Multi-CDN as a Collective Service: Towards Hot Start in Congestion Control at Scale

Tong Li
Renmin University of China
Beijing, China
State Key Laboratory of Internet Architecture (Tsinghua University)
Beijing, China
tong.li@ruc.edu.cn

Jiuxiang Zhu
Renmin University of China
Beijing, China
Tencent Technologies
Beijing, China
jiuxiangzhu@ruc.edu.cn

Bo Wu*
Tencent Technologies
Beijing, China
wub14@tsinghua.org.cn

Cheng Luo
Tencent Technologies
Beijing, China
lancelotluo@tencent.com

Haoyi Fang
Renmin University of China
Beijing, China
2021201680@ruc.edu.cn

Xiaofei Yu
Tencent Technologies
Beijing, China
xiaofeiiiyu@tencent.com

Ke Xu
Tsinghua University
Beijing, China
xuke@tsinghua.edu.cn

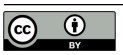
Abstract

Modern CDN vendors typically provide multi-server services for content deliveries, in which different requests from the same client are often scheduled to diverse CDN nodes. In this case, the requested data (e.g., short video chunks) always suffer from *cold start* during its transmission, especially for the *first flow* of established connections, due to the unknown client-side network quality. Existing schemes focus on single-server startup parameter optimization, whose referred metrics appear to be extremely limited. To address this issue, we propose *hStart*, a novel *collective service* paradigm that enables hot-start congestion control by efficient collaboration between CDN nodes. hStart synchronizes probed network quality information (e.g., bandwidth, latency) across CDN nodes to initialize congestion control from historical data, avoiding probing convergence. It uses hierarchical state synchronization to limit overhead, and adaptive prediction with a time-decay trust model to handle dynamics. We implement and deploy the *hStart* system for the QUIC protocol. Extensive evaluations in both controlled testbeds and large-scale production CDN environments demonstrate that *hStart* effectively eliminates cold start penalties, boosting 46.47% throughput gain and 21.43% rebuffering reduction.

CCS Concepts

• **Networks** → **Transport protocols**; *Network performance evaluation*; *Cloud computing*.

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License.
APNet 2026, Singapore, Singapore
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2664-4/26/08
https://doi.org/10.1145/3820441.3820445

Keywords

Congestion control, Hot start, Content delivery network, QUIC, Multi-CDN

ACM Reference Format:
Tong Li, Jiuxiang Zhu, Bo Wu, Cheng Luo, Haoyi Fang, Xiaofei Yu, and Ke Xu. 2026. Multi-CDN as a Collective Service: Towards Hot Start in Congestion Control at Scale. In *The 10th Asia-Pacific Workshop on Networking (APNet 2026), August 06–07, 2026, Singapore, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3820441.3820445

1 Introduction

With the explosive growth of mobile data traffic, Content Delivery Networks (CDNs) have become the cornerstone of the modern internet experience. To reduce access latency, CDN providers geographically deploy edge nodes, caching web, video, and application content on servers closest to users to enhance Quality of Experience (QoE).

In typical scenarios, despite being physically stationary, mobile users exhibit highly discrete network interaction behaviors [1, 2]. As illustrated in Figure 1, frequent switching between diverse applications and dynamic load-balancing strategies often direct subsequent requests to different CDN nodes or connections, which renders users "strangers" to new nodes. Consequently, historical Network Quality Information (NQI), such as Round-Trip Time (RTT) and Bottleneck Bandwidth Estimate (BWE), cannot be shared across nodes or connections, causing a stateless interaction pattern that triggers the transport layer cold start bottleneck.

Taking a large-scale short video application as a case study, our real-world measurements reveal two critical facts (§3). (1) *First flows dominate and perform poorly.* In mobile services, 38.8% of requests are the "first flow" (i.e., a new connection established with the current server). Compared to subsequent flows, the throughput of the first flows is 25% lower on average, with significantly higher

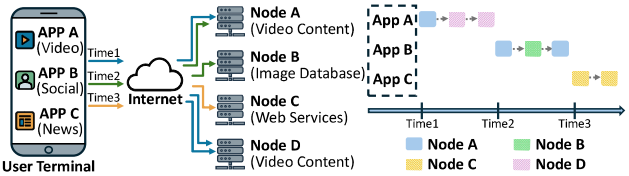


Figure 1: An example of cross-node access in modern CDN systems. A user's requests are frequently routed to different physical CDN edge nodes, either when switching between different applications or when requesting resources within the same application over time.

