole punching techniques [20] to penetrate NAT barriers for such devices. Once established, the client sends periodic keep-alive messages to maintain the connection with the device.

Even with hole punching, a client may not connect to all Hdevices in the tracker list. The library uses the first eight successful connections for video transmission; we call this set of devices the *active transmission set*. The design simultaneously serves as a filtering mechanism: Devices that complete connection establishment faster typically indicate better network conditions such as RTT. However, due to their unreliable nature, devices in the active set may be disconnected or fluctuate in their transmission speed. To maintain service quality, the library continuously makes connection attempts to the remaining Hdevices in the list, and keeps those connections in a backup pool. It also monitors the performance of the connections in the active set, including RTT, packet loss rate, and transmission rate. Devices that drop below a quality threshold are immediately replaced by connections in the backup pool, maintaining the overall transmission quality of the active set.

A connection to an Hdevice incurs low resource overhead (a few buffers and periodic keepalive messages), so the client library keeps device connections for a short period (two minutes by default) after video download. We observe temporal localities in our workloads: Some users revisit previously viewed videos within a few minutes. Therefore, the library also implements a cache for video content with an aggressive expiration policy. Caching content longer has diminishing returns based on our large-scale A/B tests; a larger DRAM cache size also increases the crash rate on user devices.

HyperEdge encodes multiple bitrates of a video into separate files. When requesting a video, the video application selects a particular bitrate based on recent network statistics and fetches the corresponding file. HyperEdge currently does not support dynamically switching bitrates during video playback, as it requires lengthy connection re-establishment and adds complexity to the system. When the network fails to sustain the current bitrate, HyperEdge falls back to the CDN or switches to alternative Hdevices.

4.1.2 Multi-path Parallel Transmission Protocol

To provide satisfactory video playback quality, HyperEdge needs to ensure video download throughput above an SLO R_{SLO}. Since the performance of a single Hdevice is likely below this target, clients perform concurrent transmissions with all Hdevices in the active set, with the goal of aggregating their transmission throughput. This is, however, a challenging task. First, unlike traditional transport protocols, video data is transmitted along multiple network paths with distinct path conditions. Second, in contrast to MPTCP algorithms, the client is communicating with *multiple data servers* for a

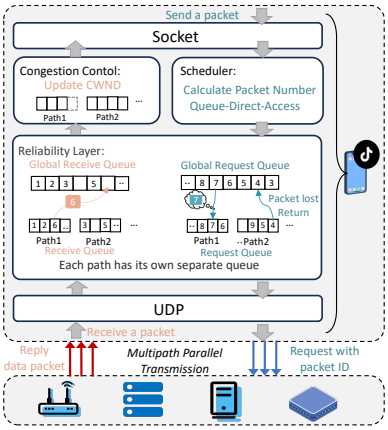


Figure 4: A more detailed illustration of the HyperEdge multi-path parallel transmission protocol

single video segment. Lastly, the transmission scheme must simultaneously guarantee *in-order delivery* of video data within the playback *deadline*.

To address these challenges, we design a novel multi-path transmission protocol for HyperEdge, as shown in the right of Figure 3. The protocol uses a *client-driven, pull-based* transmission scheme, where the client schedules and requests individual video datagram from each Hdevice. The protocol is implemented as an application-layer protocol atop UDP; as such, each requested datagram contains MTU-size-bounded video data. Figure 4 illustrates the detailed design of our protocol, including its core algorithms and workflows.

Multi-path scheduling algorithm: Simultaneously achieving our aggregated throughput target and timely in-order data delivery is nontrivial. Prior multi-path transmission approaches meet at most one of the two requirements: MPTCP [13, 19] protocols provide low-latency data stream delivery, but their transmission bandwidth is bounded by the highest-bandwidth link; P2P file sharing systems [3,9,10] can aggregate downloading throughput of multiple Hdevices, but do not provide any guarantees to meet playback deadlines.

We design a novel transmission protocol that fully utilizes the bandwidth of each connected Hdevice, while ensuring timely video stream delivery. Our main design innovation is to schedule requests based on *predicted data arrival time*: When a transmission window opens for a path, the scheduler predicts the arrival time of the requested data based on estimated path condition; it then picks a packet index that *minimizes* playback deadline violation. This *out-of-order* transmission scheme differs from previous solutions.

Concretely, the client maintains four queues for each video transmission (Figure 4): a Global Request Queue that stores the sequence number of all pending requests; a Global Receive Queue that stores the sequence number of all received (but undelivered) data; a Path Request Queue and a Path Receive Queue that function similarly but for each active Hdevice connection. The scheduler learns each path's *true performance* by monitoring how many bytes actually arrive over time on the path and how long each packet takes – including any retransmissions – rather than relying on a congestion window. It uses these measured performance to estimate the packet delivery speed on each path. When a path's transmission window frees up, the scheduler compares all paths' expected delivery speeds to predict how many packets each will finish before this one, and then pulls exactly the sequence number from the Global Request Queue that will slot into the correct arrival order. The received video data sequence numbers are placed in the Global Receive Queue and delivered to the application in sequence number order.

Reliable Data Transmission: When the library schedules a video request along a path, it records the sent timestamp of the request when inserting it into the path-local Request Queue. The library maintains a timer for each queue. When the timer expires, it locates outstanding requests in the Path Request Queue that does not have a corresponding reply in the Path Receive Queue, and picks those that have exceeded a timeout threshold. These requests (or their replies) are likely dropped in the network. The library then removes them from the path-local queue and reinserts them into the Global Request Queue. The request will later be scheduled to retransmit via a more efficient path. We don't use path-local retransmission since drops usually signal path problems. As dropped requests are closer to missing the playback deadline, insertion into the global queue ensures timely retransmission on faster paths.

4.1.3 Hybrid CDN-HyperEdge Transmission

Video transmission using Hdevices has two inherent weaknesses. First, establishing connection to an Hdevice incurs higher latency, particularly due to NAT hole-punching. Such latencies typically exceed our video startup SLOs; we provide quantitative measurements in 6.3). Second, clients may experience churns in connected Hdevices during video transmission, leading to an unsatisfactory user experience.

HyperEdge applies a hybrid CDN-HyperEdge transmission scheme to address both issues. The client library transmits the first few video segments from the CDN, which guarantees timely playback after opening a video, while it waits for Hdevice connection setup. After Hdevice connections are established, subsequent video segments are requested from HyperEdge. A key design choice is determining when to perform the switch. The cost of transmitting data from CDN is about 35% higher than that from HyperEdge. To optimize for overall cost, the switching point should be *as early as possible*, while not impacting video playback quality.

To that end, the client library concurrently connects to Hdevices while downloading from CDN. Our UDP-based connection setup is lightweight, so it has minimum impact on concurrent video transmission from CDN. When Hdevice connections are established, the client performs additional checks

before switching to HyperEdge. If the remaining video data falls below a threshold, switching to HyperEdge has little economic benefit, so the client completes all transmissions from the CDN. Next, the library estimates the network bandwidth and reliability of Hdevice based on measured RTTs and packet drops. Poor measured network quality may jeopardize transmission quality, so the library prefers the more stable CDN. Lastly, if the client loses connections to too many Hdevices (missing keepalive responses), the client also refrains from switching to HyperEdge.

