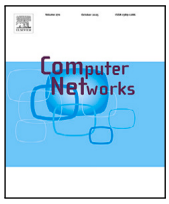
Computer Networks 284 (2026) 112352

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



ADePT: Latency prediction for edge CDN traffic scheduling via causal inference

Chuanqing Lin [a,b], Gerui Lv [a,b], Yangguang Liang [c], Fuhua Zeng [c], Xiaodong Li [c], Qinghua Wu [a,b], Zhenyu Li [a,b], Gaogang Xie [b,d,*]

[a] Institute of Computing Technology, Chinese Academy of Sciences, China
[b] University of Chinese Academy of Sciences, China
[c] Independent Researcher, China
[d] Computer Network Information Center, Chinese Academy of Sciences, China

ARTICLE INFO

Keywords:
Causal inference
Latency prediction
Content delivery
Traffic scheduling

ABSTRACT

To satisfy the unprecedented Quality of Experience (QoE) and stringent latency Service Level Agreements (SLAs) of emerging interactive applications, modern Content Delivery Networks (CDNs) are deploying massively decentralized edge nodes. However, this paradigm shift poses a significant challenge: optimal traffic scheduling fundamentally depends on acquiring real-time, full-coverage end-to-end path latency data to prevent SLA violations. Current measurement methods cannot scale to monitor every possible user-to-node path, and traditional prediction approaches (e.g., relying on additional segmented measurements or low-rank matrix decomposition) fail to achieve satisfactory accuracy on the resulting extremely sparse datasets.

In this work, we present ADePT (Application Delay PredicTion), a novel data-driven causal inference framework that provides comprehensive and precise latency predictions without requiring additional measurements. By explicitly decoupling user-side temporal variations (e.g., last-mile congestion) and node-side spatial variations (e.g., core propagation delays), ADePT successfully extracts high-dimensional latent embeddings from limited measurement data to infer the end-to-end path latency for any potential scheduling decision. Evaluated on a massive real-world dataset from a leading edge CDN, ADePT reduces prediction errors by 19.6% and achieves a median absolute error of 4.3 ms. Consequently, integrating ADePT's accurate predictions into CDN traffic scheduling significantly improves scheduling decisions, increasing the ratio of traffic meeting strict applications' latency requirements by 1.66×.

1. Introduction

Recently, the surge of interactive applications, such as live video streaming and virtual reality [1–5], has imposed unprecedented Quality of Experience (QoE) demands. To deliver flawless user experiences, such as eliminating video stalling and accelerating page load times, modern Content Delivery Networks (CDNs) proactively cache frequently queried content close to users [6]. However, fulfilling these rigorous QoE expectations requires CDNs to strictly adhere to stringent Service Level Agreements (SLAs), which are fundamentally dictated by the end-to-end path latency. To consistently guarantee that transmission performance meets these ultra-low latency SLAs, vendors have evolved toward edge CDNs. This paradigm shift involves deploying highly decentralized, densely distributed nodes in extreme proximity to users (typically at the city level per ISP, scaling into the thousands [7,8]).

At the core of this decentralized infrastructure is the traffic scheduling center, which dynamically assigns user requests to specific edge nodes. The ultimate objective of traffic scheduling is to fulfill diverse SLA requirements, thereby directly promising users' QoE. However, unlocking the full potential of this dense edge infrastructure requires intelligent scheduling, which intrinsically depends on acquiring comprehensive and accurate end-to-end path latency data. Only with precise latency visibility can the scheduler avoid congested network paths, prevent SLA violations, and continuously guarantee optimal QoE for all user groups. Yet, the massive scale-up of edge nodes poses a formidable monitoring challenge. Specifically, a single round of global traffic rescheduling (typically every 5 min) requires real-time latency evaluation for every potential user-to-node path, resulting in an explosive demand of over 10 million monitoring items in our operational scenarios.

Prevailing measurement methods are incapable of meeting the requirement for explosive measurement data (Section 2.2). As shown in

* Corresponding author.
E-mail addresses: linchuanqing20b@ict.ac.cn (C. Lin), xie@cnic.cn (G. Xie).

https://doi.org/10.1016/j.comnet.2026.112352
Received 25 September 2025; Received in revised form 10 March 2026; Accepted 30 April 2026
Available online 6 May 2026

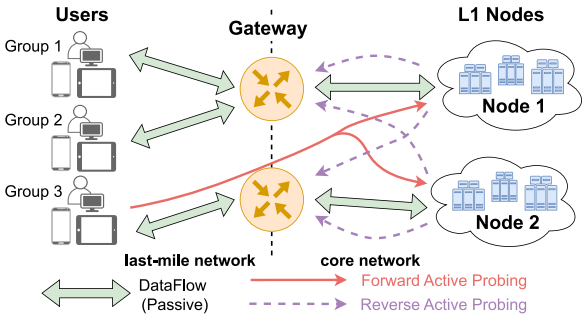


Fig. 1. Measurement approach adopted.

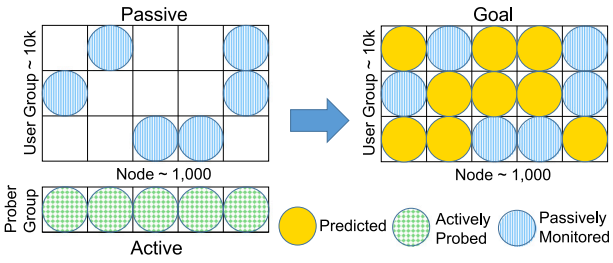


Fig. 2. The goal for latency prediction modeling.

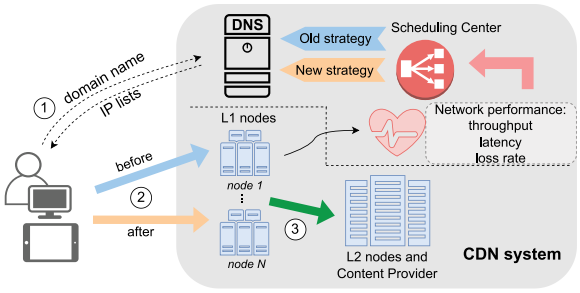


Fig. 3. Overview of the edge CDN system.

Fig. 1, widely adopted measurement methods include active probing and passive monitoring. *Active probing* methods [9,10] can be issued either forwardly from users to nodes or reversely from nodes to users. These methods can cover all possible nodes but fail to cover all potential user groups, thereby lacking representativeness for the actual latency of specific applications. *Passive probing* methods [11] can only monitor the end-to-end path latency for established paths where traffic traverses, which lacks coverage of all nodes.

The scale-up challenge necessitates a shift from direct latency measurement to latency predictive modeling (Fig. 2), which heavily relies on the efficient extraction of latent embeddings and their correlations from the measurement dataset. However, traditional approaches, such as global network positioning (GNP) [12–15], and segmented measurement [16–20], focus on direct measurement for all path segments, which is challenging for CDNs. These works fall short in the embedding extraction and correlation learning, thereby failing to achieve the desired latency prediction accuracy (Section 2.3).

Our insight lies in the fact that latency prediction can be performed based on currently deployed measurements, which have already measured all potential path segments. The inherent latent embeddings of these segments have been captured in the measurement data, though they may be highly coupled. Therefore, a well-designed framework is still needed to extract such high-dimensional latent embeddings and their correlations from the measurement dataset. In this context, low-rank decomposition-based matrix completion methods [21–24] fail due to the extremely sparse nature of the revealed dataset.

This work presents the Application Delay PredicTion (ADePT), a causal approach to perform comprehensive and precise latency prediction without additional measurement tasks. ADePT employs a data-driven learning approach to infer latent embeddings and learn their complex nonlinear correlations. To predict the end-to-end path latency for any potential path, ADePT first extracts latent embeddings from the historical measurement dataset. Then it incorporates all latent embeddings together to predict the underlying end-to-end path latency for the given path.

Our measurement reveals that the end-to-end path latency variation can be explicitly decoupled into two components: the user-side temporal variation (driven by the dynamic processing and queueing delay in the last-mile network over time) and the node-side spatial variation (driven by the stable propagation delay when being scheduled to geographically distinct static nodes) (Section 3). ADePT adopts an adversarial neural network architecture (Section 4) to extract high-dimensional latent embeddings for both the node and user sides, capturing a more nuanced representation of the path's network state compared to simple, handcrafted features. These learned embeddings are subsequently fed into a separate network to predict the true end-to-end path latency. This enables the model to learn the complex non-linear correlation between user and node side embeddings from the extremely sparse dataset, and bypasses the need for explicit segmented measurement or simple linear summation.

ADePT achieves a higher level of accuracy in predicting end-to-end path latency, making it a more robust and effective solution for scheduling in large-scale edge CDN environments. We evaluate ADePT with real-world datasets collected from one of the leading edge CDN vendors in China. Compared with existing solutions, ADePT can reduce 19.6% of latency prediction errors and achieve 4.3 ms of absolute prediction error on medium. With ADePT's accurate latency prediction data, the scheduling system can reduce the Global Latency Penalty (the optimization objective score defined in Eq. (8)) by 8.6% and increase the ratio of traffic meeting the ideal latency requirement (i.e., the SLA threshold) from 29.6% to 49.1% (1.66×).

The contributions of this paper are summarized as follows:

- We provide a comprehensive analysis of real-world CDN data, identifying that end-to-end path latency variation is composed of distinct user-side temporal and node-side spatial factors (Section 3).
- We propose a novel data-driven causal inference framework, ADePT, for end-to-end path latency prediction in large-scale edge CDNs. ADePT first extracts high-dimensional latent embeddings of both node and user sides and subsequently predicts end-to-end path latency through these embeddings (Section 4).
- We verify that ADePT achieves a higher latency prediction accuracy and effectively helps the scheduling system to schedule traffic with higher overall performance through a real-world collected dataset (Sections 5–6).