Flow Completion Time (FCT). (2) *First flows impact QoE.* In video streaming services, startup stalls are primarily determined by the transmission performance of the first flow. The slow growth of the congestion window during the cold start phase directly leads to insufficient video buffer filling.

The core conflict lies in the fact that servers cannot perceive the user's network conditions in advance. Transport protocols (such as TCP [3] or QUIC [4]) must adopt a conservative *slow-start* strategy, probing exponentially from a minimal Initial Window (IW). This creates a dilemma: a default IW (e.g., IW=10) [5] wastes several RTTs before fully utilizing a high-bandwidth network, yet blindly increasing the IW natively causes severe packet loss and queueing delay in congested networks, paradoxically degrading performance.

Given that Node A has already probed the network states during the interaction with the user, Node B can bypass the slow-start phase by acquiring this information prior to serving the user. This paper proposes *hStart*, a hot start scheme based on collaborative CDNs. The core idea is to break the information silos between CDN nodes by establishing a lightweight state-sharing plane. To achieve this, we must overcome two fundamental challenges by designing and implementing the following mechanisms in the prototype system.

Challenge 1: Cost-Benefit Trade-off. CDNs contain tens of thousands of nodes and massive users. Full-mesh synchronization incurs prohibitive bandwidth and computational overhead, potentially triggering control-plane congestion. Thus, the primary system challenge lies in achieving precise synchronization of high-value information without introducing significant overhead. To address this, we propose *Hierarchical Adaptive State Synchronization (HASS).* We abandon the full-mesh model and propose a 3D control strategy based on Space, Object, and Time. Synchronization is triggered only in high-value scenarios, reducing ineffective traffic.

Challenge 2: Information Freshness and Trust. Highly dynamic network environments, driven by user mobility and path variations, often invalidate historical priors. Blindly applying outdated information risks triggering congestion collapse. Thus, designing adaptive algorithms to safely leverage such data is critical for system robustness. Then, we propose *Adaptive Prediction and Feedback Adjustment (APFA),* a trust model based on time decay. Historical information is treated as "advice" rather than "orders". We use micro-probing at the start of connections for rapid validation and dynamically update user reputation scores based on prediction errors, forming a Predict-Validate-Adjust closed loop.

We have deployed hStart on controlled testbeds and in one of the world's largest commercial CDN systems with millions of short video streams. We find that: (1) hStart precisely synchronizes NQI with minimal overhead, reducing invalid synchronization traffic through HASS; (2) Compared to baseline, hStart effectively eliminates the cold start penalty for first flows, resulting in a 46.47% enhancement in throughput and 21.43% reduction in video rebuffering rates; (3) Under highly dynamic network conditions, the APFA scheme robustly balances aggressive initial acceleration and congestion safety, ensuring stable performance and maximizing overall network efficiency without causing congestion collapse.

Contributions. (1) hStart is the first collaborative congestion control system that enables hot-start across CDN nodes via cross-server state synchronization at scale. (2) We introduce a lightweight hierarchical mechanism to minimize synchronization overhead and a time-decay trust model to safely leverage historical network information under dynamics. (3) We validate hStart in a commercial CDN deployment, demonstrating its practicality and effectiveness in improving throughput and user QoE.

2 Related Work

Slow Start Optimization. Traditional slow-start mechanisms often suffer from slow convergence and low bandwidth utilization, particularly in short-flow or high Bandwidth-Delay Product (BDP) scenarios. Existing optimizations generally mitigate this by either explicitly modifying the probing phase (e.g., eliminating slow-start entirely or accelerating the window growth factor [6–9]) or tuning the IW using historical states and reinforcement learning [10, 11]. While effective for individual sessions, these methods fundamentally operate as isolated, local optimizations (single-connection or intra-machine). Because they lack cross-connection collaborative awareness, they cannot leverage shared bottleneck information across physically distributed edge servers. Unlike these isolated approaches, *hStart* introduces a collective service paradigm that aggregates and shares NQI across isolated edge nodes, enabling truly global, cross-server hot start without relying on a single machine's limited history.