To minimize the impact of Hdevice unreliability on user experience, the library switches video transmission back to CDN in certain situations. First, if the preloading of the next segment fails to complete when the segment is viewed, it signals poor HyperEdge network conditions and the client switches back to CDN. Second, any error reported by HyperEdge triggers a switch. Note that after the switch, the client stays with the CDN for all remaining segments of the video. In our experience, this "reverse-switch" happens rarely. The tracker carefully selects Hdevices with decent network connectivity; it includes redundant devices to handle potential failures and churns. The multi-path protocol also aggressively deprioritizes devices with poor transmission quality. We implement this scheme only as a fail-safe measure.

4.2 Tracker

The tracker system is responsible for the distribution of video data to Hdevices. Video content distribution is critical to the overall efficiency and QoE assurance of HyperEdge: Highly popular videos need to be stored on enough Hdevices that are geographically closer to users to meet demands. The tracker also assigns responsible Hdevices to clients upon video requests. Poor device selection would result in subpar video downloading experience and load balancing issues across Hdevices. In contrast to prior P2P systems, the centralized view of the tracker improves both decision-making processes. First, the tracker is capable of collecting video popularity statistics across different regions. This enables the tracker system to make globally-optimal decisions to distribute video data based on real-time demand information. Second, the tracker monitors real-time performance metrics of all online Hdevices. This ensures Hdevice assignment that satisfies our transmission requirements.

4.2.1 Video File Distribution

Since all client video requests go through the tracker system, we maintain global demand statistics at the tracker. At each distribution interval, the tracker collects the number of views for each video. Then it calculates the projected traffic demand of each video based on the view statistics and the video file size (views $\times$ file_size/interval). Next, the tracker uses the average transmission bandwidth of Hdevices to determine the number of copies for each video (proj_video_traffic/avg_dev_speed). The calculation ensures sufficient overall Hdevice capacity to serve each video. Note that the above calculations are done in a per-region-basis to account for the geographic locality of video popularity.

Subsequently, the tracker applies an optimization function to determine the distribution of video copies to Hdevices. The optimizer takes two types of constraints into consideration. First, it considers the capacity and status of online Hdevices including utilization of storage and bandwidth, network RTTs and reliability, and already allocated video files. We elaborate device status reporting in §4.2.3. The distribution is constrained to ensure that the allocated Hdevice resources can meet the projected user demand. Second, the optimizer also *balances* file distribution across ISPs, geographical regions, and device types. These constraints serves to reduce risks of correlated failures and to balances load across key dimensions. The optimizer finally outputs a mapping between video IDs to Hdevices which serves as the target file distribution for the next time interval.

Instead of eagerly pushing new video file assignment to each Hdevice, the tracker reuses the existing Hdevice-tracker message protocol. In particular, the tracker piggybacks the file assignment in its response to each Hdevice keepalive message. We discuss details of this message protocol in §4.3. To download newly assigned video files, Hdevice applies a transmission protocol similar to that of the client (§4.1). However, Hdevices typically have lower hardware specifications that user devices; video redistribution also has no real-time playback deadlines. Therefore, Hdevices set less aggressive transmission parameters for downloading. Once the Hdevice successfully retrieves the new video files, it informs the tracker, which updates its video-to-Hdevice mapping database.

Based on our deployment experience, this redistribution strategy quickly adapts to video popularity changes and promptly increases file coverage to meet dynamic demands. However, video redistribution consumes network bandwidth of both Hdevices and the CDN, which is a scare resource during peak hours. To control bandwidth usage, video redistribution is temporarily paused during peak hours. Instead, only videos that surge to exceptionally high popularity within a 5-minute interval will be redistributed to more Hdevices.

4.2.2 Bandwidth Resource Management

The above discussion assumes homogeneous user demands and Hdevice bandwidth resources. In reality, HyperEdge faces two forms of heterogeneity. First, HyperEdge is a shared infrastructure that serves multiple businesses (e.g., Douyin, TikTok and Xigua). Each business is further divided into different domains, each with a *priority level*. Second, Hdevices are provisioned by different vendors, each having its own network resource constraints. The tracker, therefore, needs to perform Hdevice bandwidth allocation to minimize inter-business and

inter-regional resource contention, while respecting priorities in the business domains. The tracker performs a greedy algorithm to match bandwidth resources to business domains.

Decomposing business requirements: The tracker first calculates the bandwidth requirement bw_need_{rs} for business s in region r using the fluctuation coefficient of s multiplied by the historical peak bandwidth in r. The capacity allocated to s in r is then $bw\_alloc_{rs} = bw\_need_{rs}/bw\_need_r * cap_r$, where bw_need_r represents the estimated bandwidth requirement for all businesses in r, and cap_r is the calibrated bandwidth available in that region.

Decomposing vendor allocation: In a given region r, the bandwidth allocated by a vendor v to a specific business s is determined by $bw\_alloc_{rvs} = cap_{rv}/cap_r * bw\_alloc_{rs}$. The bandwidth bw_alloc_{rvd} that v can allocate to a business domain d within r is the sum of bw_alloc_{rvs} for all the businesses in d. The expected bandwidth that v can allocate to d in r is then $exp\_bw_{rvd} = exp\_bw_d * bw\_alloc_{rvs}/bw\_alloc_v$, where exp_bw_d is derived from previous business data and our growth expectations for that business.

Bandwidth allocation: Lastly, we use a greedy algorithm to match bandwidth resource instances to each business domain. The algorithm prioritizes resource allocation to domains with higher priority.

4.2.3 Hdevice Management and Assignment

The tracker also maintains detailed information of each active Hdevice. When a Hdevice joins the HyperEdge, it contacts the tracker system and reports its current region, IP address, and associated vendor. Each Hdevice also periodically sends performance metrics such as CPU, disk, and bandwidth utilization to the tracker. The default reporting interval is 30 seconds. If the tracker fails to hear from a Hdevice, it assumes it has failed and removes it from the active Hdevice pool.

As detailed in §4.3, each Hdevice reports its operating status, including CPU, disk, and bandwidth utilization, in its keepalive protocol message to the tracker. The tracker records these metrics, as well as the region, IP address, and the associated vendor, of each online Hdevice. It uses this information for file distribution (§4.2.1) and Hdevice assignment.

In particular, when a client requests a video, the tracker selects the top K candidate Hdevices that store the file (20 by default). The selection process begins with a filtering phase, in which devices with high bandwidth utilization, located in different ISPs or regions from the client, or incompatible NAT types are ruled out. The remaining candidates are scored and ranked based on their reported operation metrics, distance to client and NAT matching, as well as historical statistics such as connection stability and quality. Finally, the tracker returns the K highest-scoring Hdevices to the client.

It should be noted that this assignment serves only as an estimation of Hdevice quality. Since the tracker lacks perfect visibility into real-time, end-to-end network conditions, the list includes redundant devices to compensate for estimation errors. The client transmission protocol further ensures playback quality through direct measurement of Hdevice metrics.

