Hermit: A Flow-Collaborative Transport Scheme for Multi-Source Video On-Demand Streaming

Yixuan Zhang*†, Shaorui Ren*†, Enhuan Dong*, Haiping Wang¶, Jia Zhang‡
Zili Meng§, Mingwei Xu†*, Shu Shi¶, Hebin Yu¶, Zhichen Xue¶, Yajie Peng¶, Xiaofei Pang¶, Jianping Wu*
*Tsinghua University †Quancheng Laboratory ‡Independent Researcher
§Hong Kong University of Science and Technology ¶ByteDance

ICC 2026 - IEEE International Conference on Communications | 979-8-3195-4209-0/26/$31.00 ©2026 IEEE | DOI: 10.1109/ICC59461.2026.11586829

***Abstract*—Today's fast-growing Video-on-Demand (VoD) service needs efficient content delivery to guarantee the user experience. To reduce costs, the industry has been exploring the adoption of unstable, heterogeneous, low-performance edge nodes as cost-efficient alternatives to expensive CDN servers. To compensate for the resulting degradation in user experience, Multi-source Parallel Downloading (MPD) is becoming a new VoD transport paradigm. However, existing transport optimization solutions face performance obstacles when applied to the MPD scenarios. They cannot handle the contention between MPD flows of the same download task, which is likely to occur at the shared last-hop, and lack the ability to quickly adapt to the unstable network environments brought by dynamic, heterogeneous, and low-performance edge nodes. To fill this gap, we propose Hermit, a VoD-oriented MPD transport algorithm. Hermit (1) continuously monitors the state of the flows and makes timely scheduling decisions, and (2) efficiently coordinates across the flows to mitigate self-contention at the shared last hop. As a client-driven scheme, Hermit does not require cumbersome coordination among edge nodes, nor does it increase server complexity. Through extensive experiments on real-world large-scale testbed and locally emulated network conditions, we demonstrate that Hermit can improve the consistent downloading rate by 9.2% to 21.3%.**

I. Introduction

With the explosive growth of video streaming traffic, especially in Video-on-Demand (VoD) services [1], providers are exploring cost-effective delivery architectures beyond conventional CDNs. Multi-Source Parallel Downloading (MPD), which aggregates bandwidth from multiple edge nodes, has emerged as a promising solution to reduce delivery costs and improve download performance [2]. Fig. 1 illustrates a typical network topology of MPD, where a user establishes connections[1] with multiple edge nodes, one connection per edge node, to perform a video on-demand delivery task. MPD can utilize multiple edge nodes simultaneously to improve the overall performance of the downloading task, reducing video service providers' costs significantly by extensively using edge nodes instead of expensive CDN nodes, which has aroused the interest of video service providers [2], [3], [4], [5], [6].

However, MPD introduces new transport challenges. Multiple concurrent flows may contend at the client side, causing self-induced congestion. Existing congestion control algorithms typically allow each flow to detect congestion signals only from its own traffic, while congestion at the shared bottleneck may be promptly detected by some flows but go unnoticed by others. Moreover, the number of concurrent flows significantly affects performance, yet existing schemes often ignore this factor. These issues raise new transport requirements and render existing approaches [2], [3] insufficient.

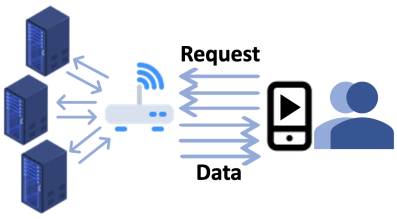


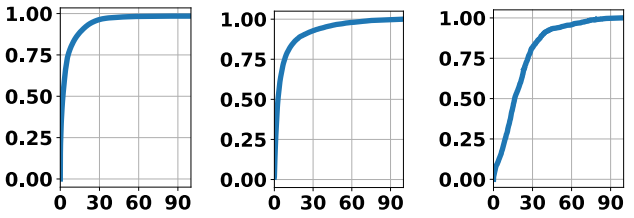
Figure 1: A typical topology of MPD.