Coordination and Distributed Control. To mitigate blind competition among concurrent flows, prior work has proposed both collaborative congestion-control schemes and distributed transport architectures. These approaches explicitly coordinate parallel connections on shared paths or within data centers to behave as a single aggregate flow [12–15]. Other architectures advance coordination by decoupling the control plane from the data forwarding plane, deploying control logic at the edge, or relying on client-side state retention [16–19]. Recent systems further extend this idea to multi-server service execution, using stream-level dispatch or request-granularity connection offloading to emulate a single client-facing connection across multiple backend servers [20, 21]. Despite these advances, existing collaborative designs mainly focus on intra-domain fairness, control placement, or L7 request dispatch, and typically depend on centralized coordination, client-side state retention, connection/stream migration, or complex deployment assumptions. This limits their applicability over heterogeneous cloud-edge-terminal paths characterized by frequent server switching. In contrast, hStart keeps normal client-server transmission

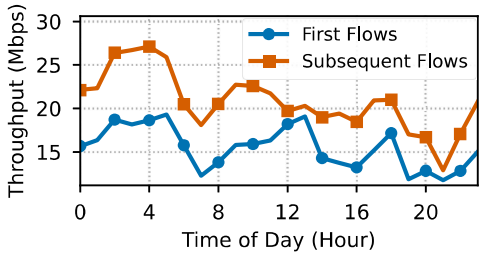
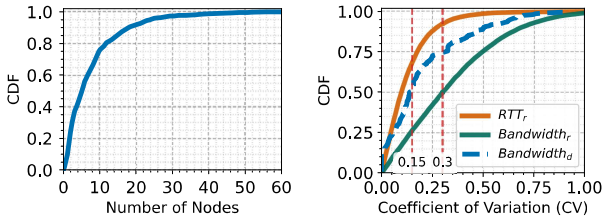


Figure 2: Download throughput over 24 hours: first flows vs. subsequent flows in QUIC connections.



(a) CDF of Accessed Nodes per User. (b) CDF of Network Metric CV.

Figure 3: CDF analysis of user access behaviors.

while sharing NQI across CDN nodes to initialize congestion control states. By operating seamlessly in dynamic network conditions without requiring client-side modifications or centralized bottlenecks, hStart natively supports the realities of modern multi-CDN environments.

3 Measurement Study

To quantify the impact of the transport layer cold start on modern mobile applications, we conducted a two-week measurement analysis on the CDN logs of a popular video app.

Observation 1: First Flows Perform Suboptimally. Figure 2 illustrates the throughput monitored over 24 hours. The throughput of the first flows is consistently and significantly lower than that of subsequent flows, achieving only 74% of the average transmission rate of subsequent flows established with the same CDN node.

Observation 2: First Flows Dominate Volume and QoE Impact. Across all requests, 38.8% of video segments are transmitted through the connection's first flow, with 45% of connections transmitting only a single flow. Crucially, users experiencing poor first flow performance face a 0.31% stall rate, nearly triple the 0.11% rate of users with good initial performance, well above the overall average of 0.13%.

Observation 3: Frequent Cross-node Scheduling Means Cold Everywhere. Even during a single long video session or when concurrently requesting diverse resources, CDN load balancing frequently redirects requests across different servers. Figure 3(a) indicates that roughly 75% of users access two or more geographically distinct CDN nodes. Each new node access forces a congestion window reset, continuously degrading performance without state inheritance.

Observation 4: Consistent Cross-node Network Quality. We examine this assumption using both controlled dial-test measurements and production CDN logs. In the dial-test data set, we deployed experimental servers on CDN nodes in three Chinese provinces and used nationwide monitoring agents as clients. These agents accessed nodes over wired networks. Figure 3(b) shows that the Coefficient of Variation (CV) [22] of bandwidth (Bandwidth_d) across nodes is below 0.3 for 75% of agents. We also analyze the production logs from CDN nodes in a Chinese city, where users access the service through mobile or wired networks. Production data corroborate this observation: RTT (RTT_r) remains stable and roughly 50% of users exhibit bandwidth variation (Bandwidth_r) below 0.3 across nodes. These results suggest that many users traverse similar bottleneck links regardless of the scheduled target node. Furthermore, a broader validation across CDN nodes in different provinces is left for future work.

Summary. A mechanism is highly required that can understand the client-side network quality before data transmission, while avoiding the negative impact of potential prolonged probing on the first flow performance. This motivates our design of hStart.

4 System Design

4.1 Overview

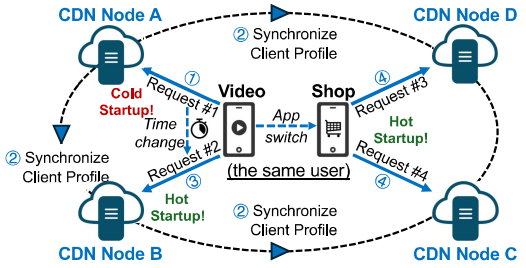
To overcome the transport layer cold start triggered by frequent cross-node access, hStart breaks the information silos among edge nodes by sharing NQI across the CDN infrastructure. As illustrated in Figure 4, when a user dynamically routes to new nodes, hStart proactively migrates their historical connection state (e.g., bandwidth and RTT) to enable a hot start, smoothly bypassing the conservative default slow-start phase. To practically achieve this at the scale of modern CDNs without overwhelming the control plane or triggering congestion via stale data, hStart proposes two core mechanisms: HASS at the synchronization layer bounds communication overhead by selectively sharing high-value states, while APFA at the transport layer ensures safety by treating historical conditions as confidence-weighted predictions rather than absolute ground truth.