The remainder of this paper is structured as follows. We introduce the background and motivation in Section 2. A large-scale measurement is shown in Section 3 to understand the impacting factors of end-to-end path latency and determine input features for the latency prediction model. In Section 4, we formalize the latency prediction problem and propose ADePT to predict end-to-end path latency. We further discuss how to employ the latency prediction data in the scheduling problem to achieve better scheduling scores in Section 5. The prediction accuracy and the improvement in the scheduling problem are evaluated in Section 6. Section 7 discusses the generalizabilty for ADePT. Sections 8 and 9 introduce the related work and conclude our work.

2. Background and motivation

2.1. Background

Edge CDN system architecture. As illustrated in Fig. 3, an edge CDN system usually contains three components: *(i)* geographically distributed nodes, *(ii)* a centralized scheduling center, and *(iii)* DNS servers. At the data plane, edge nodes form a hierarchical structure, with the majority of L1 nodes (also known as edges or Pops) located close to the users. In contrast, L2 nodes are relatively more centralized. At the control plane, the scheduling center collects real-time monitoring data from nodes and dynamically adjusts the *scheduling mapping*, which maps users' traffic flow to specific L1 edge nodes. Subsequently, DNS servers apply the scheduling mapping.

Workflow. The workflow for a user's request is marked in Fig. 3. The user first recursively issues probes for the target domain name through local DNS (LDNS) to determine the exact node to request from DNS servers (1), and then requests the desired content from the node (2). The request is directly responded to if the target content has already been cached; otherwise, the L1 node will fetch the content from L2 nodes or Content Providers (CPs) (3). This approach enables CDNs to reduce the network distance traversed by users to access their desired content, thereby decreasing the transmission latency (i.e., RTT).

Edge node characteristics. Compared to traditional CDN systems, edge CDNs deploy more heterogeneous nodes at a much higher density to meet the diverse and stringent performance requirements of various applications. Specifically, nodes can be constructed in each city per ISP [7,8] to promise extremely low latency, and equipped with different computation and memory resources, specialized for computation-intensive applications (e.g., encoding and decoding for live streaming services [25]) or large file downloading applications. As a trade-off, the bandwidth capacity of a single node is limited to tens of Gbps (compared to hundreds of Gbps in traditional CDNs). This capacity is only sufficient to serve requests from a small number of applications in neighboring geographical regions. Thereby, careful and fine-grained traffic scheduling is required to avoid node overloading while meeting the SLA requirements of applications.

Traffic scheduling. Traffic scheduling, which assigns traffic to designated nodes, is facilitated by the scheduling center. Specifically, the scheduling center is responsible for updating the following scheduling mapping M_t:

$$M_t : Domain \times Region \rightarrow Nodes \tag{1}$$

At each round t, user requests are grouped into *user groups* by their ⟨requested domain name, located region⟩ pairs, and assigned to specific nodes, respectively. The requested domain name indicates the corresponding application type, reflecting preferences for different types of nodes to achieve desired performances. The located geographical region is monitored through the IP address of users' LDNS or users' own (by eDNS [10]). To balance the traffic demand size across user groups, requests to the same application from neighboring regions (i.e., from the same province or city) are aggregated into a single unit and scheduled to the same group of nodes. Traditional methods that distinguish traffic from different/24 IP-prefix subnets become ineffective in the edge CDN scenario due to the tiny traffic demand size of a single application from a single subnet.

Foundations for optimal scheduling. In order to compute the optimal scheduling mapping (Eq. (1)), the scheduling center must acquire comprehensive and accurate end-to-end latency data across all potential user-to-node paths. This data acts as the fundamental compass for the scheduler to dynamically bypass congested networks, assign user groups to the most suitable edge nodes, and ensure that transmission delays remain strictly below application-specific SLA thresholds. Without such global visibility, the system risks severe SLA violations (e.g., video buffering or elevated response times) that directly degrade user QoE. To summarize, *acquiring real-time, large-scale end-to-end path latency is not merely a monitoring task, but the indispensable foundation for the entire edge CDN system to guarantee optimal user experiences.*

Definitions and scope. In this paper, we exclusively focus on the user content delivery phase, where real-time network transmission directly impacts the user's QoE. This explicitly excludes the CDN's internal cache population, disk I/O operations, or application-layer processing. Accordingly, the end-to-end path latency discussed herein strictly refers to the network/transport-layer transmission delay (e.g., Round Trip Time) for the path between the end-user's device and the assigned CDN edge node.

2.2. Existing approaches: Limitations and gaps

Latency measurement scale. The scale for the real-time latency measurement job can be huge in a large-scale edge CDN system. Ideally, all possible mapping entries in Eq. (1) should be monitored, leaving the real-time measurement job at the level of $O(N \times R \times D)$, where N, R, D denote the number of nodes, user regions, and serving domain names, respectively. Currently, a practical large-scale edge CDN system typically deploys more than 1000 edge nodes and serves thousands of applications in hundreds of regions, contributing to a Tbps level of real-time traffic demand. Consequently, the number of required measurement jobs for each round of scheduling exceeds 100M, posing significant challenges for the current measurement methods.

Measurement methods. CDN vendors collect measurement data through various methods at different network layers, which are illustrated in Fig. 1. (1) *Active measurement:* CDN vendors can measure the latency of the desired path by actively issuing probes. Some works (e.g., Microsoft's Odin [9]) preserve constant probing sources at the user side and issue active probes (e.g., periodical ICMP PING, HTTP GET requests, DNS requests) from sources to measure network latency. Conversely, some works (e.g., Akamai [10,38]) issue active probes (periodical ICMP PING) from their nodes to LDNS or gateways located closest to users. (2) *Passive measurement:* CDN vendors can also measure the actual latency of existing connections. Some work (e.g., Facebook [11]) captures the connection states from nodes' TCP stacks, HTTP servers, or users' browsers. To measure latency for a broader range of nodes, some work (e.g., Google [36]) proactively schedules a small group of traffic to multiple nodes and collects measurement data from corresponding nodes.

Limitations. Table 1 summarizes the approaches and limitations of prior measurement works. As discussed above, ideal path latency information provided should cover all possible *(R1)* end-to-end paths, *(R2)* from all user groups, *(R3)* to all edge nodes; while still promising accuracy. Current measurement approaches cannot provide the desired data that meets all of the goals simultaneously:

- Active probing from preserved user sources fails to represent the latency for all user groups. For example, we observe high diversity of average latency on a single edge node for content from different domain names when requested by users in the same regions, as illustrated in Fig. 4.
- Active reverse probing from CDN nodes cannot obtain the end-to-end path latency. The request performance is deeply impacted by the quality of the user-side last-mile network [39,40], which is neglected by this methodology.
- Passive monitoring fails to cover paths from users to all possible CDN nodes. Methods that proactively schedule traffic to multiple nodes still cannot tackle the problem because only a small proportion of possible paths can be covered due to the limit of the DNS redirection scheme. More seriously, scheduling traffic that requests the same domain name to multiple nodes simultaneously will compromise the efficiency of content caching, particularly on lightweight edge nodes.

Table 1
Current measurement methods.

Type	Works	Descriptions	R1: End-to-end path	R2: User group coverage	R3: Node coverage
Active	Odin [9], BISmark [26], NEL [27]	Actively probe through users	Yes	No	Yes
Active	DMS [28], King [29], [30]	Actively probe through DNS	No	Yes	Yes
Passive	CPR [31], Latlong [11,32–34]	Monitor on the network stack	Yes	Yes	No
Hybrid	EdgeFabric [35], Espresso [36,37]	Proactively redirect traffic to target nodes to measure	Yes	Yes	No

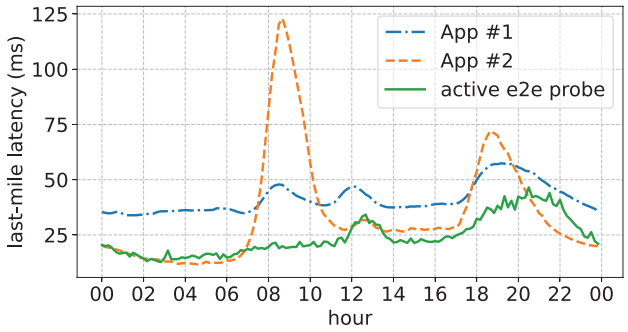


Fig. 4. Active end-to-end probing results are not consistent with actual traffic latency.

In summary, it is impractical to collect all the required latency data directly by current measurement approaches. However, as illustrated in Fig. 1, these measurement approaches are indeed capable of probing all potential path segments, thereby ensuring that the measurement data carries invariant latent embeddings of all path segments. Since the measurements have varying segment coverage ranges, these latent embeddings are highly coupled, making them difficult to extract and utilize directly. Therefore, end-to-end path latency prediction becomes a promising approach to exploit current measurement data to extract these latent embeddings and reconstruct the desired latency knowledge for the scheduling center.

2.3. Filling missing entries via latency prediction

The latency prediction task can be formulated as an entry-filling problem illustrated in Fig. 2. We can define a matrix M where m_{un} denotes the end-to-end path latency for a user group u (each row) assigned to node n (each column). According to the passive measurement, we observe m_{un} for each user group and several corresponding nodes. Conversely, active probing monitors from a full-node coverage perspective for specific probing groups. The goal of latency prediction is to fill missing entries in M, i.e., the underlying latency when a user group is scheduled to a node that has not yet been assigned. This completion task is typically accomplished by first inferring irrelevant latent embeddings for each axis (i.e., user groups and nodes), and then finding the correlation among these axes.