4.3 HyperEdge Devices

The central role of a Hdevice is to store and serve video data to clients. Each video is stored as a standard file in the Linux file system. We leverage the OS buffer cache for in-memory caching of video data. For content integrity, we divide each file into smaller chunks. A checksum is calculated and stored alongside each chunk.

When a Hdevice joins HyperEdge, it establishes a long-lived WebSocket connection with the tracker. Establishing this connection is simpler, since the tracker exposes public IPs. The Hdevice sends a keepalive message to the tracker every 30 seconds. It piggybacks real-time device performance metrics in its keepalive message; the tracker attaches file distribution list in the keepalive response (§4.2.1).

4.3.1 Video Storage Replacement

Due to the limited storage space, Hdevice may not have capacity to store all video files as instructed by the tracker. When capacity is reached, Hdevice deletes old video files to make space for newly downloaded videos. This replacement decision is made locally by Hdevice, i.e., the tracker is not involved in the process. Performing replacement requires both tracking information about each stored file and selecting the files to be replaced. The storage and computational overhead grows linearly with more Hdevices and videos. We therefore assign this responsibility to all Hdevices for scalability.

A good replacement strategy is crucial to ensuring satisfactory video transmission performance for HyperEdge. We apply the following heuristics to rank the replacement priority. First, videos not currently transmitting have a higher chance of being replaced. Second, we prefer replacing videos with lower access frequency. Lastly, videos not currently cached in memory are more likely to be replaced. If multiple videos have the same replacement rank, we pick the one with size closest to the new video to reduce fragmentation.

4.3.2 Unified Hdevice Management Platform

Unlike centrally managed data centers or clusters, Hdevices vary significantly in their hardware platform (e.g., x86_64, MIPS, ARM32, and ARM64), software stack (e.g., Linux, OpenWrt, and Android), performance, and reliability. This heterogeneity presents challenges in software deployment and system state monitoring. To manage these highly diverse devices, we develop an Infrastructure as a Service (Iaas) component as a unified management platform. The platform leverages Kubernetes to orchestrate Docker containers running on each Hdevice. When a Hdevice connects to HyperEdge, the

platform probes the instruction set architecture (ISA) and the operating system running on the Hdevice. It then configures its cross-compiler toolchain to produce a Hdevice software executable for the target execution environment. Software updates are pushed to the Hdevices in a similar way. The process improves operational efficiency by simplifying and automating the software deployment and update cycles.

The platform also performs continuous monitoring of Hdevice status, offering more fine-grained visibility than the tracker. It exposes real-time runtime and contextual information, such as geographic location, network type, and available memory, via a unified on-device API. This platform has proven invaluable for operational simplicity. Failures such as hardware malfunctions, OS crashes, and resource exhaustion are promptly detected and automatically resolved. Without such infrastructure, identifying and mitigating failures across heterogeneous device types would be more labor-intensive and error-prone. The system enables reaction to frequent device churns within seconds. Moreover, the platform dynamically adds or removes Hdevices to accommodate fluctuating traffic demands across regions. This dynamic resource allocation provides performance and reliability guarantees to HyperEdge, while minimizing resource over-provisioning.

5 Cost Analysis

Our main motivation of leveraging edge devices for content distribution is cost efficiency. Does HyperEdge actually reduce the cost of serving video content to users? In this section, we present detailed economic analysis of deploying HyperEdge and demonstrate its cost saving benefits.

As explained in §3 and §4, our content delivery architecture uses a mixture of HyperEdge and traditional CDNs to handle user video demands. The total cost, C_{total}, of content delivery is modeled using the following formula:

$$C_{total} = U_{HE} * T_{HE} + U_{CDN} * T_{CDN} + C_{H \leftarrow C}$$

In the formula, U_{HE} and U_{CDN} represent the unit traffic cost of HyperEdge and CDN, respectively. Note that they encompass all operational costs (e.g., hardware, manpower, network bandwidth) to serve a unit of traffic. T_{HE} and T_{CDN} are the total traffic handled by HyperEdge and CDN [†]. Lastly, $C_{H \leftarrow C}$ is the cost of sourcing data from CDN to HyperEdge.

To avoid revealing sensitive traffic information at ByteDance, our analysis is based on the cost of handling 30 TB/s of peak user traffic. ByteDance purchases CDN services from external providers. In the market where ByteDance operates, CDN providers charge by *peak traffic*, not the actual usage. The monthly traffic cost of CDN, i.e., U_{CDN}, is around $1 per MB/s in our deployed market. We do not further break down the cost of CDN, since its operation is outside the management of ByteDance. Our hybrid architecture ensures that the CDN is auxiliary, only facilitating initial video downloading and HyperEdge fallback. As such, CDN handles just 10% of the total user traffic on average. To handle 30 TB/s of peak user traffic, the annual CDN cost is thus around $36M.

Next, we analyze the cost of operating HyperEdge, i.e., U_{HE}, which comprises three parts. First, HyperEdge rents edge devices from third-party vendors. The hardware and electricity cost of these devices are fractional of those of data center servers. As illustrated in §2.2, they use public networks, which are substantially cheaper than dedicated commercial networks in our deployed market. Moreover, the computation and network resources are *under-utilized* on these devices; our content delivery *shares* the cost with other services deployed on the devices. Consequently, these third-party vendors offer attractive prices: On average, the offered unit traffic cost of edge networks is roughly 55% that of a CDN (edge vendors also charge by peak usage). With 30 TB/s of peak traffic, the annual cost of the edge network is approximately $198M.

Next, HyperEdge deploys around 20 dedicated servers to run HyperEdge trackers and management services in ByteDance's data centers for 30 TB/s of peak traffic. The cost of running these servers (including hardware, network, power, and maintenance) is comparable to other cloud companies, at roughly $72K per year. Lastly, for 30 TB/s of traffic, we employs 20 dedicated engineers and system administrators to maintain HyperEdge. With an average annual salary of $100K, the manpower cost is roughly $2.4M per year.

Besides operational cost within HyperEdge, edge devices also fetch video data from the CDN when they are not available within the edge network. This data sourcing, hypothetically, incurs additional CDN cost ($C_{H \leftarrow C}$) if done on-demand. We design a prefetching strategy, based on traffic patterns, to minimize such cost. Specifically, services that use our content delivery network show clear *diurnal traffic patterns*. As shown in Figure 6, user demand during peak hours (18:00 – 22:00) is 20% - 5× higher than that of the non-peak hours. Exploiting such patterns, HyperEdge prefetches most video content from the CDN "asynchronously", before demand arrives. The same graph shows that HyperEdge heavily relies on the under-utilized CDN bandwidth during non-peak hours for fetching data to the edge. This cost-aware scheme virtually eliminates the additional CDN cost $C_{H \leftarrow C}$.

Cost decomposition and saving We illustrate the cost decomposition of HyperEdge in Figure 5. The cost of the edge network dominates, at 83.7%. The hardware and manpower cost are negligible, combined to be around 1%. The total cost of operating HyperEdge is roughly $235M per year. To handle the same total peak traffic (30 TB/s) using CDN alone costs $360M per year. Therefore, our HyperEdge system offers a 35% overall cost reduction for content delivery at ByteDance.