4.2 HASS

To achieve low-overhead state sharing across massive nodes, HASS avoids omniscient consistency. Instead, it balances cost and benefit through pruning strategies in *Space* (geographic grouping), *Object* (hot-spot traffic awareness), and *Time* (dynamic benefit assessment) dimensions.

Spatial Dimension: Geographic Grouping. Traditional inter-node communication often uses a Full-Mesh topology with $O(N^2)$ complexity. HASS divides CDN nodes into logical sync groups $G = \{N_1, N_2, \ldots, N_k\}$ based on physical location and network topology (AS/IDC info).

Object Dimension: Traffic Hotspot Awareness. According to the Pareto Principle (80/20 rule), a minority of hotspot domains account for the majority of requests. Synchronizing states for long-tail traffic is inefficient. HASS introduces a *Traffic-Aware Filter*. Each node maintains an online hotspot domain list D_{hot} using the space-saving algorithm. User network state is added to the synchronization queue only if one of the following conditions is met: (1) User is accessing $d \in D_{hot}$. (2) Traffic is identified as high-priority business flow (e.g., live streaming signaling).



Step ①: At 8:00, a client sends request #1 to CDN node A with cold-startup response;
Step ②: CDN node A synchronizes this client's profile to other CDN nodes (B, C and D);
Step ③: At 8:01, same client sends new request #2 to CDN node B with hot-startup response;
Step ④: Upon application switching, the client sends new requests (#3 and #4) to other CDN Nodes (C and D), with hot-startup responses.

Figure 4: An example of the hStart workflow.

Temporal Dimension: Benefit-Driven Triggering. Even within a group, not every state update requires immediate broadcasting. A single synchronization is triggered only when the difference (Standardized Euclidean Distance) between the new sample value and the stored historical value exceeds a set threshold.

4.3 APFA

Directly applying synchronized historical state ($State_{history}$) is risky because edge network quality dynamically fluctuates due to wireless handovers, bursty background traffic, router queueing, or server load changes. Consequently, APFA avoids the naive assumption of state stationarity ($State_{current} = State_{history}$). Instead, it ensures robust transport through a three-stage mechanism: *Predictive Initialization*, *Safe Validation*, and *Closed-Loop Calibration*.

Time-Decayed Predictive Initialization. We use a Trust Score $T \in [0, 1]$ (long-term stability) and information freshness Δt to control our confidence $\alpha(t)$ in historical states. The initial congestion window $cwnd_{\text{init}}$ is computed as a weighted aggregation:

$$\alpha(t) = T \cdot e^{-\lambda \Delta t} \tag{1}$$

$$cwnd_{\text{init}} = \alpha(t) \cdot cwnd_{\text{history}} + (1 - \alpha(t)) \cdot cwnd_{\text{default}} \tag{2}$$

where λ is a predefined decay constant. This formulation adapts the startup strategy: highly fresh and stable histories ($\alpha \to 1$) trigger aggressive hot start, while stale or volatile data ($\alpha \to 0$) seamlessly downgrade to a safe standard slow-start using $cwnd_{\text{default}}$.

Paced Validation. Even if the calculated $cwnd_{init}$ is large, we do not allow direct bursting to avoid overwhelming bottleneck links. APFA sets the first RTT of the connection as the validation phase: (1) *Pacing*: Send packets smoothly using a pacing rate, rather than dumping the window at once. (2) *Anomaly Detection*: Monitor ACKs for the first batch of packets in real-time. If $RTT_{sample} > \gamma \cdot min_RTT_{history}$ or any packet loss occurs, it is immediately judged as Prediction Failure. (3) *Fast Fallback*: Upon failure, reset *cwnd* to default immediately. If validation passes, exit validation and enter congestion avoidance directly.