Segmented measurement. Prior works have proposed segmented measurement-based prediction approaches such as iPlane [16] and GNP [13]. These approaches measure each path segment separately to infer the segment latency Lat_u, Lat_n and consequently predict the end-to-end path latency through straightforward summation: $M_{un} = Lat_u + Lat_n$. The drawback of such methods is twofold. First, it is difficult for CDNs to measure each segment of the entire path (especially the last-mile network), because CDNs cannot deploy active probing on actual user groups. Second, the segmented measurement-based approach can only measure simplistic features from each segment and deploy a straightforward summation to predict the latency, which fails to capture the complex nonlinear correlation that influences the true end-to-end path latency. As shown in our experiments (Section 6.1), the segmented measurement-based methods cannot achieve the desired predictive accuracy.

Matrix completion. The effectiveness of matrix completion in real-world predictive problems where the low-rank structure holds has been proven, particularly in network measurements [21,22,24]. However, our experiments demonstrate that the average prediction error of standard matrix completion methods that rely on matrix decomposition exceeds 51%, indicating they are not capable of achieving the desired results.

The reason also comes from a two-fold. First, the number of revealed entries is insufficient to recover the original matrix. To achieve high cache efficiency, a user group is assigned to fewer than 4 nodes. As a result, for each row (user group) of the matrix, only a small portion of the entries is revealed. Prior works have shown that the lower bound on the number of revealed entries required to recover a matrix accurately is $4Ur - r^2$ [41]. Here, U denotes the maximum number of rows (user groups) and columns (nodes). In the case of rank $r = 2$, all user groups are required to be randomly assigned to at least 8 nodes, which is impractical in the real world. Second, although the low-rank structure may be valid in our scenarios, the precise rank remains unknown and may be non-integer, resulting in high prediction error for matrix completion.

Solution. The extremely sparse dataset and the complex correlation among segments motivate us to adopt a data-driven causal framework to perform high-dimensional latent embedding inference and non-linear end-to-end path latency prediction. Nonetheless, before designing a learning-based predictive model and determining the corresponding input features, it is important to first understand the factors that impact end-to-end path latency from both the user side and the node side, which the model is intended to extract from the dataset.

3. Identifying input features for end-to-end path latency prediction

In this section, we investigate the factors that influence the perceived latency between users and edge nodes, with the aim of designing the latency prediction model and selecting input features. We first describe the hybrid methodology we adopted to collect real-world latency measurement data (Section 3.1). Next, we split the end-to-end path into two independent segments to analyze the node-side and user-side impacting factors, respectively (Sections 3.2–3.4).

3.1. Dataset collection

In order to collect real-time measurement data that covers all possible paths, we adopt the following hybrid measurements in a leading CDN vendor in China, as illustrated in Fig. 1. All of the measurements last for one month and involve more than 1500 nodes serving 100+ Tbps real traffic demand, which comes from more than 10,000 domain names and 31 regions (i.e., provinces).

Active reverse measurement ①: We deploy active measurement from the node side to measure the core network latency. Based on the IP addresses of user requests recorded at CDN nodes, we maintain a list of routers located closest to users in all provinces through reverse `traceroute`, which respond to the `ICMP PING`. We let each L1 edge node randomly send at least 10,000 `ICMP PING` requests to some of the

routers in each province at each 5-minute time slot. This measurement collects the real-time latency for each ⟨province, node⟩ pair of the core network.

Active end-to-end path measurement ②: We also maintain a group of fixed user probers located in all provinces in China. Probers in each province periodically issue `ICMP PING` requests to each L1 node more than 300 times in each 5-minute time slot. This measurement records the end-to-end ⟨province, node⟩ path latency for a fixed group of users.

Passive measurement ③: We capture established connection states from the TCP stack and the HTTP server of edge nodes, aggregated at the ⟨user group, node⟩ level. Each item of record includes user province, requested domain, requested node, requesting time, connection build-up time, requested file size, transmission time, and retransmitted bytes. This measurement records the accurate mean performance (latency, throughput, loss rate) for users from each user group (⟨domain name, region⟩) to assigned nodes.

3.2. Path segmentation

We manage to split the end-to-end path into two segments: the last-mile segment and the core segment, to study the impacting factors of the user side and the node side, respectively. The former starts from the end device of users to the first router with a public IP address, and the latter refers to the rest of the link on the public Internet, as illustrated in Fig. 1. It is worth noting that the last-mile segment is in accordance with a fixed group of users instead of the fixed IP-prefix subnet.

We compute the latency of two path segments in the following way. The latency of the core segment Lat_n can be obtained directly through measurement ①. Whereas the last-mile segment latency Lat_u of each user group is hard to measure directly. Instead, we compute it through end-to-end path latency $Lat_{passive}$ collected from passive measurement ③ minus the corresponding core segment latency, which has been proven to be a practical method in [40,42,43]:

$$Lat_u = Lat_{passive} - Lat_n \tag{2}$$

The absolute value of Lat_u may be meaningless in practice, because $Lat_{passive}$ and Lat_n are collected from different network layers and different connection sides [27,44,45]. Nevertheless, this value remains adequate to indicate the aggregated variation feature of the last-mile segment for each user group. As we will show in Section 3.4, the computed results Lat_u show a high similarity among a fixed group of probers, even though probers are assigned to nodes with different geographical and network distances.

3.3. Node-side features

As the core network is maintained by ISPs, we observe that the node-side latency exhibits high stability and is primarily related to the geographical transmission distance.

High stability. We observe that the core segment latency exhibits high stability over a day, which can be attributed to the ISP's well-constructed facilities and efforts. According to measurement ①, the fluctuation in segment latency throughout a day is negligible for a fixed core network path, either on weekends or weekdays. Conversely, as evidenced by measurement ②, for a constant end-to-end path, the full path latency displays a distinct periodicity. Specifically, during periods of traffic peak (i.e., 9 p.m.), latencies are observed to exceed 25 ms compared to off-peak periods (i.e., 3 a.m.). This discrepancy in stability suggests that the deterioration in Quality of Service (QoS) during peak periods is not typically attributable to the core network.

Proportional to geographical distance. We observe a strong correlation between the mean core segment latency and transmitted geographical distance. Based on the data collected from measurement ①, we find an approximately 1.9 ms increase in latency for every 100 km geographical distance, with a high linear coefficient (R^2) of 0.92.

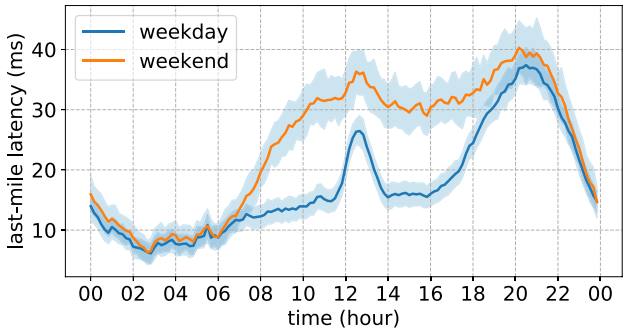


Fig. 5. Temporal pattern of the last-mile segment latency. The shadow part represents the sampling variability from all probes to all CDN nodes.

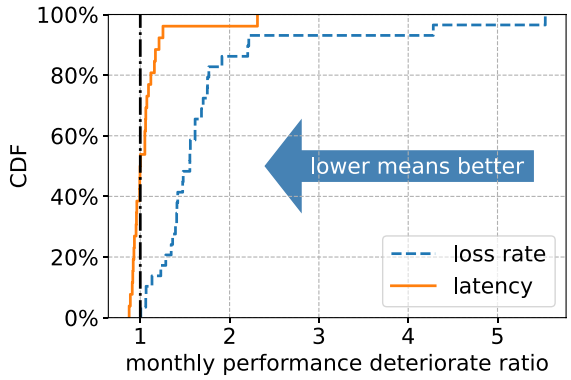


Fig. 6. CDF of the performance deterioration ratio (month end divided by month start).

Considering the propagation delay at the speed of light in fiber is close to 1 ms per 100 km, the processing and queuing delay by in-network routers does exist, but remains stable.

The core network latency monitored by measurement ① also has a strong similarity with the full path latency monitored by measurement ②. The mean Kendall similarity [46] can reach 73.7% over a month. This strong similarity indicates the core segment latency to be a good input feature for end-to-end path latency prediction.

Implication. The node-side spatial variation significantly influences the end-to-end path latency. This variation refers to the latency differences arising when traffic from the same user group is scheduled to geographically distinct static nodes. Given the stability of ISP core networks, this spatial variation is heavily dominated by the physical distance, i.e., the propagation delay over optical fibers. Consequently, *the core network latency monitored by the active reverse probe ① serves as a solid node-side input feature for predicting the propagation-dominated component of the end-to-end path latency.*

3.4. User-side features

Previous work [39,47] has pointed out that the last-mile segment becomes the bottleneck of the entire path. In this part, we provide a quantitative illustration of the temporal pattern of the last-mile segment latency and the diversity among different user groups.