Recognizing the cost saving benefits, ByteDance has been gradually shifting user video traffic from monolithic CDN to HyperEdge in our production network. Figure 7 shows histor-

[†] As explain later, cost is calculated by peak traffic handled by each system.

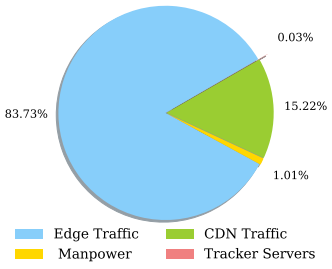


Figure 5: HyperEdge cost decomposition

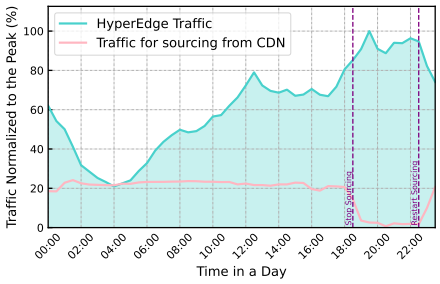


Figure 6: HyperEdge and CDN sourcing traffic patterns within a day

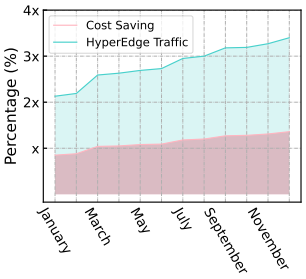


Figure 7: HyperEdge traffic percentage and total savings ratio

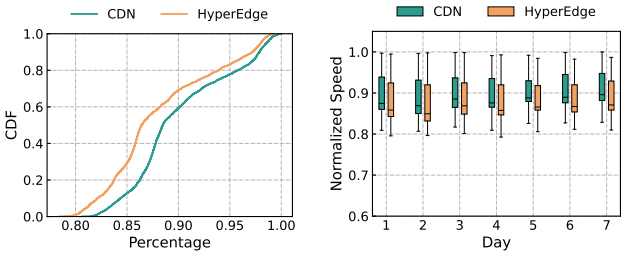
ical data on the traffic usage ratio handled by HyperEdge, and the corresponding savings ratio, which increases linearly with higher traffic proportion. Our deployment statistics clearly validate the economic benefits of our HyperEdge design.

6 Evaluation

HyperEdge has been operational within ByteDance for over six years. We conducted large-scale A/B testing consisting of two comparison groups in deployed production networks. The first group serves user video request solely from CDN, while the second group uses our HyperEdge system. Each group includes approximately ten million participants, generating billions of video playbacks daily. The comparison system, ByteDance CDN, is an industry-leading content delivery network which is optimized for both long and short video services. Our results indicate that HyperEdge could closely match the quality of experience (QoE) offered by HyperEdge CDN, while reducing operational cost by significant margin. This demonstrates the potential of HyperEdge as a reliable content delivery solution. Due to our confidentiality policy, all data are normalized to the maximum value obtained in each experiment (across all systems) unless otherwise specified. Data outliers are not shown in the graphs. For simplicity, we refer to ByteDance CDN simply as CDN in this section.

6.1 Video Transmission Speed

We first examine the video transmission speed of HyperEdge (excluding initial CDN download), where speed is calculated as the average downloading throughput of all users in five-minute intervals. Higher transmission speed correlates directly to higher video bitrate and lower rebuffering rate. Figure 8(a) compares the CDF of transmission speed between CDN and HyperEdge over a one-month period; Figure 8(b) groups the data into days-of-the-week to show fine-grained temporal information. The results indicate that the median transmission speed of CDN is 2.89% higher than that of HyperEdge. CDN achieves higher speed mainly due to better server hardware and faster network infrastructure. To our surprise, the gap between CDN and HyperEdge is only marginal. Furthermore, we observe that even HyperEdge's 99th percentile tail speed is within 2.58% that of CDN. The main contributor of HyperEdge's fast transmission speed is our novel multi-path parallel transmission protocol (§4.1.2). By effectively *aggregating* the lower network bandwidth of individual edge devices, our transmission protocol provides video downloading performance comparable to traditional CDN. When grouping data into days of a week, we observe that the results remain stable throughout the week for both systems. HyperEdge thus can consistently provide speed on par with CDN without significant variance.

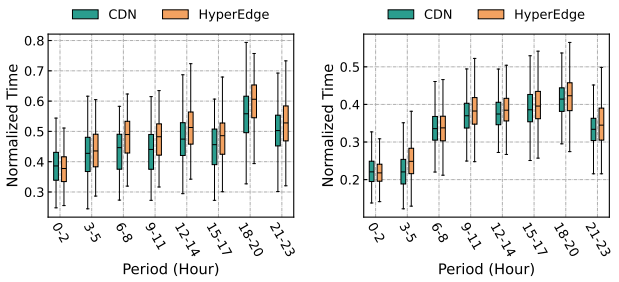


(a) CDF of transmission speed (b) Box plot for per-day data

Figure 8: Video transmission speed of CDN and HyperEdge. Data are collected over a period of a month. The box plot groups data into days of a week.

6.2 Video Rebuffering Rate and Time

Figure 9 shows the rebuffering rate, which counts the number of rebuffering events, and the rebuffering time, which measures the duration of rebuffering in each video, during various hours-f-day over one month. Both CDN and HyperEdge experience increased rebuffering events during peak hours, such as 6pm to 8pm. However, the relative difference between the two systems is small: for rebuffering rate, the median (99th percentile) of HyperEdge is 6.41% (4.51%) higher than that of CDN respectively; HyperEdge also achieves median (99th



(a) Rebuffering Rate (b) Rebuffering Time

Figure 9: Comparing video rebuffering metrics between CDN and HyperEdge during various hours-of-day over one month.

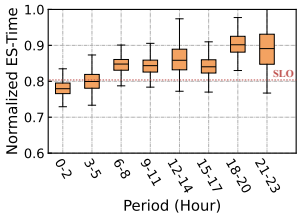


Figure 10: Normalized HyperEdge establishment time. The dotted line shows the video startup SLO.

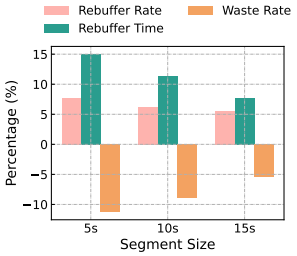


Figure 11: Impact of different segment sizes on rebuffering and preloading wastage.

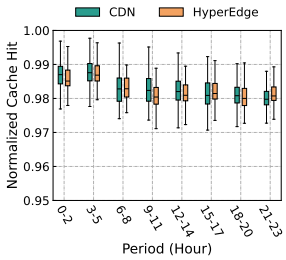


Figure 12: Cache hit ratio of CDN and HyperEdge over one month.

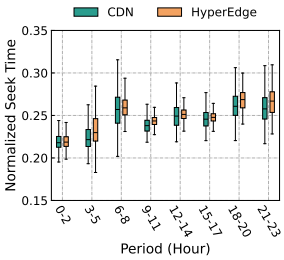


Figure 13: Video seek time of CDN and HyperEdge over one month.

percentile) rebuffering time within 3.41% (4.57%) that of CDN. Even during peak hours, HyperEdge maintains similar rebuffering performance relative to CDN. Our results demonstrated that HyperEdge can maintain similar user-perceived experience in terms of video playback smoothness compared to traditional CDN. Our multi-path transmission protocol both provides high aggregating speed and masks Hdevice connection unreliability; initial parallel CDN download hides the influence of long connection time with Hdevices.