Closed-Loop Feedback and Calibration. When a connection ends, the system establishes a closed-loop learning mechanism by quantifying the relative prediction error E:

$$E = \frac{|state_{\text{sync}} - state_{\text{final}}|}{state_{\text{final}}} \tag{3}$$

where $state_{\text{sync}}$ is the historical state initially shared and $state_{\text{final}}$ is the true converged state. This error iteratively updates the Trust T:

$$T_{\text{new}} = (1 - \eta) \cdot T_{\text{old}} + \eta \cdot (1 - E) \tag{4}$$

where η is the learning rate. This calibration ensures accurate predictions increase system trust in the user, while major network shifts cause T to drop safely, forcing conservative behavior in future requests. To safely sync the updated state back to the shared pool and avoid transient noise, we use an adaptive EWMA (Exponential Weighted Moving Average) :

$$state_{\text{stored}} = \alpha_{\text{update}} \cdot state_{\text{old}} + (1 - \alpha_{\text{update}}) \cdot state_{\text{sample}} \tag{5}$$

Here, $state_{\text{old}}$ is the stored metric and $state_{\text{sample}}$ is the newly probed state. To tie the update pace directly with the confidence in the data, α_{update} is formulated as:

$$\alpha_{\text{update}}(T) = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot T \tag{6}$$

where $[\alpha_{\min}, \alpha_{\max}]$ bounds the smoothing (e.g., $[0.6, 0.95]$) to prevent extreme parameter drift. Consequently, high T securely preserves history against noise, whereas low T swiftly absorbs new states for fast convergence.

5 Evaluation

We implement the hStart prototype for the QUIC protocol, which only requires server-side modifications, including the HASS synchronization agent and the APFA congestion control logic. We evaluate it through controlled simulations (§5.2) and a large-scale production A/B testing (§5.3). The evaluation aims to answer the following three core questions.

Consistent Robustness and Fairness. hStart maintains superior performance across different network conditions, particularly in weak networks and high-jitter scenarios. Compared to aggressive static configurations, hStart improves throughput without significantly increasing SRTT (Smoothed RTT) (§5.2).

Performance in Real-world Environments. We deployed hStart in a large-scale mobile web service. A/B testing based on real users shows that hStart significantly increases throughput and improves user experience metrics, such as the number of requested video chunks and the rebuffering rate, without introducing significant negative impacts (§5.3).

Design Effectiveness and Low Overhead. We analyzed the effectiveness of the HASS mechanism in handling massive state synchronization. Experimental data show that hStart achieves high-precision state hits with extremely low bandwidth overhead (§5.4).

5.1 Experimental Setup

Controlled Simulation Environment. To investigate behavior under specific network conditions, we utilized Mahimahi to build a reproducible testbed. First, we evaluated performance with constant bandwidth across different file sizes. Furthermore, based on previously collected real-channel traces, we evaluate the system in three typical network scenarios, parameterized as shown in Table 1. We deployed the server on three different machines. When a client sends a request, it is randomly assigned to one of the servers to simulate users requesting resources from different CDN nodes.

Baselines. To demonstrate its effectiveness, the experimental evaluation consists of the following mechanisms:

Table 1: Parameter settings for emulated network environments.

Profile	Bandwidth	RTT	Lose Rate	Description
Stable High Bandwidth	50–100 Mbps	< 30ms	< 0.1%	Indoor Wi-Fi/5G
Mobile Weak Network	2–20 Mbps	30–200 ms	< 1%	Fast-moving (High Jitter)
High Packet Loss Scenario	20–100 Mbps	< 30ms	2%–5%	Poor edge coverage

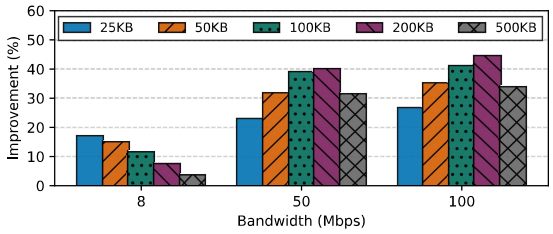
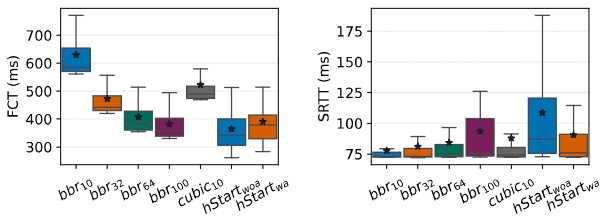


Figure 5: Throughput improvement over baseline under varying file sizes and bandwidths.



(a) Flow Completion Time (FCT). (b) Smoothed RTT (SRTT).

Figure 6: Performance distributions in different network conditions.

- **BBR$_{10}$, BBR$_{32}$, BBR$_{64}$, BBR$_{100}$**: Standard IETF QUIC [23] using BBRv1 [24] with IW of 10 (default standard), 32, 64 and 100 (aggressive static configurations), respectively.
- **Cubic$_{10}$**: Standard IETF QUIC using Cubic [25] with IW=10.
- **hStart$_{\mathbf{woa}}$**: Ablation variant applying historical states directly without the APFA mechanism.
- **hStart$_{\mathbf{wa}}$**: The complete framework equipped with APFA.