We observe that effective MPD transport requires (1) adaptive flow-level scheduling to prevent excessive aggregation and (2) collaborative congestion control to quickly respond to shared bottlenecks. However, realizing such a design entails overcoming two critical difficulties. First, due to the heterogeneous nature of edge nodes, dynamically selecting sources based on network conditions can incur substantial overhead, especially for short-lived VoD tasks. Existing multi-source VoD systems such as Twist [2] typically assume a fixed number of flows; continuously reevaluating node selections for each task may therefore offer limited benefit. Second, it remains nontrivial to distinguish self-contention-induced congestion from ordinary single-flow congestion. Misinterpreting congestion signals (e.g., packet loss or delay) as self-contention can trigger excessive reactions, resulting in bandwidth underutilization.

To solve these problems, we propose Hermit, a fast adaptive and flow-collaborative transport solution specifically designed for VoD MPD. Hermit consists of two major components. The flow scheduler carefully selects the optimal number of edge nodes to avoid excessive bandwidth waste, ensuring efficient resource utilization (§III-B). It uses a simple yet effective two-stage probing method that does not require maintaining history link states and can converge on an appropriate number of flows. The congestion controller of Hermit shares congestion signals across flows and constructs adaptive judgments from network statistics to accurately interpret and proactively respond to self-congestion events(§III-C). Our emulations demonstrate that Hermit improves download speed by 5.6% and enhances consistent throughput by an average of 11.3% (§IV-B). Large-scale evaluations show that Hermit effectively

[1]The words connection and flow in MPD are interchangeable in this paper.

979-8-3195-4209-0/26/$31.00 ©2026 IEEE



(a) Bandwidth (MB/s) (b) Loss Rate (%) (c) Bw. Utilization(%)

Figure 2: CDF of the metrics for edge server nodes.

utilizes edge nodes, improving consistent data delivery rates by 9.2% to 21.3% (§IV-C), while maintaining low CPU and memory consumption.

In summary, our key contributions are:

- We reveal the key performance problems of the existing transport solutions under MPD scenarios (§II).
- We propose Hermit, a fast adaptive and flow collaborative transport scheme for MPD scenarios, which can consistently fully utilize the bottleneck link capacity along with fast adaptability (§III).
- We implement Hermit on top of a VoD application to be lightweight, and then evaluate Hermit by extensive experiments in emulated networks and large-scale, real-world experiments to prove its effectiveness (§IV).

II. Background and Motivation

A. Background and Related Work

Multi-Source Parallel Downloading (MPD). Extensive research has explored replacing costly CDN servers with cost-effective edge nodes [4], [5], [6], [2], [3]. These approaches enable clients to download content from multiple edge nodes simultaneously, forming the MPD paradigm. Twist, a large-scale system from ByteDance [2], [3], has been operating for over three years, serving more than 300 million users and accounting for about 35% of the video traffic.

Multipath Transport. Transport protocols such as MPTCP [7] and MPQUIC [8] also leverage multiple paths to improve access throughput and reliability for multi-interface mobile devices [9]. Before the advent of MPD, P2P-based systems [4], [10], [11], [12], [13] were widely adopted for large-scale data delivery, focusing primarily on maximizing throughput. MPD differs fundamentally by involving more heterogeneous nodes and a shared last-hop bottleneck near the client.

Congestion Control Algorithms (CCAs). Single-path CCAs, including Reno [14], Cubic [15], Vegas [16], Copa [17], GCC [18], and BBR [19], rely on various congestion signals such as loss, delay, or delivery rate. For bulk data transfers over CDNs, specialized CCAs have been proposed [20], [21], [22], [23]). Since the introduction of MPTCP and MPQUIC, multi-path CCAs [24], [25], [26], [27], [28] have been developed to extend these algorithms to multi-path environments. However, even these multipath CCAs treat flows independently and lack mechanisms for collaborative congestion handling required in MPD scenarios.

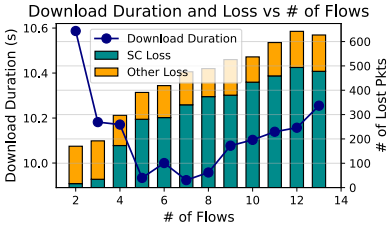


Figure 3: The video downloading duration and packet loss vs. varying numbers of flows, where SC refers to self-contention.