Strong temporal pattern. The last-mile segment latency highly varies in one day and behaves differently on weekends and weekdays, as depicted by Fig. 5 through measurement ②. Based on our fixed probing users, the latency on weekdays peaks at midday and evening, while the latency on weekends continues to be high since morning. In the evening peak, the last-mile segment latency can rise to 40 ms,

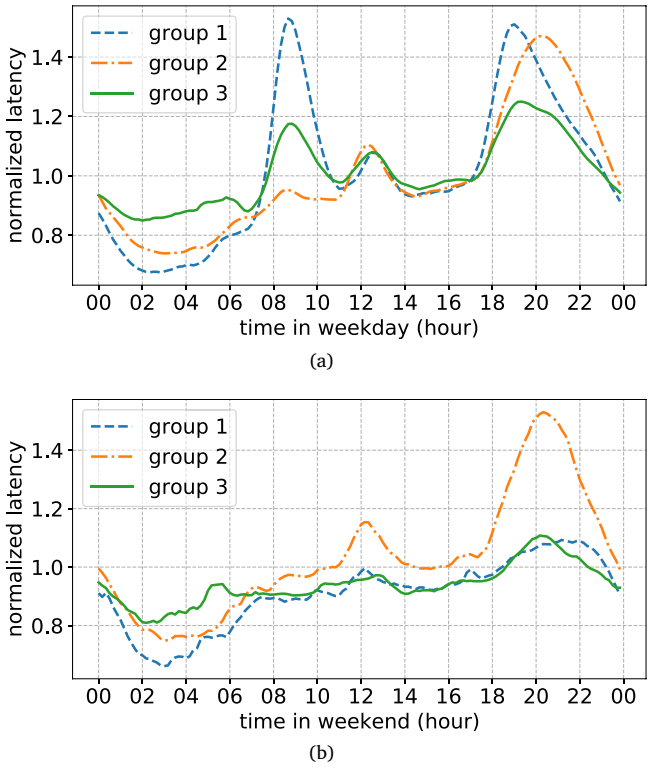


Fig. 7. Cluster center of each application group on weekdays or weekends.

accounting for more than 80% of the full path latency. The high last-mile segment latency, which is out of the control of CDN, severely hinders CDN's efforts to reduce the content delivery latency through traffic scheduling.

The last-mile performance also exhibits a monthly cyclic pattern. Fig. 6 illustrates the CDF of the performance deterioration ratio for users from all user groups monitored by measurement ③, which is computed by the latency (or loss rate) at the month end divided by that at the month start. Most applications witness increased latency and much more packet loss at the end of the month. For example, more than 50% of applications exhibit a 1.5× loss rate and higher end-to-end path RTT. This can be attributed to the possibility that cellular network users' transmission rate may be limited by ISP when the data cap is reached [48]. For different applications, the ratios of cellular network users result in the corresponding performance deterioration ratios.

High diversity among applications. Temporal patterns for different applications (or domain names, user groups) are highly diverse, based on their types of applications and use cases. Firstly, users exhibit a preference for divergent access methods across various applications. To illustrate, in comparison with cellular access with high latency, WiFi or wired access, with their steadier and lower latency, prove more suitable for long video-on-demand and live streaming applications, given these applications require high transmission performance and the transmission of large datasets. Conversely, access methods for music or short video applications appear to be more arbitrary. In practice, evidence suggests that different cellular access ratios are observed for different types of applications, according to the labeled data collected from the relevant content providers [48]. Secondly, the use case of different types of applications also varies. Reconsidering Fig. 4, application 2 (a music APP) shows increased latency during commuting hours, as users suffer poor cellular access performance while in transit.

Resulting from the above two reasons, the mean last-mile segment latency of different applications exhibits significant disparity. For example, the maximum mean last-mile segment latency exceeds 84 ms,

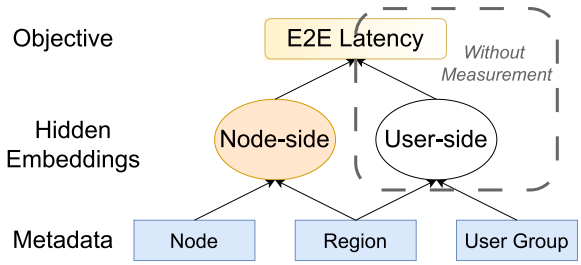


Fig. 8. Graph model of the prediction problem.

whereas the minimum is only 16 ms. If a latency monitor fails to diagnose an application's large last-mile segment latency and consequently the scheduling center does not specially optimize its assigned nodes, the corresponding QoS will face severe degradation.

In order to more effectively illustrate the temporal fluctuations in last-mile segment latency patterns, a clustering approach was employed on the daily last-mile segment latency time series of applications. For each user group, the last-mile segment latency is recorded in 10-minute time slots and subsequently normalized to trace the changing pattern. The pattern difference between weekdays and weekends is also considered. As demonstrated in Fig. 7, most applications can be clustered into three groups by employing the K-means clustering algorithm. The traffic of each group accounts for 14.5%, 25%, and 60.5% of the total traffic, respectively.

The primary distinctions between the three groups are as follows: (1) the severity of latency degradation, i.e., the disparity between the minimum and maximum value; (2) the similarity of the temporal pattern on weekdays and weekends. For example, Group 2 demonstrates comparable patterns during weekdays and weekends, suggesting a similar use case. However, Group 1 experiences more severe latency degradation during weekday commuting hours, while the latency changes exhibited by Group 3 are more modest.

Implication. The user-side temporal variation also heavily influences the end-to-end path latency. This variation captures the temporal latency fluctuations within the last-mile access network for stationary user groups. These fluctuations are predominantly characterized by processing and queueing delays at the congested ISP edge. Furthermore, these temporal patterns exhibit a high degree of diversity among different aggregated user groups (applications) due to their unique usage behaviors. Therefore, *application types and time information are both indispensable user-side input features to capture the queueing-dominated temporal dynamics for end-to-end path latency prediction.*

4. ADePT design

Measurements in Section 3 indicate that the end-to-end path latency can be predicted based on the features of both nodes and user sides. In this section, we propose ADePT, an Application Delay Prediction tool. We first formulate the problem in Section 4.1. In Sections 4.2–4.4, we introduce the design and deployment issues of the causal neural network (NN) predictor, respectively.

4.1. Problem statement

As analyzed in Section 3, the end-to-end path latency for a specific application (or user group) $u \in U$ from users in region $r \in R$ to the node $n \in N$ is depicted by the mapping $Lat^t(u, r, n)$. If we consider the impacting factors of the user side and the node side separately, the mapping can be rewritten to:

$$Lat^t(u, r, n) = f^t(User^t(u, r), Node^t(r, n)) \quad (3)$$

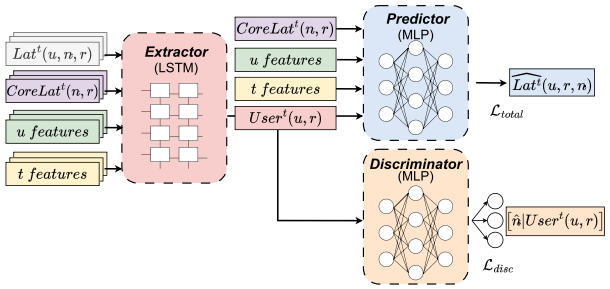


Fig. 9. ADePT architecture.

where $User^t(\cdot)$ and $Node^t(\cdot)$ denote the hidden embedding of the user group and the node, respectively. The function f^t illustrates the correlation between two embeddings. Such a relation is illustrated in Fig. 8.

The monitoring system collects two real-time measurements: *(i)* the partly observed $Lat^t(u,r,n)$: for any user group u in region r, the end-to-end path latency is logged by passive measurement ③ on the assigned node n; *(ii)* the full observation for $Node^t(r,n)$: the active reverse measurement ① monitors the latency for each node n to all regions r. Note that the observed core network latency is not entirely equivalent to the hidden high-dimensional node-side embedding.

The goal of the predictive model is to learn the mapping $Lat^t(u,r,n)$ completely. Formally, for any given user group u at region r, we wish to predict the end-to-end path latency when it is assigned to any possible node n at time t.

4.2. Causal NN model

As discussed in Section 2.3, traditional prediction methods cannot efficiently capture the hidden high-dimensional embeddings and their correlations. The data-driven learning model, conversely, has been proven to be more proficient in accomplishing this task with limited data. However, a naive supervised NN model may be unsuitable as the user-side embedding $User^t$ in Eq. (3) cannot be directly observed by CDNs. Inspired by former causal inference works [48,49], we model this problem as a counterfactual prediction and split it into two steps:

- predicting the hidden user-side embedding $User^t(u,r)$ for each user group u by observed $Lat^t(u,r,n)$ and the corresponding observation $Node^t(r,n)$.
- making counterfactual predictions of end-to-end path latency $\widehat{Lat^t}(u,r,n)$ for specific nodes through function f^t with user-side and node-side embeddings.

The key challenge of ADePT is to extract the node-invariant user-side embedding $User^t(u,r)$ for each user group u, which is achieved by the adversarial neural network framework.

ADePT architecture. As illustrated in Fig. 9, ADePT consists of three neural networks: the hidden user-side embedding extractor, the hidden embedding discriminator, and the end-to-end path latency predictor.

Extractor. To extract the high-dimensional hidden user-side embedding $User^t(u,r)$, ADePT adopts an LSTM that takes in $Lat^t(u,r,n)$, corresponding $Node^t(r,n)$, features of the user group u, and explicit time-related features (denoted as *t features*). Based on the temporal patterns identified in Section 3.4, we design *t features* to include `HourOfDay`, `DayOfWeek`, and `DayOfMonth` to capture the intra-day, weekly, and monthly patterns of last-mile networks, respectively. The input data is aggregated at a 10-minute granularity. To capture the hidden state stably and filter out real-world measurement noise, the LSTM takes a sliding window of historical records spanning 3 h (i.e., a sequence of 18 timesteps) as input.