Large-scale Production Environment. We deploy hStart on the edge nodes of one of the world's largest CDN vendors, which provide over thirty million video streaming services across three provinces in China. Adopting an A/B testing methodology, we randomly assigned 50% of user traffic to the experimental group (enabled with hStart) and the other 50% to the control group (using default configurations) for a duration of 7 days. We collect key metrics via server-side access logs and client-side logs, including throughput (evaluated using average throughput rather than flow completion time due to varying video chunk sizes), video bitrate, the average number of requested video chunks per user, and the application-layer rebuffering rate.

5.2 Simulation Results

Theoretical Optimum under Constant Network. We tested different file sizes under a constant simulated network to explore the theoretical optimal gains of the scheme. As shown in Figure 5, the experimental results demonstrate that hStart achieves significant performance improvements compared to the baseline (BBR_{10}) in all test scenarios. We found that when files are too small or network bandwidth is too low, the default initial parameters are sufficient for transmission. When files are excessively large, the cold start process accounts for a very small proportion of the total transmission time, so the performance improvement in these cases is not significant.

Benchmarking in Dynamic Network Environments. To verify the algorithm's robustness in rapidly changing networks where user network information stored in the data synchronization layer may have changed significantly, we use the Mahimahi [26] network emulation tool to build a reproducible network test environment. We orchestrated different link conditions via shell scripts to accurately reproduce mobile network characteristics. (1) *Network Environment Emulation*: Using Mahimahi's `mm-link` and `mm-delay` tools, we constructed three typical network scenarios (summarized in Table 1). (2) *Traffic Model*: We replayed real short video application traffic logs, containing video slices of different sizes and covering both unpopular and popular content.

As shown in Figure 6, compared to baseline (BBR_{10}), the experiments show that the median FCT of hStart was reduced by 35%, and its 95th percentile was reduced by 34%. Under the default algorithm, FCT is longer, but it possesses the lowest SRTT. This is because the default algorithm probes from a lower window, which requires more time to converge to the maximum network bandwidth and results in less congestion during the process. More aggressive initial congestion window configurations (BBR_{32}, BBR_{64} and BBR_{100}) can shorten the FCT to some extent, confirming that a larger IW is an important source of startup acceleration. However, static large-IW schemes apply the same window to every connection regardless of user history or path dynamics, which makes them vulnerable to mismatched assumptions. In a representative low-bandwidth setting (e.g., 8 Mbps), setting IW to 100 leads to severe packet loss. BBR$_{100}$ incurs an 18% loss rate, while hStart reduces it to 11% through APFA-based validation and fallback. For hStart, when the APFA module is disabled, and the user network information returned by the synchronization layer is crudely set as the initial window, it achieves the optimal FCT performance, but also causes the worst SRTT. After enabling the APFA module, thanks to its confidence decay mechanism, hStart can keenly identify network fluctuations or the obsolescence of historical information. It rapidly falls back to a conservative strategy when detecting anomalies. Therefore, hStart is not simply equivalent to BBR$_{64}$ or BBR$_{100}$, it adaptively derives the startup window from cross-node NQI and validates it before committing to an aggressive rate. This allows hStart to achieve FCT performance close to the aggressive approach while simultaneously reducing SRTT and packet loss, reducing the likelihood of congestion in dynamically changing networks.

5.3 Real-world Deployment

Throughput Performance. Table 2 illustrates the performance gains under real user access. Compared to baseline (BBR_{10}), hStart

Table 2: Performance comparison on real-world test.

Metric	Baseline	hStart	Ratio*
Overall Thr. (Mbps)	21.20	25.27	19.15%
First Flows Thr. (Mbps)	16.70	24.46	46.47%
Video Bitrate (Kbps)	794	826	4.03%
Request Chunks (#)	3.25	4.25	30.77%
Rebuffering Ratio (%)	0.14	0.11	21.43%

*Ratio indicates the relative performance improvement over the baseline.

improves the overall average throughput by 19.15% and increases the average throughput of the first flows by 46.47%. This significant improvement is mainly attributed to the hot-start mechanism of hStart. While Default-QUIC can gradually probe bandwidth via slow start, it wastes 3–4 RTTs during the initial phase to increase the window. In contrast, hStart uses historical information to transmit data at a higher rate, reducing bandwidth waste.