B. Performance Obstacles

The performance heterogeneity of edge server nodes in MPD is evident from their link capacity and packet loss. As depicted in Fig. 2(a) as the cumulative distribution function (CDF), 82.9% of the nodes exhibit a bandwidth below 10 MB/s, while 71.5% have a bandwidth lower than 5 MB/s. Conversely, the maximum 10% of the nodes can attain a bandwidth up to 10 times the minimum 10%. Considering that the average downloading speed under mobile networks can reach 43 MB/s and the bitrate for 4K video is at about 18 Mbps [1], downloading on a single edge node can potentially result in significant performance degradation due to the physical capacity gap. Fig. 2(b) illustrates the CDF of packet loss rates observed from edge nodes with more than 20.6% exceeding 10% loss rate. Packet loss can lead to retransmissions, Head-of-Line (HoL) blocking, and stalls in VoD playback. This inherent poor performance of edge nodes also poses a risk of performance degradation. Compared to traditional scenarios involving multiple flows, MPD presents a novel set of performance challenges. Specifically, VoD in MPD encounters two key performance obstacles due to its unique characteristics: (1) strong node heterogeneity and (2) a topology in which multiple flows aggregate and share bottleneck near the client.

Obstacle 1: Diminishing returns with increasing flow heterogeneity. When the number of edge nodes exceeds a certain threshold, adding more nodes no longer improves delivery efficiency and can even degrade performance. To validate this phenomenon, we conduct experiments using an MPD application in a Mininet-based topology (Fig. 3), where the number of nodes varies from 2 to 13 with random RTTs (5–50 ms) and small random loss rates (0–1%). As shown in the figure, increasing the number of nodes initially accelerates downloads, but beyond five nodes, the benefit quickly diminishes, and performance declines when more than ten nodes are used. This degradation mainly stems from two factors. First, the growing disparity among heterogeneous nodes amplifies tail-performing nodes can delay decoding and disrupt the overall VoD experience. Second, excessive flows can reduce consistent throughput (the in-order data rate perceived by users), even with high aggregate bandwidth. In our experiments, consistent throughput dropped up to 20–30% below total throughput as node counts increased. Moreover, maintaining many active flows adds overhead to both clients and edge nodes.

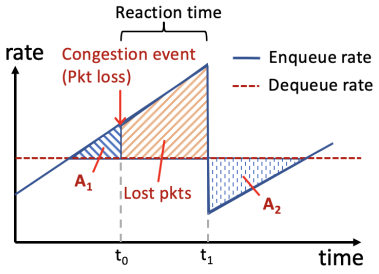


Figure 4: Rate change during a congestion event in the case of single flow.

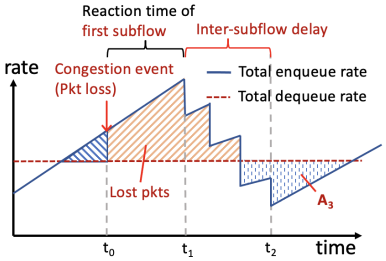


Figure 5: Rate change during a congestion event in the case of multiple flows.

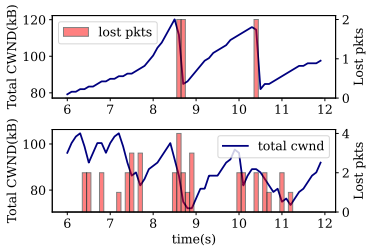


Figure 6: The total CWND trendline of single flow (the upper figure) and multiple flows (the lower figure) with Cubic.

Obstacle 2: Self-contention at the shared last hop. As shown in Fig. 3, increasing the number of flows amplifies packet loss, with self-contention at the shared last hop becoming the dominant cause. Even under optimal configurations, such loss exceeds random link loss, resulting in redundant retransmissions and reduced efficiency. The primary reason lies in asynchronous congestion detection among subflows converging on the same wireless bottleneck. Our analysis in Figs. 4 and 5 illustrate that, unlike single-flow CCAs reacting promptly at t_1, multi-flow scenarios introduce inter-subflow delay ($t_2 - t_1$) before all flows adjust, leading to prolonged contention and slower recovery. Consequently, MPD scenarios exhibit higher packet loss and lower throughput stability despite potentially higher aggregate throughput. Empirical validation using Mininet [29] and Iperf [30] (Fig. 6) confirms frequent loss and oscillation patterns consistent with this analysis.