Discriminator. ADePT employs a multilayer perceptron (MLP) to ensure the invariance of the hidden user-side embedding according to nodes. This NN aims to predict the assigned node $\hat{n}$ for a given predicted user-side embedding $User^t(u,r)$. If the hidden embedding is invariant to the assigned node, as expected, the discriminator will fail to distinguish the assigned node but will predict randomly at the prior probability. ADePT uses cross-entropy loss to train the discriminator:

$$\mathcal{L}_{disc} = L_{cross-entropy}(n,\hat{n}) \tag{4}$$

Predictor. ADePT uses another MLP to extract node-side embeddings and then predict the counterfactual end-to-end path latency. This predictor takes the predicted hidden embedding $User^t(u,r)$, the core network observation (to extract node-side embedding $Node^t(r,n)$), features of the user group u, and the current *t features* (`HourOfDay`, `DayOfWeek`, `DayOfMonth`) as input, and outputs the predicted counterfactual latency $\widehat{Lat^t}(u,r,n)$. ADePT formulates a Mean Squared Error(MSE) loss for the predictor:

$$\mathcal{L}_p = \mathbb{E}\left[\left(Lat^t(u,r,n) - \widehat{Lat^t}(u,r,n)\right)^2\right] \tag{5}$$

4.3. Training

Similar to the training procedure of the Generative Adversarial Network (GAN) framework, ADePT alternates between two training phases. Firstly, ADePT trains the discriminator through loss $\mathcal{L}_{disc}$ with fixed extractor and predictor parameters. Subsequently, the extractor and the predictor are trained by the following total loss $\mathcal{L}_{total}$ while keeping the discriminator fixed:

$$\mathcal{L}_{total} = \mathcal{L}_p - k \cdot \mathcal{L}_{disc} \tag{6}$$

where k is the hyperparameter to balance the trade-off between model fitting and generalization. Note that here the negative sign of the discriminator loss means the extractor is trained to fool the discriminator to ensure the hidden state is invariant to the assigned node.

The NN structure and hyper-parameters are listed in Table 2. The training procedure takes less than 2 h on a 16-core CPU with 150k samples.

4.4. Deployment issue

Data cleaning. The real-world collected latency data needs careful pre-processing. Firstly, the measurement data is unstable and exhibits huge noise. Secondly, the latency often incurs long-tailed outliers and does not follow a Gaussian distribution. To fix this, ADePT first applies a logarithmic transformation to all of the latency data and then performs the standardization. This approach can effectively limit the impact of the instability of the real-world data on the learning model.

Node labeling. The labeling of nodes for the discriminator is complex because the number of nodes in the edge CDN system exceeds 1000. ADePT labels nodes by their observed core network latency for users in each region. The latency metric, which is continuous, is discretized into several uniform bins, thereby transforming the discriminator's task into a classification problem to accommodate the cross-entropy loss.

User group feature extraction. The user group feature is supposed to identify the seasonality and long-term trends of user groups, as shown in Section 3.4. Thereby, we collect the last-mile segment latency time series for each user group from the historical dataset and extract the daily, weekly, and monthly seasonality, respectively. Principal components of such features are further filtered out through principal components analysis (PCA) and selected as features of user groups.

5. Optimization of the scheduling problem

By utilizing more accurate end-to-end path latency predicted by ADePT as the latency metric input to the scheduling system, CDNs can

Table 2
Training setup and hyperparameters for ADePT.

Model	Hyperparameters	Value	Model	Hyperparameters	Value
Predictor	Hidden layers	(128,64,32)	Extractor	Hidden layers	(32,32)
	Loss function	MSE	Common	Activation function	ReLU
	Max epoch	20,000		Optimizer	Adam
Discriminator	Hidden layers	(128,64,32)		Learning rate	0.001
	Iteration per epoch	10		Batch	2^{12}
	Loss function	Cross entropy		k	{0, 1e−4, 5e−4, 1e−3, 1e−2, 0.1}

effectively guarantee more traffic with latency that satisfies the requirements of SLAs through intelligent scheduling, thereby improving user QoE. In this section, we introduce the methodology for incorporating the predicted latency metric in the CDN scheduling center. We begin with the optimization goal, followed by the latency metrics and the solution method for the scheduling problem.

Optimization goal. Improving the access performance (QoS) is the most important goal for the CDN scheduling center. CDN vendors typically execute Service Level Agreements (SLAs) with content providers to ensure reliable content delivery. Specifically, CDNs guarantee that the latency for most traffic stays below a pre-determined threshold for each user group. For example, considering the perceived latency $Lat^t(u,r,n)$ for a user group u from an application in region $r \in R$ when requesting node $n \in N$, we can define the latency penalty P^t function at time t as follows.

$$P^t_{un} = \begin{cases} Lat^t(u,r,n), \text{if } Lat^t(u,r,n) < Thresh_u \\ \delta * Lat^t(u,r,n), \text{if } Lat^t(u,r,n) \geq Thresh_u \end{cases} \tag{7}$$

where $Thresh_u$ denotes the corresponding latency threshold for the application and corresponding user group u, and δ (where $\delta > 1$) denotes the penalty multiplier when the perceived latency exceeds the threshold.

The introduction of the penalty multiplier δ for latencies exceeding the threshold $Thresh_u$ is crucial for both practical and mathematical reasons. Practically, $Thresh_u$ reflects the ideal latency requirement for different applications. In real-world CDN scenarios (e.g., live streaming), once the latency crosses this threshold, the user's QoE suffers a cliff-like degradation (e.g., severe video stalling). The multiplier δ acts as a heavy penalty to reflect this unacceptable QoE drop. Mathematically, without the multiplier δ, the penalty function only reflects the average global latency. Under a linear objective, swapping the scheduling decisions of two equal-volume traffic streams often results in a zero-sum exchange with no net reduction in the penalty, thereby treating a threshold-violated stream and a well-performing stream with equal priority. By introducing a piecewise penalty function via δ, we break this symmetry. It strictly forces the solver to prioritize pulling traffic below the threshold, thereby steering the objective toward maximizing the ratio of traffic meeting the ideal latency requirement rather than blindly minimizing the mean latency.

According to the given underlying latency data (predicted by ADePT), to achieve the best global QoE assurance, the scheduling problem aims to find the optimal scheduling mapping M (Eq. (1)) to minimize the global latency penalty under traffic demand T_u for all user groups u:

$$\min_{M^t} \sum_{u,n} P^t_{un} \cdot T_u \tag{8}$$

Latency metrics. Since the perceived latency $Lat^t(u,r,n)$ remains unknown until the actual scheduling decision is applied, the scheduling center requires a metric to characterize the underlying latency. As analyzed in Section 2.2, monitoring all possible paths through direct measurement is impractical. CDNs typically select the following alternative metrics:

Core segment latency from active reverse probes ①:

$$Lat^t(u,r,n) = Node^t(r,n), \quad \forall u \tag{9}$$

Full path latency from active probing ② issued by a fixed group of users. This path includes a fixed last-mile segment, depending on the probing user in the specific region, as well as a core segment that varies depending on the probed node:

$$\begin{aligned} Lat^t(u,r,n) &= f^t(User^t(u_p,r), Node^t(r,n)) \\ &= RTTProp^t_{rn}, \ \forall u \end{aligned} \tag{10}$$

Both of the two directly measured metrics overlook the impacting factor from the user side. This is because CDNs mostly focus on the core network latency. The traversed core network is selected by the scheduling of CDNs, whereas the user-side last-mile network is out of the control of CDNs. As a result, such metrics are unable to accurately reflect actual end-to-end path latency and can easily lead to a decline in QoS due to improper scheduling. For example, as shown in Fig. 4, if application 1 and application 2 are scheduled to the same node according to the latency in Eq. (10) through measurement ②, application 2 will suffer severe latency deterioration in morning commuting hours.

Solution. Theoretically, the scheduling problem can be formulated as a linear or convex optimization problem to solve [50,51]. However, the real-world scheduling problem is complicated by other complex commercial constraints and deployment limitations. As a result, CDNs typically adopt heuristic approaches to adjust the scheduling decision dynamically [35,36].

6. Evaluation

In this section, experiments are conducted based on collected data from a leading CDN vendor in China. Specifically, two key questions are examined: (1) the accuracy of ADePT in predicting end-to-end path latency for any specified path, and the contribution of its causal design (Section 6.1); (2) the optimization improvement that can be achieved by the scheduling center through more accurate latency prediction provided by ADePT (Section 6.2).

6.1. Prediction accuracy

We first validate the superior prediction accuracy that ADePT achieves on the collected measurement dataset.

Experiment setup. The latency dataset was collected in 5-minute time intervals for more than 10,000 domain names in 31 regions, covering more than 1500 nodes, over a period of one month, based on the hybrid measurement method deployed in the real world (Section 3.1). After record aggregating and decorating the sliding window of records for the extractor, the dataset is further filtered, leaving more than 150k items. Only when a scheduling decision change happens, the sample is reserved, i.e., the traffic from a specific user group switches from one node to another. This is due to the fact that it is only necessary to predict the end-to-end path latency of paths that do not carry traffic and cannot be monitored by measurement ③. Consequently, the time interval during traffic rescheduling constitutes an ideal opportunity for validating the prediction accuracy of prediction models. The data from the last 30% of days is split out as the test set, and prediction models are trained with the remaining data.

Baselines. We compare ADePT with two methods. The baseline method [16] relies on segmented measurement (Section 2.3), which

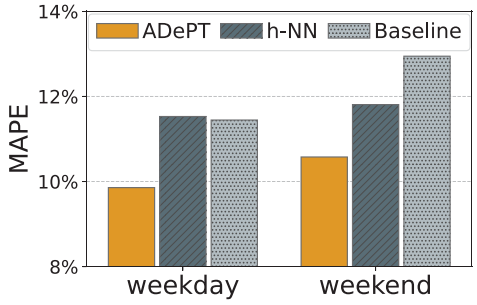


Fig. 10. MAPE among different prediction models.