6.3 HyperEdge Establishment Time

We leverage faster CDN establishment to ensure satisfactory video startup time before switching to HyperEdge. However, the sooner the clients can establish connection to Hdevices, the larger percentage of video data is transmitted through HyperEdge, which reduces overall system cost. Figure 10 shows the normalized HyperEdge establishment time measured on the clients. The latency includes both tracker interactions and establishing connections with Hdevices. For reference, we also plot the video startup time SLO as a dotted line in the graph. The median (99th percentile) establishment latency for HyperEdge exceeds the SLO by 3.08% (17.12%), mostly due to additional tracker communication and longer Hdevice connection setup. The absolute latency exceeding the SLO, however, is below a few hundred milliseconds. As such, most clients can quickly switch to HyperEdge for video downloading. This results in overall traffic costs reduction by around 35%. Our scheme therefore offers significant economic benefits but without compromising video streaming experience.

6.4 Segment Size

As discussed in §3, HyperEdge partitions each video into equal-length *segments*. Both client-side video preloading and CDN-HyperEdge switching operate at per-segment granularity. Specifically, the client library only preloads one segment ahead to minimize download data wastage. The size of each segment, however, poses a cost vs. QoE trade-off. As users frequently request for next videos during playback, preloading larger segments can lead to lower utilization of the preloaded data; smaller preloaded segments, however, risk introducing more frequent rebuffering which impacts user experiences.

We test HyperEdge with various segment sizes (5s, 10s, and 15s segments) in a real-world network. We measured both QoE (represented using rebuffering rate and time) and data waste rate, which is calculated by dividing the sum of unwatched data by the total downloaded data across all clients. Figure 11 shows the measurements normalized to a baseline that always preloads the entire video. The experiments validate our hypothesis: reducing the segment size linearly reduces data wastage, but simultaneously increases rebuffering time in a linear fashion. Based on our production requirements on data cost and user experience, we eventually settle down on 10s segments which offer a satisfactory trade-off.

6.5 Cache Hit Ratio

Both HyperEdge and CDN caches video content in memory: HyperEdge caches data in Hdevices, while CDN uses the memory of edge servers. Accessing videos that are cached in memory improves overall downloading performance. We measured the cache hit ratio of HyperEdge and CDN over a course of a month, and group the results into 3-hour buckets, as shown in Figure 12. All results are normalized to the

maximum value. HyperEdge matches closely with the cache hit ratio of CDN. CDN achieves near-optimal cache hit rate for any workload, since its edge servers can cache content based on request patterns, refilling cache misses from higher hierarchies. Hdevices, however, rely solely on the tracker to store video data. Our predictive deployment strategy outlined in §4.2 is a main contributor why our cache hit rate can match that of CDN. The tracker redistributes video data to Hdevices based on real-time video access patterns, both in temporal and spatial localities. The global view of the tracker ensures optimal distribution of video files, improving the likelihood that popular videos in the region are cached in memory.

6.6 Seek Time

Though most users view videos sequentially, in practice, some may also perform random seek in video playback. The video loading latency between user dragging the progress bar and displaying the corresponding frame is critical to user experience. We measured this *seek time* for both HyperEdge and CDN, and plot the results normalized to the maximum value in Figure 13. Similar to previous experiments, the results are grouped into 3-hour buckets. HyperEdge offers comparable seek performance as traditional CDN: The median seek time of HyperEdge is only 3.56% higher than that of CDN. When a seek event occurs, HyperEdge clients likely have already established connections with Hdevices. Our multi-path protocol then ensures data transmission speed comparable to CDN for the new segment. Even in the rare case of HyperEdge network issues, the client library quickly switches video transmission to the CDN to avoid noticeable delay or playback failures.

7 Related Work

Content distribution systems. Most commercial CDNs are infrastructure-based. Akamai deploys widely distributed servers across the world [40], while Limelight [7] uses fewer, stronger servers in major cities. More recent CDN architectures, such as LiveNet [33], build on a flat CDN overlay with a centralized controller for global optimization to reduce latency. However, CDNs incur significant infrastructure and operational costs. The main goal of HyperEdge is to reduce content distribution cost, while ensuring satisfactory QoE.

Traditional peer-to-peer (P2P) systems, such as BitTorrent [59], Kazaa [1], and IPFS [53], primarily focus on distributed data storage and delivery through distinct file retrieval mechanisms - ranging from centralized-cache-based indexing [3,9,60] to decentralized DHT-based lookup [28,30,37,46, 49]. However, these systems optimize primarily for file download completion time rather than meeting real-time playback deadlines, making them unsuitable for streaming applications.

Several studies have adapted P2P architectures, such as tree-based and mesh-based approch, for video-on-demand (VoD) services [14, 22, 26, 38, 51]. While these solutions leverage peer resources for content distribution, they require users to maintain persistent storage of video files and provide continuous upload bandwidth. Even with incentive mechanisms like tit-for-tat data exchange, peers retain autonomy to refuse service requests and selectively store preferred content, which risks persistent service unavailability [50].

To address reliability challenges, hybrid P2P-CDN architectures have emerged [21,43,55,65]. Notable examples include RCA [5] and NetSession [69], which combine CDN guarantees with P2P scalability. Nevertheless, the P2P components in these systems remain outside CDN operators' administrative control, preventing granular file management and quality-of-service enforcement across the hybrid infrastructure. In contrast, Hdevices are fully managed by ByteDance, and therefore exhibit lower churn rate. Furthermore, HyperEdge remains compatible with a P2P transmission mechanism.

Multi-path protocols and scheduling algorithms. Multi-path transmission protocols leverage multiple interfaces on modern hardware [11, 19, 23]. Protocols like MPTCP [19] and MPQUIC [13] augment single-path TCP and UDP to enable end-to-end multi-path transmissions. MPTCP scheduling has evolved from its original round-robin approach [27] to the minRTT algorithm in the Linux kernel. Algorithms such as Otias [64], ECF [34], and BLEST [18] focus on optimizing fast paths, potentially reducing aggregate throughput. Redundancy schemes like TWC [61] and XLINK [70] ensure structured data transmission while addressing the HOL challenge, but impacts overall throughput. Unlike these scenarios, our multi-path protocol transmits data from *multiple* data sources. Furthermore, our protocol ensures both aggregated path bandwidth, and timely in-order data delivery.

8 Conclusion

HyperEdge is a new approach to content delivery by organizing heterogeneous edge devices into a powerful platform that ensures quality video service. Relying on a centralized control plane, HyperEdge distributes files to Hdevice to dynamically adjust resources according to the demand. Our design minimizes the computational requirements of Hdevice, aligning well with their diverse and generally weaker capabilities. To date, HyperEdge has been operating for six years, saving ByteDance hundreds of millions of dollars while maintaining performance comparable to commercial CDNs.