III. Methodology

In this section, we first formalize the challenges and provide an overview of the corresponding components of Hermit design. Then, we give a detailed introduction of each component.

A. Design Overview

Designing and implementing a transport solution for commercial VoD MPD scenarios presents two design challenges: **(1) Choosing appropriate edge nodes and scheduling in a highly dynamic, heterogeneous environment.** In establishing connections, our approach should select the appropriate nodes from a pool of candidates with unknown performance characteristics. This introduces 3 key issues: determining the required number of nodes, selecting suitable nodes based on link statistics, and scheduling flows in a timely manner based on the monitored network states. **(2) Accurately capturing and handling congestion events at the shared last-hop.** Even with the correct number of nodes, self-contention can still occur frequently at the last hop. If lost packets are not retrieved before the user buffer is depleted, video stalls will occur. However, self-contention congestion signals can be confused with other events like random fluctuations or congestion on a single link. Our method should accurately capture and interpret congestion signals, share them across flows, and differentiate them from other causes of congestion.

As shown in Fig. 7, the workflow of Hermit operates as

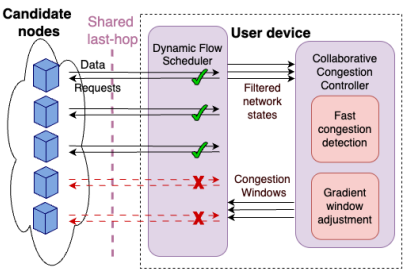


Figure 7: The overview of Hermit design.

follows. In the context of MPD-based VoD scenarios, the client initiates requests to edge nodes, which respond with the corresponding data payloads. The client collects the data pieces, requests retransmissions for lost packets, and assembles the complete video frame. To control the data arrival rate, the client adjusts the size of its request congestion window (CWND). Our methodology includes two novel components:

1) **A state-based dynamic flow scheduler** that performs coarse-grained management, continuously monitoring and managing the state of each flow (§III-B).
2) **A flow-collaborative congestion controller** that infers congestion and makes timely joint window adjustment decisions by coordinating across the flows under the same download task (§III-C).

B. Scheduling the Flows

The flow scheduler is responsible for maintaining link statistics and managing the number of active flows. To ensure a lightweight yet effective scheduling mechanism, we propose storing a simple piece of data instead of the history of node states: the final number of flows N_0 from the history VoD task. The flow scheduler adopts a self-converging two-stage strategy.

Stage 1: Connection establishment. Before downloading, the VoD client retrieves a list of $2N_0$ candidate edge nodes containing the requested video. The initial choice of edge node quantity can be coarse-grained, because within a range (such as 5 to 8 flows in Fig. 3) user experience remains relatively stable. A slightly over-provisioned pool ensures node availability and tolerance to poor links. The scheduler initializes $N_0 + 1$ active flows, allowing one spare flow that can be deactivated if performance degradation occurs.

Stage 2: Connection reduction. Once connections are

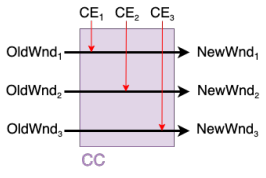


Figure 8: Existing CCAs only make decisions on the window of each flow separately based on the congestion events of the single flow.

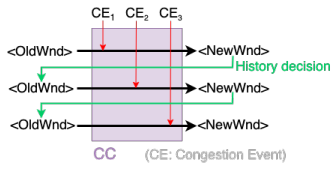


Figure 9: Hermit makes joint decisions on the windows of all the flows based on both congestion events and history decisions.

established, the scheduler monitors CWND, RTT, and loss to refine the flow set. At this stage, the scheduler prioritizes reducing the number of flows. It progressively disables one flow at a time and observes the impact on performance. If throughput remains stable, the reduction is kept; otherwise, another candidate node is tested. To avoid oscillation, a flow is deactivated only when its loss rate exceeds 20% and dominates all other links.