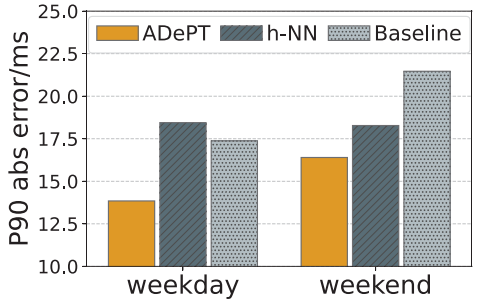


Fig. 11. 90th percentile of absolute prediction error.

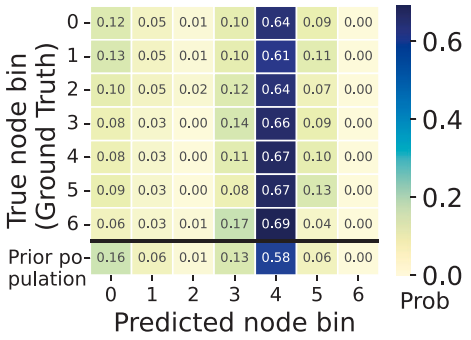


Fig. 12. Prediction distribution of the discriminator.

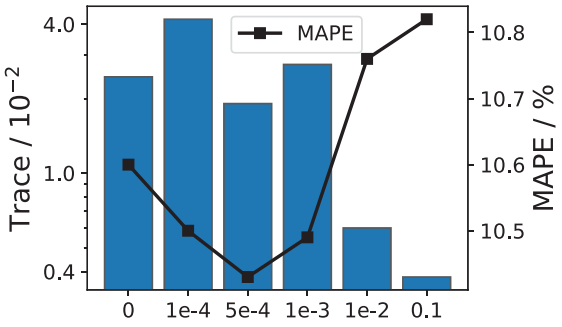


Fig. 13. Prediction accuracy under different k.

predicts the end-to-end path latency by simply summing up the last-mile segment latency of a user group and the core segment latency from the corresponding region to the node. Note that the last-mile segment latency is difficult to measure directly for CDNs. Therefore, it is calculated through Eq. (2) in Section 3.2. To evaluate the design of ADePT, we also train an integrated NN model (h-NN) to predict the end-to-end path latency directly. This hybrid NN model combines an LSTM for historical latency data and 3 fully connected layers for other features, sharing the same input as ADePT and outputting the predicted latency for the new destination node.

Prediction accuracy. ADePT achieves superior prediction accuracy compared to the baseline and the h-NN model. Figs. 10 and 11 illustrate three models' mean absolute percentage error (MAPE) and the 90th-percentile absolute prediction error, respectively. We distinguish the weekdays and weekends due to their different user-side latency patterns, as discussed in Section 3.4. The MAPE of ADePT is 10.5%, representing a 19.6% reduction compared to the baseline (13.0%) and a 9.8% decrease compared to the h-NN model (11.6%). ADePT also outperforms the other two models in the control of the tailed error (90th-percentile value), which is 22.4% and 19.4% less than baseline. The higher prediction error observed on weekends may indicate the existence of complex interactions for node-side and user-side embeddings within a specific path. In this context, ADePT exhibits superior adaptability in comparison to the baseline.

Contribution of the discriminator. The improvement in prediction accuracy from h-NN to ADePT is attributed to the design of the discriminator. The discriminator promises the extractor to extract the user-side latent embeddings, which are irrelevant to the assigned node. Recall that nodes are labeled by their corresponding observed core network latency and further classified into multiple bins (Section 4.4). Ideally, a successful extractor will completely blind the discriminator, making it impossible to infer the true assigned node bin from the extracted user-side embeddings. In such a case, to minimize the cross-entropy loss $\mathcal{L}_{disc}$, the discriminator's best strategy is to simply guess the output based on the prior probability (i.e., the overall population distribution) of the node bins in the dataset.

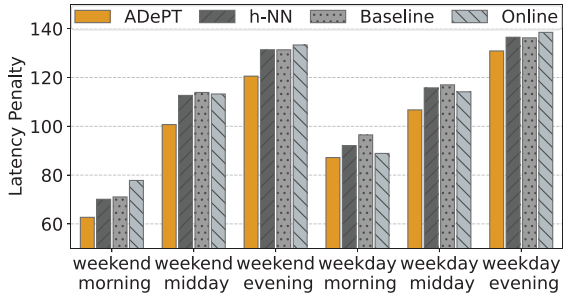
To visualize this, Fig. 12 plots the prediction distribution of the discriminator. The y-axis represents the ground-truth node bin, and the x-axis represents the discriminator's prediction. The bottom row illustrates the prior population for node bins in the whole dataset. As shown in the heatmap, there is no density concentration along the diagonal. Instead, the prediction distribution for any given ground-truth bin (each row) is nearly identical to the overall prior population (the bottom row). This explicit uniform guessing pattern firmly validates our hypothesis: the discriminator fails to distinguish the assigned nodes, proving that the extractor successfully purifies the user-side latent embeddings by stripping away any node-related information.

Setting of the parameter k. The successful extraction of user-side latent embeddings requires careful parameter selection, especially the weight k in the total loss Eq. (6). Fig. 13 shows the variation (bar, left y-axis) and the MAPE (line, right y-axis) of ADePT of the extractor's output under different k. The variation is characterized by the trace of the covariance matrix of the extractor's output. As demonstrated, the extractor trained with a lower k is incapable of extracting the latent features purely, thus failing to fool the discriminator and resembling h-NN. Conversely, the heavy punishment under high k suppresses the variation of latent embeddings, resulting in the extractor becoming useless.

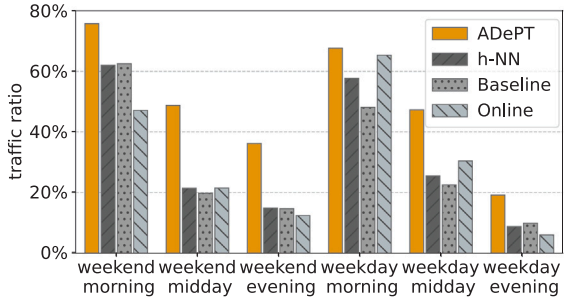
6.2. Scheduling improvement

In this part, we demonstrate that ADePT's latency prediction can improve CDN traffic scheduling decisions.

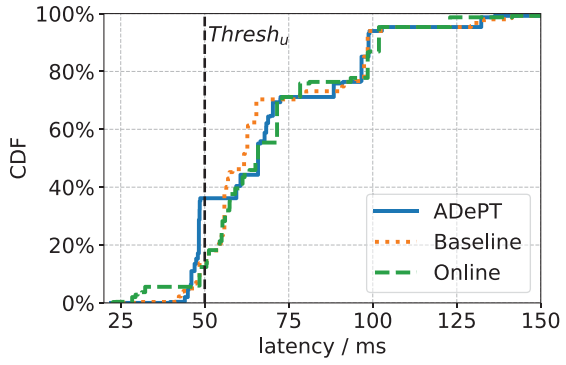
Experiment setup. We select a smaller range of 26 user groups in the same region from major video-on-demand applications in China. Traffic of these applications is easier to schedule flexibly and swiftly through HTTP-DNS [52]. Traffic demands of these user groups are also stable and higher than 100 Mbps on average over a month. To obtain the actual end-to-end path latency of any specified path, we temporarily schedule a small portion of traffic (i.e., thousands of connections) from



(a) Comparison of latency penalty.



(b) Comparison of the ratio of traffic with latency under $Thresh_u$.



(c) The CDF of requests' latency.

Fig. 14. The latency penalties, ratio of traffic with latency under $Thresh_u$ and the distribution of traffic's latency.

these user groups to a total of 62 nodes on six selected time points (morning 9 a.m., midday 1 p.m., and evening 9 p.m. on both weekends and weekdays) to perform traffic scheduling experiments, which cover major traffic demand and quality scenarios.

With the above dataset, we managed to reschedule these traffic demands to minimize the global latency penalty Eq. (8) in a simulated environment. When a user group is assigned to a specific node, the latency penalty is set according to Eq. (7) and the actual latency is collected as aforementioned. The penalty multiplier δ in Eq. (7) is set to 2, and the ideal latency requirement $Thresh_u$ is set according to the priority of the user group u (50 ms in most cases), which resembles the real-world setting. We do not minimize the global latency but the penalty because we want to serve as much traffic as possible with latency under the corresponding latency requirement $Thresh_u$. The resource supplement is limited to ensure that the low-latency resources are incapable of serving all requests. Eventually, we compare the objective score (global latency penalty) and the ratio of traffic that meets the ideal latency requirement ($Thresh_u$) for the scheduling problem when taking different sources of latency information.

To isolate the impact of latency prediction accuracy on scheduling performance, we formulate this simulated scheduling task as a standard Linear Programming (LP) problem, focusing exclusively on minimizing the global latency penalty defined in Eq. (8). We solve this LP using standard optimization solvers (e.g., SciPy [53]).

It is worth noting that designing a novel scheduling algorithm is not the focus or contribution of this paper. While production CDNs rely on complex heuristic algorithms to balance multidimensional commercial constraints, as discussed in Section 5, incorporating those factors would confound the evaluation of our prediction model. Using a pure LP simulation based solely on the latency objective, we provide a clean, controlled environment to evaluate and demonstrate that ADePT's superior prediction accuracy directly translates into better scheduling outcomes.