Acknowledgments

This work is partly supported by the National Key R&D Program of China under Grant No. 2022YFB2901700. Jialin Li is supported by a Singapore Ministry of Education, Academic Research Fund Tier 1 (T1 251RES2104) and a Singapore Ministry of Education, Academic Research Fund Tier 2 (MOE-T2EP20222-0016).

References

[1] Kazaa. http://www.kazaa.com.

[2] Statistics. https://www.statista.com/companies/c/25087492/akamai-technologies.

[3] The gnutella protocol specification v.0.4. http://www.clip2.com/GnutellaProtocol04.pdf, March 2001.

[4] ADHIKARI, V. K., GUO, Y., HAO, F., VARVELLO, M., HILT, V., STEINER, M., AND ZHANG, Z.-L. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *IEEE INFOCOM* (2012).

[5] ADITYA, P., ZHAO, M., LIN, Y., HAEBERLEN, A., DRUSCHEL, P., MAGGS, B., AND WISHON, B. Reliable client accounting for {P2P-Infrastructure} hybrids. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)* (2012).

[6] BUYYA, R., PATHAN, M., AND VAKALI, A. *Content delivery networks*, vol. 9. Springer Science & Business Media, 2008.

[7] CANALI, C., CORBELLI, A., AND LANCELLOTTI, R. Designing a private cdn with an off-sourced network infrastructure: model and case study. In *Software, Telecommunications and Computer Networks, SoftCOM* (2018).

[8] CHA, M., KWAK, H., RODRIGUEZ, P., AHN, Y.-Y., AND MOON, S. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on networking 17*, 5 (2009), 1357–1370.

[9] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. W. Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies: international workshop on design issues in anonymity and unobservability Berkeley* (2001).

[10] COHEN, B. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems* (2003).

[11] CONINCK, Q., AND BONAVENTURE, O. Multipath extensions for quic (mp-quic). *IETF, Individual Submission, Internet Draft draftdeconinck-quic-multipath-04* (2020).

[12] CUI, S., ASGHAR, M. R., AND RUSSELLO, G. Multi-cdn: Towards privacy in content delivery networks. *IEEE Transactions on Dependable and Secure Computing 17*, 5 (2018), 984–999.

[13] DE CONINCK, Q., AND BONAVENTURE, O. Multipath quic: Design and evaluation. In *Proceedings of the international conference on emerging networking experiments and technologies* (2017).

[14] DO, T. T., HUA, K. A., AND TANTAOUI, M. A. P2vod: Providing fault tolerant video-on-demand streaming in peer-to-peer environment. In *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)* (2004), vol. 3, IEEE, pp. 1467–1472.

[15] EGEVANG, K., AND FRANCIS, P. The ip network address translator (nat). Tech. rep., 1994.

[16] EGEVANG, K., AND FRANCIS, P. Rfc1631: The ip network address translator (nat), 1994.

[17] FARAHANI, R., AMIRPOUR, H., TASHTARIAN, F., BENTALEB, A., TIMMERER, C., HELLWAGNER, H., AND ZIMMERMANN, R. Richter: hybrid p2p-cdn architecture for low latency live video streaming. In *Mile-High Video Conference* (2022).

[18] FERLIN, S., ALAY, Ö., MEHANI, O., AND BORELI, R. Blest: Blocking estimation-based mptcp scheduler for heterogeneous networks. In *IFIP networking conference and workshops* (2016).

[19] FORD, A., RAICIU, C., HANDLEY, M., AND BONAVENTURE, O. Tcp extensions for multipath operation with multiple addresses. Tech. rep., 2013.

[20] FORD, B., SRISURESH, P., AND KEGEL, D. Peer-to-peer communication across network address translators. In *USENIX Annual Technical Conference, General Track* (2005).

[21] FREEDMAN, M. J. Experiences with coralcdn: A five-year operational view. In *NSDI* (2010), pp. 95–110.

[22] GRAFFI, K., GROSS, C., STINGL, D., HARTUNG, D., KOVACEVIC, A., AND STEINMETZ, R. Lifesocial. kom: A secure and p2p-based solution for online social networks. In *2011 IEEE consumer communications and networking conference (CCNC)* (2011), IEEE, pp. 554–558.

[23] GUO, Y. E., NIKRAVESH, A., MAO, Z. M., QIAN, F., AND SEN, S. Accelerating multipath transport through balanced subflow completion. In *Proceedings of the Annual International Conference on Mobile Computing and Networking* (2017).

[24] GUPTA, T., CROOKS, N., MULHERN, W., SETTY, S., ALVISI, L., AND WALFISH, M. Scalable and private media consumption with popcorn. In *13th USENIX symposium on networked systems design and implementation (NSDI 16)* (2016), pp. 91–107.

[25] HASANOV, N. Content delivery networks (cdn): Opportunities, challenges and future perspectives. *Computer network technology* (2023).

[26] HEI, X., LIANG, C., LIANG, J., LIU, Y., AND ROSS, K. W. A measurement study of a large-scale p2p iptv system. *IEEE transactions on multimedia 9*, 8 (2007), 1672–1687.

[27] IMADUDDIN, M. F., PUTRADA, A. G., AND KARIMAH, S. A. Multipath tcp scheduling performance analysis and congestion control on video streaming on the mptcp network. In *International Conference on Software Engineering & Computer Systems and International Conference on Computational Science and Information Management* (2021).

[28] ISDAL, T., PIATEK, M., KRISHNAMURTHY, A., AND ANDERSON, T. Privacy-preserving p2p data sharing with oneswarm. *ACM SIGCOMM computer communication review 40*, 4 (2010), 111–122.

[29] JIA, Y., BAI, G., SAXENA, P., AND LIANG, Z. Anonymity in peer-assisted cdns: Inference attacks and mitigation. *Proceedings on Privacy Enhancing Technologies* (2016).

[30] KAASHOEK, M. F., AND KARGER, D. R. Koorde: A simple degree-optimal distributed hash table. In *Peer-to-Peer Systems II: Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22, 2003. Revised Papers 2* (2003), Springer, pp. 98–107.

[31] KANG, S., AND YIN, H. A hybrid cdn-p2p system for video-on-demand. In *Future Networks* (2010).

[32] KIHEI, B., DAVISON, T., OKPOK, M., AND SONG, J. Comparison of v2n stun/turn round trip time performance on a public 5g network. In *IEEE 96th Vehicular Technology Conference (VTC2022-Fall)* (2022).

[33] LI, J., LI, Z., LU, R., XIAO, K., LI, S., CHEN, J., YANG, J., ZONG, C., CHEN, A., WU, Q., ET AL. Livenet: a low-latency video transport network for large-scale live streaming. In *Proceedings of the ACM SIGCOMM 2022 Conference* (2022).

[34] LIM, Y.-S., NAHUM, E. M., TOWSLEY, D., AND GIBBENS, R. J. Ecf: An mptcp path scheduler to manage heterogeneous paths. In *International conference on emerging networking experiments and technologies* (2017).

[35] LIU, Y., GUO, Y., AND LIANG, C. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications 1* (2008), 18–28.