C. Collaborative Congestion Control

The congestion controller dynamically adjusts each flow's CWND based on inferred congestion states while maintaining scalability. Unlike existing CCAs that act in isolation, Hermit coordinates across multiple flows by sharing congestion information, acting as an upper-layer policy compatible with existing CCAs. It consists of two modules: **collaborative congestion detection** and **joint CWND adjustment**.

Collaborative congestion detection. When self-contention occurs at the shared last hop, the flow that first detects congestion propagates this signal to others, enabling faster global awareness. To distinguish such self-contention from random losses or transient fluctuations, Hermit smooths RTT variations using an exponentially weighted moving average (EWMA) as $\Delta RTT_i'[t] = \alpha \Delta RTT_i[t] + (1-\alpha)\Delta RTT_i'[t-1]$. Each flow reports both the smoothed RTT gradient and packet loss $L_i[t]$, which are aggregated in $Y[t] = \sum(w_i^R \Delta RTT_i'[t] + w_i^L L_i[t])$ to infer the shared congestion state. w_i^R and w_i^L, the adaptive weights for RTT and loss, are iteratively updated based on how well the individual flow signals align with the inferred congestion:

$$w_i^R[t+1] = w_i^R[t](1+\eta\delta_i^R[t]),$$
$$w_i^L[t+1] = w_i^L[t](1+\eta\delta_i^L[t])$$

where η is the learning rate and δ indicates the consistency with the inferred state. The aggregated congestion signal $Y[t]$ is compared with an adaptive threshold to get

$$C[t] = \begin{cases} 1, & Y[t] > T[t] \\ 0, & \text{otherwise} \end{cases}, \quad T[t] = \beta \cdot EWMA(Y[t])$$

where $T[t]$ evolves dynamically to reflect network changes without requiring labeled data or complex models.

Joint CWND adjustment. Upon detecting self-contention ($C[t]=1$), Hermit jointly adjusts the CWNDs of all affected flows, in contrast to traditional CCAs that act independently. It further refines these adjustments via regression feedback, improving stability and responsiveness over time. To avoid excessive oscillations, the total sending window is regulated using a dynamic per-flow coefficient:$w_i^B[t+1] = w_i^B[t](1+\eta\delta_i^B[t])$, where $\delta_i^B[t]$ penalizes flows causing HoL blocking or slow loss recovery (e.g., when receiver buffer depth exceeds one BDP). This cooperative mechanism allows flows to converge toward stable utilization under shared last-hop contention.

IV. Evaluation

In this section we evaluate Hermit to answer the following questions:

1) *Can Hermit achieve consistently high performance and converge fast?* We conducted emulations for various network scenarios, and the results show that Hermit can improve the downloading speed by 5.6% and enhance the consistent throughput by 11.3% on average (§IV-B).
2) *Can Hermit deliver high performance in real-world, large-scale commercial services?* We performed both mobile unit tests (§IV-C1) and large-scale, real-world experiments (§IV-C2) to evaluate the performance of Hermit in practical deployment scenarios. The results demonstrate that Hermit can improve the consistent data delivery rate by 9.2% to 21.3%.
3) *Is Hermit lightweight enough to run on mobile devices?* We evaluated the overhead of Hermit by analyzing its CPU and memory consumption on a mobile phone. Our findings indicate that it is a lightweight solution with good scalability, making it suitable for deployment on mobile devices (§IV-C3).

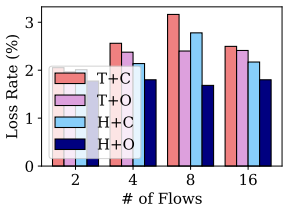
A. Experiment Setup

To validate the practical applicability of Hermit, we integrate the algorithm as a patch into a VoD application implemented on the Android 13 platform and built with Gradle 8.0. As our baseline, we adopt the joint flow control mechanism from Twist [2], a multi-source transport framework deployed at scale in commercial systems. We assess Hermit's impact on real-world network performance including consistent downloading rate and packet loss. Hermit was successfully deployed and tested in real-world conditions, including both home WiFi scenarios and across a wide range of mobile devices with varying performance capabilities. For emulation, we leverage Mininet [29], a widely-used network emulator, to explore a wide array of topologies and network conditions.