Baseline. We compare the performance of scheduling decisions computing based on different sources of predicted latencies from the aforementioned methods: ADePT, h-NN, and baseline. We also compare with the current online scheduling decision. Current CDNs typically select the full path latency Eq. (10) as the latency metric, which overlooks the diversity of user-side features among applications, as discussed in Section 5. Note that selecting core segment latency Eq. (9) as the latency metric results in negligible differences compared to the full path latency.

Scheduling improvements. The global latency penalties for different latency prediction methods are shown in Fig. 14(a). ADePT can effectively reduce the penalty at all time points. With ADePT, the scheduling center improves the scheduling results by 8.6% compared with the currently adopted latency prediction source, whereas the other methods result in negligible scheduling improvements due to their latency prediction errors. Specifically, the system gains more improvement in the morning when the latency among applications varies the most.

We investigate the percentage of traffic that meets the latency requirement ($Thresh_u$) by different solutions, and the results are shown in Fig. 14(b). Compared to baseline, ADePT increases the percentage of traffic with latency below $Thresh_u$ significantly from 29.6% to 49.1% (1.66×). Even on weekday evenings, when the average latency and traffic demand are highest, adopting accurately predicted latency by ADePT can promise nearly 20% more of the traffic with guaranteed latency.

Promising more portions of traffic with guaranteed latency under $Thresh_u$ sacrifices the performance of traffic with originally the lowest latency. Fig. 14(c) illustrates the distribution of the latencies of all traffic on weekend evenings. Requests from user groups with the lowest last-mile segment latency are scheduled to more distant nodes, so that more requests can be guaranteed under 50 ms of latency and avoid multiplicative penalties.

7. Discussion: Generalizability and extensions

Generalizability to other networks. ADePT is evaluated on a single, albeit highly representative, large-scale commercial edge CDN that processes 100+ Tbps traffic across 1500+ nodes. While we acknowledge the limitation of not evaluating on multiple distinct CDN architectures due to the practical unavailability of proprietary commercial datasets, the core contribution of ADePT, a causal inference methodology for decoupling coupled path segment embeddings from extremely sparse measurements, is highly generalizable. For instance, in inter-autonomous system (inter-AS) transmissions or anycast routing [19], end-to-end active probing from all ASes to all catchments is impossible. ADePT's adversarial architecture can be readily adapted to these scenarios to decouple and extract the hidden state for each AS, thereby inferring the end-to-end path latency for unseen BGP paths. **Extension to emerging satellite-assisted CDNs.** As network architectures evolve, satellite-assisted CDNs [54,55] are emerging as

a cost-effective solution to offload terrestrial core networks via multicast capabilities. In such converged architectures, satellite links are primarily utilized to feed edge caches (i.e., acting as CDN nodes). This paradigm shifts the node-side latency from distance-proportional fiber delays to satellite propagation delays (e.g., uniform delays across a wide spot-beam footprint), while the volatile user-side temporal variation (last-mile congestion) remains the critical QoS bottleneck.

The decoupled, causal nature of ADePT is perfectly suited for this architectural evolution. Because ADePT extracts node-invariant user-side embeddings, it can seamlessly evaluate these new satellite-assisted nodes. By simply introducing the satellite's specific propagation characteristics as new node-side features, ADePT can accurately predict the end-to-end path latency for these hybrid paths, aiding the scheduling center in dynamically balancing traffic between purely terrestrial nodes and satellite-assisted nodes.

8. Related work

CDN measurement methodologies. CDNs have proposed various measurement methodologies to evaluate or improve the performance of CDNs. By collecting TCP connection states from the TCP stack, passive measurements are used for event localization in CPR [31,33]. Active measurements issue probes periodically either from the user end device (e.g., Odin [9]) or home routers (e.g., BISmark [26]) to monitor the quality of networks. Proactive measurements are used in EdgeFabric [35] and Espresso [36] by directing a small proportion of user traffic to alternative paths or in [30] by directing DNS queries to different servers. A hybrid method of measurement is used in Blameit [11,56] for troubleshooting and rerouting. These methods cannot support the large-scale latency data that an edge CDN needs.

Latency prediction. Traditional end-to-end path latency prediction methods include matrix completion [21,24], global network positioning [12,13,15] and segmented measurement [16,19,20]. In recent years, data-driven causal models [57–60] have been adopted for the purpose of predicting microservice latency and troubleshooting. This work focuses on predicting the end-to-end network transmission latency for specified paths.

CDN Scheduling Strategy. CDN scheduling can be formulated as the path selection problem [10,35,36,50,61] or the node placement problem [51]. These scheduling methods are incapable of obtaining optimized solutions without accurate end-to-end path latency data.

9. Conclusion

In this paper, we propose ADePT, a novel data-driven causal inference framework that accurately predicts end-to-end path latency. ADePT is capable of extracting high-dimensional latent embeddings of user and node characteristics from sparse measurement datasets and of learning the complex non-linear correlations of embeddings. Our comprehensive evaluation on a real-world dataset demonstrated that ADePT significantly surpasses existing latency prediction approaches. ADePT reduced prediction errors by 19.6% and achieved a highly accurate median absolute error of 4.3 ms. The practical impact of this improved accuracy was further validated by showing that ADePT can improve the percentage of traffic meeting ideal latency requirements (the SLA threshold) by 1.66×, proving its value in real-world traffic scheduling.

CRediT authorship contribution statement

Chuanqing Lin: Writing – original draft, Visualization, Validation, Methodology, Investigation. **Gerui Lv:** Writing – review & editing, Conceptualization. **Yangguang Liang:** Resources. **Fuhua Zeng:** Resources. **Xiaodong Li:** Resources. **Qinghua Wu:** Writing – review & editing, Conceptualization. **Zhenyu Li:** Writing – review & editing, Funding acquisition. **Gaogang Xie:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We sincerely thank all anonymous reviewers for their constructive feedback. This work was supported in part by the National Key R&D Program of China (2022YFB2901800) and the Postdoctoral Fellowship Program of China Postdoctoral Science Foundation (Grant No. GZC20251071).

Data availability

The authors do not have permission to share data.

References

[1] S. Dhawaskar Sathyanarayana, K. Lee, D. Grunwald, S. Ha, Converge: QoE-driven multipath video conferencing over WebRTC, in: Proceedings of the ACM SIGCOMM 2023 Conference, in: ACM SIGCOMM '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 637–653, http://dx.doi.org/10.1145/3603269.3604822.

[2] B. Wu, K. Qian, B. Li, Y. Ma, Q. Zhang, Z. Jiang, J. Zhao, D. Cai, E. Zhai, X. Liu, X. Jin, XRON: A hybrid elastic cloud overlay network for video conferencing at planetary scale, in: Proceedings of the ACM SIGCOMM 2023 Conference, in: ACM SIGCOMM '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 696–709, http://dx.doi.org/10.1145/3603269.3604845.

[3] H. Zhang, C. An, A. Zhou, Y. Zhu, W. Sun, Y. Lu, J. Chen, L. Liu, H. Ma, A. Fei, Venus: Enhancing QoE of crowdsourced live video streaming by exploiting multiflow viewer assistance, in: Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, 2024, pp. 170–184.

[4] R.-X. Zhang, H. Wang, S. Shi, X. Pang, Y. Peng, Z. Xue, J. Liu, Enhancing resource management of the world's largest PCDN system for On-Demand video streaming, in: 2024 USENIX Annual Technical Conference, USENIX ATC 24, USENIX Association, Santa Clara, CA, 2024, pp. 951–965, URL: https://www.usenix.org/conference/atc24/presentation/zhang-rui-xiao.

[5] Y. Zhou, T. Wang, L. Wang, N. Wen, R. Han, J. Wang, C. Wu, J. Chen, L. Jiang, S. Wang, H. Liu, C. Xu, AUGUR: Practical mobile multipath transport service for low tail latency in real-time streaming, in: 21st USENIX Symposium on Networked Systems Design and Implementation, NSDI 24, USENIX Association, Santa Clara, CA, 2024, pp. 1901–1916, URL: https://www.usenix.org/conference/nsdi24/presentation/zhou-yuhan.

[6] C. Huang, A. Wang, J. Li, K.W. Ross, Measuring and evaluating large-scale CDNs, in: ACM IMC, vol. 8, 2008, pp. 15–29.

[7] T.K. Dang, N. Mohan, L. Corneo, A. Zavodovski, J. Ott, J. Kangasharju, Cloudy with a chance of short RTTs: analyzing cloud connectivity in the internet, in: Proceedings of the 21st ACM Internet Measurement Conference, 2021, pp. 62–79.

[8] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, X. Liu, From cloud to edge: a first look at public edge platforms, in: Proceedings of the 21st ACM Internet Measurement Conference, 2021, pp. 37–53.

[9] M. Calder, R. Gao, M. Schröder, R. Stewart, J. Padhye, R. Mahajan, G. Ananthanarayanan, E. Katz-Bassett, Odin: Microsoft's scalable Fault-Tolerant CDN measurement system, in: 15th USENIX NSDI 18, 2018, pp. 501–517.

[10] F. Chen, R.K. Sitaraman, M. Torres, End-user mapping: Next generation request routing for content delivery, ACM SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 167–181.

[11] B. Schlinker, I. Cunha, Y.-C. Chiu, S. Sundaresan, E. Katz-Bassett, Internet performance from facebook's edge, in: Proceedings of the Internet Measurement Conference, 2019, pp. 179–194.

[12] F. Dabek, R. Cox, F. Kaashoek, R. Morris, Vivaldi: A decentralized network coordinate system, ACM SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 15–26.

[13] M. Szymaniak, D. Presotto, G. Pierre, M. Van Steen, Practical large-scale latency estimation, Comput. Netw. 52 (7) (2008) 1343–1364.