[36] MA, Z., ROUBIA, S., GIROIRE, F., AND URVOY-KELLER, G. When locality is not enough: Boosting peer selection of hybrid cdn-p2p live streaming systems using machine learning. In *IFIP TMA 2021-Network Traffic Measurement and Analysis Conference* (2021).

[37] MAYMOUNKOV, P., AND MAZIERES, D. Kademlia: A peer-to-peer information system based on the xor metric. In *International workshop on peer-to-peer systems* (2002), Springer, pp. 53–65.

[38] NANAO, S., MASUYAMA, H., KASAHARA, S., AND TAKAHASHI, Y. Performance analysis of data block synchronization mechanism in coolstreaming. In *Proceedings of the 5th International Conference on Queueing Theory and Network Applications* (2010), pp. 53–58.

[39] NETFLIX. Open connect appliance deployment guide. https://openconnect.netflix.com/deploymentguide.pdf, 2018. Accessed: April 26, 2025.

[40] NYGREN, E., SITARAMAN, R. K., AND SUN, J. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review* (2010).

[41] OTTO, J. S., SÁNCHEZ, M. A., RULA, J. P., AND BUSTAMANTE, F. E. Content delivery and the natural evolution of dns: remote dns trends, performance issues and alternative solutions. In *Internet Measurement Conference* (2012).

[42] PENG, G. Cdn: Content distribution network. *arXiv preprint cs/0411069* (2004).

[43] PETERSON, R., AND SIRER, E. G. Antfarm: Efficient content distribution with managed swarms. In *NSDI* (2009), vol. 9, pp. 107–122.

[44] PLAGEMANN, T., GOEBEL, V., MAUTHE, A., MATHY, L., TURLETTI, T., AND URVOY-KELLER, G. From content distribution networks to content networks—issues and challenges. *Computer Communications* (2006).

[45] RAMZAN, N., PARK, H., AND IZQUIERDO, E. Video streaming over p2p networks: Challenges and opportunities. *Signal Processing: Image Communication 27*, 5 (2012), 401–411.

[46] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SHENKER, S. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (2001), pp. 161–172.

[47] Reids key-value database. https://www.cloudflare.com/.

[48] SILVA, S. D., MOKHTAR, S. B., CONTIU, S., DANIEL, N., LAURENT, R., AND ETIENNE, R. Privatube: Privacy-preserving edge-assisted video streaming [c]. In *international middleware conference* (2019), pp. 189–201.

[49] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (San Diego, California, USA, 2001), SIGCOMM '01, Association for Computing Machinery.

[50] STUTZBACH, D., AND REJAIE, R. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (2006), pp. 189–202.

[51] TAN, B., AND MASSOULIÉ, L. Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM transactions on networking 21*, 2 (2012), 566–579.

[52] TANG, S., ALOWAISHEQ, E., MI, X., CHEN, Y., WANG, X., AND DOU, Y. Stealthy peers: Understanding security and privacy risks of peer-assisted video streaming. In *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2024), IEEE, pp. 324–337.

[53] TRAUTWEIN, D., RAMAN, A., TYSON, G., CASTRO, I., SCOTT, W., SCHUBOTZ, M., GIPP, B., AND PSARAS, Y. Design and evaluation of ipfs: a storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference* (2022), pp. 739–752.

[54] VO, V., LAI, S., YUAN, X., NEPAL, S., AND LI, Q. Oblivcdn: A practical privacy-preserving cdn with oblivious content access. *arXiv preprint arXiv:2501.07262* (2025).

[55] VU, L., GUPTA, I., NAHRSTEDT, K., AND LIANG, J. Understanding overlay characteristics of a large-scale peer-to-peer iptv system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 6*, 4 (2010), 1–24.

[56] WANG, H., ZHANG, R., LI, C., XUE, Z., PENG, Y., PANG, X., ZHANG, Y., REN, S., AND SHI, S. Twist: A multi-site transmission solution for on-demand video streaming. *Proceedings of the ACM on Networking 2*, CoNEXT2 (2024), 1–19.

[57] WEI, D., ZHANG, J., LI, H., XUE, Z., PENG, Y., AND HAN, R. Multipath smart preloading algorithms in short video peer-to-peer cdn transmission architecture. *IEEE Network* (2023).

[58] WEI, D., ZHANG, J., LI, H., XUE, Z., PENG, Y., PANG, X., LIU, Y., HAN, R., AND LI, J. Pscheduler: Qoe-enhanced multipath scheduler for video services in large-scale peer-to-peer cdns. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications* (2024), IEEE, pp. 2508–2517.

[59] XIA, R. L., AND MUPPALA, J. K. A survey of bittorrent performance. *IEEE Communications surveys & tutorials 12*, 2 (2010), 140–158.

[60] XIE, H., AND YANG, Y. R. A measurement-based study of the skype peer-to-peer voip performance. In *IPTPS* (2007), vol. 7, Citeseer, pp. 1–6.

[61] XING, Y., XUE, K., ZHANG, Y., HAN, J., LI, J., LIU, J., AND LI, R. A low-latency mptcp scheduler for live video streaming in mobile networks. *IEEE Transactions on Wireless Communications* (2021).

[62] XU, D., KULKARNI, S. S., ROSENBERG, C., AND CHAI, H.-K. Analysis of a cdn–p2p hybrid architecture for cost-effective streaming media distribution. *Multimedia Systems 11* (2006), 383–399.

[63] XU, N., YANG, J., NEEDHAM, M., BOSCOVIC, D., AND VAKIL, F. Toward the green video cdn. In *IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing* (2010).

[64] YANG, F., WANG, Q., AND AMER, P. D. Out-of-order transmission for in-order arrival scheduling for multipath tcp. In *International conference on advanced information networking and applications workshops* (2014).

[65] YIN, H., LIU, X., ZHAN, T., SEKAR, V., QIU, F., LIN, C., ZHANG, H., AND LI, B. Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky. In *ACM international conference on Multimedia* (2009).

[66] ZHANG, G., LIU, W., HEI, X., AND CHENG, W. Unreeling xunlei kankan: Understanding hybrid cdn-p2p video-on-demand streaming. *IEEE Transactions on Multimedia* (2014).

[67] ZHANG, R.-X., WANG, H., SHI, S., PANG, X., PENG, Y., XUE, Z., AND LIU, J. Enhancing resource management of the world's largest {PCDN} system for {On-Demand} video streaming. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)* (2024), pp. 951–965.

[68] ZHANG, Y., GAO, C., GUO, Y., BIAN, K., JIN, X., YANG, Z., SONG, L., CHENG, J., TUO, H., AND LI, X. Proactive video push for optimizing bandwidth consumption in hybrid cdn-p2p vod systems. In *IEEE INFOCOM* (2018), IEEE.

[69] ZHAO, M., ADITYA, P., CHEN, A., LIN, Y., HAEBERLEN, A., DRUSCHEL, P., MAGGS, B., WISHON, B., AND PONEC, M. Peer-assisted content distribution in akamai netsession. In *Proceedings of the 2013 conference on Internet measurement conference* (2013).

[70] ZHENG, Z., MA, Y., LIU, Y., YANG, F., LI, Z., ZHANG, Y., ZHANG, J., SHI, W., CHEN, W., LI, D., ET AL. Xlink: Qoe-driven multi-path quic transport in large-scale video services. In *SIGCOMM* (2021).