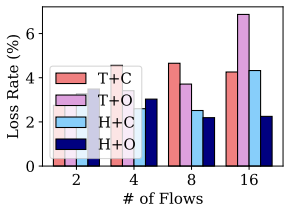
B. Emulation

1) Controlled Environment: We first emulate a homogeneous network where each node has 50 ms delay, 10 Mbps bandwidth, and no random loss (Fig. 10). The shared last-hop link provides 20 Mbps bandwidth with 1 % random loss. We vary the number of flows (2, 4, 8, 16) to compare Hermit with Twist under two CCAs: Cubic (single-flow) and OLIA (multi-flow).

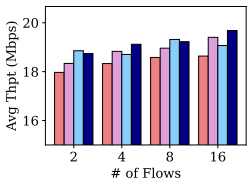
- Loss ratio (Fig. 10(a)). Integrating Hermit with OLIA



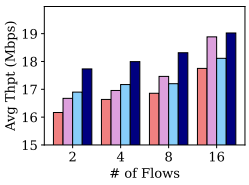
(a) Loss rate (%) with homogeneous flows.



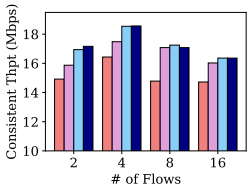
(b) Loss rate (%) with heterogeneous flows.



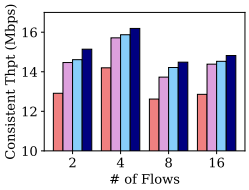
(c) Average Thpt (Mbps) with homogeneous flows.



(d) Average Thpt (Mbps) with heterogeneous flows.



(e) Consistent Thpt (Mbps) with homogeneous flows.



(f) Consistent Thpt (Mbps) with heterogeneous flows.

Figure 10: Performance of Hermit compared with other algorithms. 'T+C' denotes Twist flow control with Cubic, 'T+O' denotes it with OLIA, and 'H+C' and 'H+O' denote Hermit with Cubic and OLIA respectively as the underlying CCA.

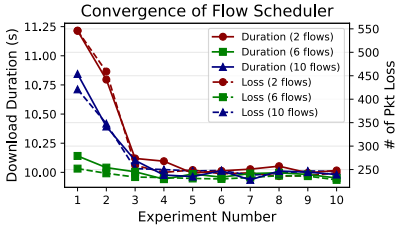
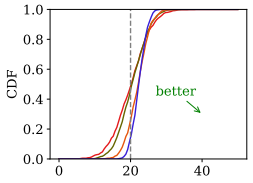


Figure 11: The converging trend of the flow scheduler in a series of VoD tasks.

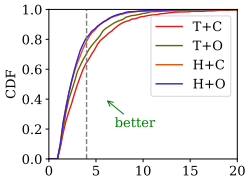
achieves the lowest loss (2.01 %), compared with Cubic's 2.78%, showing Hermit 's ability to mitigate congestion-induced loss.

- Average throughput (Fig. 10(c)). Hermit improves throughput by 4.2% with Cubic and 3.7% with OLIA, demonstrating efficient bandwidth utilization.
- Consistent throughput (Fig. 10(e)). Hermit enhances throughput stability, reducing retransmission ratios by 9.5% (Cubic) and 3.7% (OLIA), confirming its fast adaptability to congestion dynamics.

2) Heterogeneous Flows: Following the experimental topology in §IV-B1, we evaluate heterogeneous link scenarios with delays of 30-100 ms, bandwidths of 5-15 Mbps, and

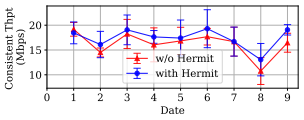


(a) Average downloading rate(Mbps).

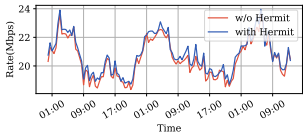


(b) Timeout rate(%).