User Experience. During the experiment, we also collected data closely related to user experience. Table 2 indicates that in the hStart experimental group, the average video bitrate watched by users increased by 4.03%, the average number of requested video chunks per user increased by 30.77%, and the video rebuffering rate decreased by 21.43%. This implies that we not only improved the quality of video watched by users but, more importantly, the increased number of requested video chunks suggests improved user retention for the *video application*. For *CDN providers*, this translates to increased data transmission. This benefits revenue growth for both parties, thereby enhancing our competitiveness [27].

5.4 Overhead

We evaluated the overhead and efficiency of the HASS mechanism in handling large-scale state synchronization. We measured the query hit rate, query latency, and storage and CPU usage of the data synchronization layer. As shown in the Figure 7, within one hour of hStart deployment, the query hit rate was 4.09%. As time passed and more user information was collected, the query hit rate increased, reaching 71.34% 24 hours after deployment. In steady-state operation, the average query latency for user information from the data synchronization layer is 5 ms, with storage usage of 300 MB and CPU usage of 3%.

In high-concurrency edge scenarios (with state update frequencies reaching 10,000 times per second), adopting a traditional Full-Mesh broadcast strategy would impose a prohibitive uplink bandwidth overhead, rendering the approach practically infeasible. After adopting HASS's hierarchical and on-demand synchronization strategy, the proportion of synchronization traffic was significantly compressed to less than 0.8%. Meanwhile, thanks to precise sensing of hot traffic, the system maintained a hot user state hit rate of 75%–85%. This means that connections for the vast majority of high-value users could reuse historical information. HASS successfully balanced synchronization overhead and performance gains, proving its high feasibility for engineering deployment.

6 Conclusion

To the best of our knowledge, hStart is the first work on an adaptive initial parameter tuning scheme that runs at the *multi-CDN as a collective service* paradigm. To overcome distributed state synchronization challenges, hStart efficiently minimizes overhead via HASS, which prunes synchronization traffic by using multi-dimensional filtering and ensures congestion safety via APFA, a closed-loop trust system that handles network dynamics by utilizing historical data as probabilistic hints. Large-scale production A/B testing validates this design, demonstrating a higher throughput and a significantly improved user experience. By breaking information silos between edge nodes, hStart establishes a practical foundation for next-generation stateful transport infrastructure.

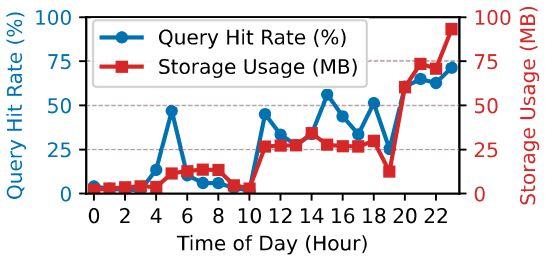


Figure 7: HASS data sync layer performance.

Our next step is to expand the geographical and topological scale of our large-scale deployment, validating hStart's robustness across diverse global ISP environments and cross-domain edge networks. Furthermore, we plan to conduct more comprehensive experimental comparisons. Although our current evaluation demonstrates significant improvements over default congestion control algorithms, future studies will rigorously benchmark hStart against other state-of-the-art transport initialization mechanisms and parameter-caching schemes to thoroughly contextualize its performance advantages and overhead boundaries.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 62572473, No. 62441230), the Scientific Research Innovation Capability Support Project for Young Faculty (No. SRICSPYF-ZY2025001), the National Science Foundation for Distinguished Young Scholars of China (No. 62425201), the funding from the State Key Laboratory of Internet Architecture (No. HLW2025ZD17), and the Tencent Basic Platform Technology Rhino-Bird Focused Research Program.

References

[1] Anika Schwind, Cise Midoglu, Özgü Alay, Carsten Griwodz, and Florian Wamser. 2020. Dissecting the Performance of YouTube Video Streaming in Mobile Networks. 30, 3 (2020), e2058.

[2] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. 2015. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. 381–394.

[3] Mark Allman, Vern Paxson, and Ethan Blanton. 2009. *TCP congestion control.* Technical Report.

[4] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *ACM SIGCOMM* (2017). 183–196.

[5] Jerry Chu, Nandita Dukkipati, Yuchung Cheng, and Matt Mathis. 2013. RFC 6928: Increasing TCP's initial window.

[6] Dan Liu, Mark Allman, Shudong Jin, and Limin Wang. 2010. Congestion Control Without a Startup Phase. https://api.semanticscholar.org/CorpusID:2298441

[7] Qingxi Li, Mo Dong, and P. Brighten Godfrey. 2015. Halfback: Running Short Flows Quickly and Safely. In *ACM CoNEXT* (2015). 1–13.