[71] ZIRNGIBL, M., JOYNER, C., STULZ, L., DRAGONE, C., PRESBY, H., AND KAMINOW, I. Larnet, a local access router network. *IEEE Photonics Technology Letters* (1995).

A Experience and Lessons

The Evolution from Single-Path Transmission to Multi-Path Transmission: The multi-path transmission protocol of HyperEdge has evolved from a single-path transmission protocol based on specific system requirements and technical considerations. Initially, HyperEdge opted for a traditional

TCP single-path transmission, primarily because of its relatively simple implementation and widespread compatibility, which is a common choice in many similar systems. However, over time, its drawbacks became apparent. Firstly, single-path transmission exhibited significant limitations in speed, and secondly, TCP-based protocols struggled with NAT traversal. This prompted HyperEdge to transition to a UDP-based multi-path transmission protocol. To achieve this, we carefully designed video file segmentation, transmission scheduling, and flow control mechanisms, effectively overcoming the performance and technical limitations of traditional TCP.

Download Failure Pinpointing: Download failures are a common issue during the initial operation of the HyperEdge system. Initially, we were unable to accurately pinpoint the specific reasons causing these failures, as they could stem from various factors. For example, cold video files may fail to be queried due to ineffective deployment, some devices may be overloaded and unable to respond promptly, or there could be failures in NAT traversal. The inability to effectively identify the root causes of these failures hindered us from proposing corresponding solutions. Therefore, our core solution was to establish an end-to-end traffic loss funnel model. Each layer of the funnel corresponds to every step of the download process from start to finish, such as file querying, establishing connections, and NAT traversal. Each step in the connection process is labeled accordingly, and the location of failures is automatically categorized into the corresponding layer of the funnel. This allows us to continuously monitor the failure rates at each layer of the funnel, identify and optimize those steps with the greatest potential for improvement. This approach also led to subsequent solutions such as file distribution and bandwidth load balancing.

Isolate from User's Services: HyperEdge utilizes the remaining resources of edge shared devices for data storage and transmission. Hence, priority should be given to ensuring the original services of the device. Careful isolation measures should be implemented. When the demand for user's services increases, HyperEdge will immediately release the resources already in use on the device. Additionally, HyperEdge avoids using all the remaining resources (for example, only using 80%) to prevent the situation where resources are not released in time and thus affecting user's services.

Content Pre-distribution Optimization: In HyperEdge, real-time data redistribution will be halted during peak periods to reduce back-to-source bandwidth and prevent congestion. Nevertheless, this might result in an insufficient number of copies of highly popular videos being deployed during peak times. Users who cannot be served will switch to CDN, and the capacity utilization of Hdevices is inadequate. Fortunately, we observe that the distribution of video popularity is distinctly differentiated, and the proportion of high-popularity video files is relatively low. Hence, HyperEdge will pre-distribute these videos prior to the peak period, predict their popularity, and distribute more copies accordingly.

HyperEdge Disaster Recovery Measures: Although the multi-path transmission feature of HyperEdge offers a quantum of robustness, disaster recovery must still be considered to prevent business collapse. However, as the traffic volume of HyperEdge increases, it becomes difficult to find third-party services to provide disaster recovery. Hence, HyperEdge needs to implement disaster recovery on its own. Specifically, this involves making backups between different vendors and setting aside some emergency equipment.

Privacy Considerations in HyperEdge: Data security is a crucial aspect of content distribution systems. Unlike purely P2P systems or peer-assisted CDNs, HyperEdge maintains strict control over edge resources. We audit and verify all stored video content to prevent malicious uploads and t ampering [52, 54]. Additionally, we employ centralized key distribution mechanisms such as TLS to encrypt the content, enhancing the security of data in transit. However, privacy risks still persist in content distribution systems. For instance, shared devices in our network could potentially monitor user access. To mitigate this, we have established privacy agreements with third-party device providers. Moreover, considerable existing research focuses on enhancing the privacy of user access [12, 24, 29, 48], which we could leverage for future improvements to our system.

Security of Hdevices: Deploying a large-scale Hdevice network inevitably raises concerns regarding their security. We design mechanisms to address four categories of security vulnerabilities. 1) *Content security*: HyperEdge never cache non-public videos on any Hdevice. All cached videos are also encrypted and integrity-checked upon request. 2) *User privacy*: HyperEdge excludes any personal identifiers in its content delivery protocol. It only maintains point-to-point IP mappings, which decouple user identities from video traffic. 3) *Vendor privacy*: All Hdevice information is virtualized and anonymized. Consequently, Hdevices only expose IP addresses, without revealing vendor or owner identities. 4) *System security*: The centralized HyperEdge tracker system is protected by the same security mechanisms as other core infrastructure at ByteDance. All HyperEdge endpoints are mutually authenticated; HyperEdge also verifies the capabilities of each reported device at runtime to filter out spoofed or malicious devices.

Cost Implications to Device Owners and ISPs: Deploying HyperEdge introduces no additional cost to owners of Hdevices. All Hdevices are provisioned by third-party vendors, who compensate participating users with rent or other forms of incentive (e.g., reward points). These users thus join the network with informed information and consent, and their interests are not compromised. For ISPs, however, the cost implication is more complex. Commercial ISPs often rely on high-margin uplink bandwidth services to subsidize low-cost residential broadband. When content is distributed via edge devices, part of the traffic bypasses these premium uplink services, potentially reducing ISPs' revenue from commercial

bandwidth sales. We view this as an opportunity to rebalance costs rather than a pure loss. By collaborating with large content providers (CPs), ISPs can integrate CPs' distribution logic with their own network resources to build intelligent, efficient edge caching systems. Such cooperation can reduce backbone and international transit expenses, lower overall operational costs, and simultaneously enhance user QoE. In the long run, this approach not only preserves ISP profitability but also unlocks the economic potential of edge infrastructures to handle extreme traffic scenarios.

Future Work - An Open Peer-to-Peer Multimedia Caching and Distribution Protocol: HyperEdge is not the end of the story; we have high hopes for the future of P2P network systems. Our goal is to redefine the supply-demand relationship of edge-based CDNs through an open distributed protocol. In the current network architecture, key roles include the Tracker, controlled by ByteDance; Intermediaries, assisting ByteDance in leasing nodes globally; Node Owners, renting out their storage and bandwidth resources to intermediaries; and End Users, consumers of on-demand multimedia content using ByteDance applications.

Looking ahead, we envision changes in network participants: eliminating intermediaries to simplify processes and enhance efficiency; Node Owners only need to install service applications compliant with the open distributed protocol to directly join the edge caching network; End Users, regardless of the application they use, can benefit from the convenience of the multimedia edge caching network. Of course, this innovative design also faces some challenges that require thorough exploration and resolution: How can we achieve precise matching of supply-demand heat in a globally distributed system to enhance overall system utilization and revenue? How can we guide more rational supply-demand relationships through differential pricing strategies for cached resources? We will continue to explore these issues, seeking practical solutions to drive network systems towards a more open, efficient, and fair direction.