Figure 12: Performance of Hermit in real-world mobile test.



(a) Consistent throughput.



(b) Downloading rate.

Figure 13: Performance of Hermit in large-scale testbed.

random single-link losses of 0-2%. Results are shown in Fig. 10(b), 10(d), and 10(f).

- **Impact of heterogeneity.** Network heterogeneity leads to throughput degradation, average throughput drops by 11.0% and consistent throughput by 13.5%, mainly due to increased vulnerability to packet loss, which worsens with more concurrent flows.
- **Benefits of Hermit.** Applying Hermit improves both throughput and loss rate across Cubic and OLIA. It increases average throughput by 5.6% and consistent throughput by 11.3%, while reducing OLIA's packet loss by up to 63%. The gains stem from proactive mitigation of HoL blocking among heterogeneous flows.

C. Real-world Evaluation

1) Mobile Test: We evaluated Hermit in a real-world WiFi environment using a VoD application that simultaneously downloaded a video from 10 edge nodes (Fig. 12). Each experiment was repeated 300 times to construct CDFs of download rate and timeout rate.

- Overall performance. Among all schemes, Hermit +OLIA achieves the highest download rate, demonstrating its effectiveness under realistic mobile conditions.
- Tail performance. In the timeout CDF, Hermit +OLIA exceeds a 4% loss rate in only 20% of runs, compared with about 40% for Cubic. For the download rate CDF, less than 10% of Hermit +OLIA cases fall below 20 Mbps, whereas Cubic and OLIA alone exhibit nearly half of trials below this threshold. These results confirm that Hermit substantially improves both median and tail performance in mobile MPD scenarios.

2) Practice in Large-scale Testbed: We further tested Hermit on a large-scale mobile network with 1,000 Android devices using both cellular and WiFi connections. Devices running Cubic+Hermit were compared with baselines over multiple days of video downloads. As shown in Fig. 13(a) and Fig. 13, Hermit consistently maintains higher average and

Table I: CPU and memory overhead of Hermit enhanced application on a mobile phone.

# of nodes	CPU (%) w/ H(Hermit)	CPU (%) w/o H	Memory (MB) w/ H	Memory (MB) w/o H
10	11.5	11.2	20.4	18.6
20	11.7	11.4	26.2	22.8

consistent throughput over time, highlighting its scalability and adaptability to varying network conditions.

3) Resource Consumption: Table I depicts CPU and memory usage of the Hermit-enhanced VoD application. The CPU utilization increases only slightly (from 11.2% to 11.5%), and memory consumption remains modest, rising from 18.6 MB to 20.4 MB with ten nodes, and from 22.8 MB to 26.2 MB with twenty nodes. These results confirm that Hermit introduces negligible computational overhead and is suitable for mobile deployment at scale.

V. Conclusion

As the industry shifts toward cost-efficient yet unstable edge nodes as alternatives to CDN servers, MPD has emerged as a promising VoD transport paradigm. We presented Hermit, a fast adaptive and flow-collaborative transport scheme that mitigates contention among concurrent MPD flows and adapts effectively to heterogeneous, dynamic edge environments. Our evaluations demonstrate that Hermit significantly improves download efficiency and throughput stability. The proposed flow-collaborative mechanism also holds potential for broader applications, including live streaming, distributed cloud data transfers, and IoT data aggregation.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant 62221003 and Grant 62502471), the Shandong Provincial Natural Science Foundation, China (Grant ZR2024LZH011), the Research Project of Provincial Laboratory of Shandong, China (Grant SYS202201), and the Guangdong NSF Project (Grant 2025A1515010460). Mingwei Xu, Enhuan Dong, and Haiping Wang are the corresponding authors of the paper.

References

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," Cisco, Tech. Rep., 2020.

[2] H. Wang, R. Zhang, C. Li, Z. Xue, Y. Peng, X. Pang, Y. Zhang, S. Ren, and S. Shi, "Twist: A multi-site transmission solution for on-demand video streaming," *Proceedings of the ACM on Networking, CoNEXT*, pp. 1–19, June 2024.