[8] Mahdi Arghavani, Haibo Zhang, David Eyers, and Abbas Arghavani. 2024. SUSS: Improving TCP Performance by Speeding Up Slow-Start. In *ACM SIGCOMM* (2024). 151–165.

[9] Kazuho Oku. 2026. *Rapid Startup of Congestion Control.* Internet-Draft draft-kazuho-ccwg-rapid-start-02. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-kazuho-ccwg-rapid-start/02/ Work in Progress.

[10] Xiaohui Nie, Youjian Zhao, Zhihan Li, Guo Chen, Kaixin Sui, Jiyang Zhang, Zijie Ye, and Dan Pei. 2019. Dynamic TCP Initial Windows and Congestion Control Schemes Through Reinforcement Learning. 37, 6 (2019), 1231–1247.

[11] Jia Zhang, Haixuan Tong, Enhuan Dong, Xin Qian, Mingwei Xu, Xiaotian Li, and Zili Meng. 2024. Cold Start or Hot Start? Robust Slow Start in Congestion Control with A Priori Knowledge for Mobile Web Services. In *ACM WWW* (2024). 2870–2878.

[12] Lars Eggert, John Heidemann, and Joe Touch. 2000. Effects of Ensemble-TCP. 30, 1 (2000), 15–29.

[13] Safiqul Islam, Michael Welzl, Kristian Hiorth, David Hayes, Grenville Armitage, and Stein Gjessing. 2018. ctrlTCP: Reducing Latency through Coupled, Heterogeneous Multi-Flow TCP Congestion Control. In *IEEE INFOCOM Workshop* (2018). 214–219.

[14] Safiqul Islam, Michael Welzl, and Stein Gjessing. 2018. Lightweight and Flexible Single-Path Congestion Control Coupling. In *IEEE NOMS* (2018). 1–6.

[15] Danyang Zhuo, Qiao Zhang, Vincent Liu, Arvind Krishnamurthy, and Thomas Anderson. 2016. Rack-Level Congestion Control. In *ACM HotNets* (2016). 148–154.

[16] Akshay Narayan, Frank Cangialosi, Deepti Raghavan, Prateesh Goyal, Srinivas Narayana, Radhika Mittal, Mohammad Alizadeh, and Hari Balakrishnan. 2018. Restructuring Endpoint Congestion Control. In *ACM SIGCOMM* (2018). 30–43.

[17] Tushar Dhoot and Joe Zhang. 2022. Reproducing Key Results from "Restructuring Endpoint Congestion Control". In *IEEE CPS-IoTBench* (2022). 17–21.

[18] Jiaxing Zhang, Furong Yang, Ting Liu, Qinghua Wu, Wu Zhao, Yuanbo Zhang, Wentao Chen, Yanmei Liu, Hongyu Guo, Yunfei Ma, et al. 2024. TECC: Towards efficient QUIC tunneling via collaborative transmission control. In *USENIX NSDI*. 253–266.

[19] Bo Wu, Tong Li, Cheng Luo, Xu Yan, Fuyu Wang, Changkui Ouyang, Linfeng Guo, Haiyang Wang, and Ke Xu. 2024. Reducing First-Frame Delay of Live Streaming by Simultaneously Initializing Window and Rate. In *IEEE ICDCS* (2024). 232–242.

[20] Ziqi Wei, Zhiqiang Wang, Qing Li, Yuan Yang, Cheng Luo, Fuyu Wang, Yong Jiang, Sijie Yang, and Zhenhui Yuan. 2024. QDSR: Accelerating Layer-7 Load Balancing by Direct Server Return with QUIC. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 715–730.

[21] Shuo Li, Steven Chien, Tianyi Gao, and Michio Honda. 2026. Remote TCP Connection Offload and Applications. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. 527–541.

[22] Wikipedia. 2023. Coefficient of variation. https://en.wikipedia.org/wiki/Coefficient_of_variation.

[23] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. https://www.rfc-editor.org/info/rfc9000

[24] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-Based Congestion Control. 60, 2 (2017), 58–66.

[25] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A New TCP-friendly High-Speed TCP Variant. 42, 5 (2008), 64–74.

[26] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. 417–429. https://www.usenix.org/conference/atc15/technical-session/presentation/netravali

[27] Ioannis Arapakis, Xiao Bai, and B. Barla Cambazoglu. 2014. Impact of Response Latency on User Behavior in Web Search. In *ACM SIGIR* (2014) *(ACM SIGIR)*. 103–112.