[3] R.-X. Zhang, H. Wang, S. Shi, X. Pang, Y. Peng, , Z. Xue, and J. Liu, "Enhancing resource management of the world's largest pcdn system for on-demand video streaming," in *USENIX ATC*, 2024.

[4] G. Zhang, W. Liu, X. Hei, and W. Cheng, "Unreeling xunlei kankan: Understanding hybrid cdn-p2p video-on-demand streaming," *IEEE Transactions on Multimedia*, vol. 17, no. 2, pp. 229–242, 2014.

[5] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Computer Networks*, vol. 116, pp. 79–95, 2017.

[6] "Grand challenge," in *Proceedings of the 14th ACM Multimedia Systems Conference (MMSys '23)*. ACM, 2023.

[7] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "Rfc 8684: Tcp extensions for multipath operation with multiple addresses," 2020.

[8] Y. L. et al., "Multipath extension for quic," IETF, Internet-Draft draft-ietf-quic-multipath-07, work in Progress, Apr 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/07/

[9] O. Bonaventure, C. Paasch, and G. Detal, "RFC 8041: Use Cases and Operational Experience with Multipath TCP," 2017.

[10] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec, "Peer-assisted content distribution in akamai netsession," in *ACM IMC*, 2013, pp. 31–42.

[11] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Livesky: Enhancing cdn with p2p," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 6, no. 3, pp. 1–19, 2010.

[12] R. Farahani, A. Bentaleb, E. Çetinkaya, C. Timmerer, R. Zimmermann, and H. Hellwagner, "Hybrid p2p-cdn architecture for live video streaming: An online learning approach," in *IEEE GLOBECOM*, 2022, pp. 1911–1917.

[13] H. Zhang, A. Zhou, Y. Hu, C. Li, G. Wang, X. Zhang, H. Ma, L. Wu, A. Chen, and C. Wu, "Loki: improving long tail performance of learning-based real-time video adaptation by fusing rule-based models," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 775–788.

[14] M. Allman, V. Paxson, and E. Blanton, "RFC 5681: TCP Congestion Control," 2009.

[15] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[16] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *Proceedings of the conference on Communications architectures, protocols and applications*, 1994, pp. 24–35.

[17] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *USENIX NSDI*, 2018, pp. 329–342.

[18] S. Holmer, H. Alvestrand, and H. Lundin, "A google congestion control algorithm for real-time communication on the world wide web," *IETF Draft, June*, 2015.

[19] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[20] X. Sun and N. Ansari, "Improving bandwidth efficiency and fairness in cloud computing," in *IEEE GLOBECOM*, 2013, pp. 2313–2318.

[21] J. Liu, Q. Yang, and G. Simon, "Congestion avoidance and load balancing in content placement and request redirection for mobile cdn," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 851–863, 2018.

[22] D. Yuan, W. Zhang, Y. Qiu, H. Huang, M. Yang, P. Chen, K. Xiao, H. Yan, Y. He, and Y. Zhang, "Context-aware cross-layer congestion control for large-scale live streaming," *IEEE/ACM Transactions on Networking*, 2024.

[23] X. Sun, Z. Wang, Y. Wu, H. Che, and H. Jiang, "A price-aware congestion control protocol for cloud services," *Journal of Cloud Computing*, vol. 10, pp. 1–15, 2021.

[24] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "Mptcp is not pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Transactions On Networking*, vol. 21, no. 5, pp. 1651–1665, 2013.

[25] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath tcp: Analysis, design, and implementation," *IEEE/ACM Transactions on networking*, vol. 24, no. 1, pp. 596–609, 2014.

[26] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," Tech. Rep., 2011.

[27] T. Gilad, N. Rozen-Schiff, P. B. Godfrey, C. Raiciu, and M. Schapira, "Mpcc: Online learning multipath transport," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 121–135.

[28] S. R. Pokhrel and M. Mandjes, "Improving multipath tcp performance over wifi and cellular networks: An analytical approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2562–2576, 2018.

[29] "Mininet: An instant virtual network on your laptop (or other pc)," http://mininet.org, 2014.

[30] "iperf 3.10.1 - the ultimate speed test tool for tcp, udp and sctp," https://iperf.fr 2019.