[14] P.R. Pietzuch, J. Ledlie, M.I. Seltzer, Supporting network coordinates on PlanetLab, in: WORLDS, vol. 5, 2005, pp. 19–24.

[15] T.E. Ng, H. Zhang, Predicting internet network distance with coordinates-based approaches, in: Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, IEEE, 2002, pp. 170–179.

[16] H.V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, A. Venkataramani, iPlane: An information plane for distributed services, in: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, 2006, pp. 367–380.

[17] H.V. Madhyastha, E. Katz-Bassett, T.E. Anderson, A. Krishnamurthy, A. Venkataramani, Iplane nano: Path prediction for peer-to-peer applications, in: NSDI, vol. 9, 2009, pp. 137–152.

[18] J. Ledlie, P. Gardner, M.I. Seltzer, Network coordinates in the wild, in: NSDI, vol. 7, 2007, pp. 299–311.

[19] X. Zhang, T. Sen, Z. Zhang, T. April, B. Chandrasekaran, D. Choffnes, B.M. Maggs, H. Shen, R.K. Sitaraman, X. Yang, Anyopt: Predicting and optimizing ip anycast performance, in: Proceedings of the 2021 ACM SIGCOMM 2021 Conference, 2021, pp. 447–462.

[20] H.V. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, A. Venkataramani, A structural approach to latency prediction, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, 2006, pp. 99–104.

[21] S. Athey, M. Bayati, N. Doudchenko, G. Imbens, K. Khosravi, Matrix completion methods for causal pnel data models, J. Amer. Statist. Assoc. 116 (536) (2021) 1716–1730.

[22] A. Lakhina, M. Crovella, C. Diot, Diagnosing network-wide traffic anomalies, ACM SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 219–230.

[23] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E.D. Kolaczyk, N. Taft, Structural analysis of network traffic flows, in: Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, 2004, pp. 61–72.

[24] M. Roughan, Y. Zhang, W. Willinger, L. Qiu, Spatio-temporal compressive sensing and internet traffic matrices (extended version), IEEE/ACM Trans. Netw. 20 (3) (2011) 662–676.

[25] J. Li, Z. Li, R. Lu, K. Xiao, S. Li, J. Chen, J. Yang, C. Zong, A. Chen, Q. Wu, et al., Livenet: a low-latency video transport network for large-scale live streaming, in: Proceedings of the ACM SIGCOMM 2022 Conference, 2022, pp. 812–825.

[26] S. Sundaresan, S. Burnett, N. Feamster, W. De Donato, BISmark: A testbed for deploying measurements and applications in broadband access networks, in: 2014 USENIX Annual Technical Conference, USENIX ATC 14, 2014, pp. 383–394.

[27] P. Goenka, K. Zarifis, A. Gupta, M. Calder, Towards client-side active measurements without application control, ACM SIGCOMM Comput. Commun. Rev. 52 (1) (2022) 20–27.

[28] X. Zhang, G. Min, Q. Fan, H. Yin, D.O. Wu, Z. Ma, A light-weight statistical latency measurement platform at scale, IEEE Trans. Ind. Inform. 18 (2) (2021) 1186–1196.

[29] K.P. Gummadi, S. Saroiu, S.D. Gribble, King: Estimating latency between arbitrary internet end hosts, in: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment, 2002, pp. 5–18.

[30] C. Huang, N. Holt, Y.A. Wang, A. Greenberg, J. Li, K.W. Ross, A DNS reflection method for global traffic management, in: 2010 USENIX Annual Technical Conference, USENIX ATC 10, 2010.

[31] R. Landa, L. Saino, L. Buytenhek, J.T. Araújo, Staying alive: Connection path reselection at the edge, in: 18th USENIX NSDI 21, 2021, pp. 233–251.

[32] Y. Zhu, B. Helsley, J. Rexford, A. Siganporia, S. Srinivasan, LatLong: Diagnosing wide-area latency changes for CDNs, IEEE Trans. Netw. Serv. Manag. 9 (3) (2012) 333–345.

[33] P. Sun, M. Yu, M.J. Freedman, J. Rexford, Identifying performance bottlenecks in CDNs through TCP-level monitoring, in: Proceedings of the First ACM SIGCOMM Workshop on Measurements Up the Stack, 2011, pp. 49–54.

[34] C. Alvarez, K. Argyraki, Using gaming footage as a source of internet latency information, in: Proceedings of the 2023 ACM on Internet Measurement Conference, 2023, pp. 606–626.

[35] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H.V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, H. Zeng, Engineering egress with edge fabric: Steering oceans of content to the world, in: Proceedings of the Conference of the ACM SIGCOMM, 2017, pp. 418–431.

[36] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, et al., Taking the edge off with espresso: Scale, reliability and programmability for global internet peering, in: Proceedings of the Conference of the ACM SIGCOMM, 2017, pp. 432–445.

[37] R. Krishnan, H.V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, J. Gao, Moving beyond end-to-end path information to optimize CDN performance, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, 2009, pp. 190–201.

[38] K. Vermeulen, E. Gurmericliler, I. Cunha, D. Choffnes, E. Katz-Bassett, Internet scale reverse traceroute, in: Proceedings of the 22nd ACM Internet Measurement Conference, 2022, pp. 694–715.

[39] S. Sundaresan, N. Feamster, R. Teixeira, Home network or access link? Locating last-mile downstream throughput bottlenecks, in: International Conference on Passive and Active Network Measurement, Springer, 2016, pp. 111–123.

[40] R. Fontugne, A. Shah, K. Cho, Persistent last-mile congestion: not so uncommon, in: Proceedings of the ACM Internet Measurement Conference, 2020, pp. 420–427.

[41] Z. Xu, The minimal measurement number for low-rank matrix recovery, Appl. Comput. Harmon. Anal. 44 (2) (2018) 497–508.

[42] R. Fontugne, C. Pelsser, E. Aben, R. Bush, Pinpointing delay and forwarding anomalies using large-scale traceroute measurements, in: Proceedings of the 2017 Internet Measurement Conference, 2017, pp. 15–28.

[43] A. Dhamdhere, D.D. Clark, A. Gamero-Garrido, M. Luckie, R.K. Mok, G. Akiwate, K. Gogia, V. Bajpai, A.C. Snoeren, K. Claffy, Inferring persistent interdomain congestion, in: Proceedings of the 2018 Conference of the ACM SIGCOMM, 2018, pp. 1–15.

[44] S.D. Strowes, Passively measuring TCP round-trip times, Commun. ACM 56 (10) (2013) 57–64.

[45] C. Pelsser, L. Cittadini, S. Vissicchio, R. Bush, From Paris to Tokyo: On the suitability of ping to measure latency, in: Proceedings of the 2013 Conference on Internet Measurement Conference, 2013, pp. 427–432.

[46] H. Abdi, The Kendall rank correlation coefficient, Encycl. Meas. Stat. 2 (2007) 508–510.

[47] V. Bajpai, S.J. Eravuchira, J. Schönwälder, Dissecting last-mile latency characteristics, ACM SIGCOMM Comput. Commun. Rev. 47 (5) (2017) 25–34.

[48] X. Zeng, H. Xu, C. Chen, X. Zhang, X. Zhang, X. Chen, G. Chen, Y. Qiu, Y. Zhang, C. Hao, et al., Learning production-optimized congestion control selection for alibaba cloud CDN, in: 22nd USENIX Symposium on Networked Systems Design and Implementation, NSDI 25, 2025, pp. 827–845.

[49] A. Alomar, P. Hamadanian, A. Nasr-Esfahany, A. Agarwal, M. Alizadeh, D. Shah, CausalSim: A causal framework for unbiased trace-driven simulation, in: 20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 23, USENIX Association, Boston, MA, 2023, pp. 1115–1147.

[50] Z. Zhang, M. Zhang, A.G. Greenberg, Y.C. Hu, R. Mahajan, B. Christian, Optimizing cost and performance in online service provider networks, in: NSDI, 2010, pp. 33–48.

[51] H.H. Liu, R. Viswanathan, M. Calder, A. Akella, R. Mahajan, J. Padhye, M. Zhang, Efficiently delivering online services over integrated infrastructure, in: 13th USENIX NSDI 16, 2016, pp. 77–90.

[52] P.E. Hoffman, P. McManus, DNS Queries over HTTPS (DoH), 2018, http://dx.doi.org/10.17487/RFC8484, URL: https://www.rfc-editor.org/info/rfc8484. RFC 8484.

[53] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in python, Nature Methods 17 (2020) 261–272.

[54] M. Luglio, S.P. Romano, C. Roseti, F. Zampognaro, Service delivery models for converged satellite-terrestrial 5G network deployment: A satellite-assisted CDN use-case, IEEE Netw. 33 (1) (2019) 142–150, http://dx.doi.org/10.1109/MNET.2018.1800020.

[55] M. Luglio, S.P. Romano, C. Roseti, F. Zampognaro, Satellite multi-beam multicast support for an efficient community-based CDN, Comput. Netw. 217 (2022) 109352, URL: https://api.semanticscholar.org/CorpusID:252140108.

[56] Y. Jin, S. Renganathan, G. Ananthanarayanan, J. Jiang, V.N. Padmanabhan, M. Schroder, M. Calder, A. Krishnamurthy, Zooming in on wide-area latencies to a global cloud provider, in: Proceedings of the ACM SIGCOMM, 2019, pp. 104–116.

[57] S. Huang, Y. Wei, L. Peng, M. Wang, L. Hui, P. Liu, Z. Du, Z. Liu, Y. Cui, xNet: Modeling network performance with graph neural networks, IEEE/ACM Trans. Netw. 32 (2) (2024) 1753–1767, http://dx.doi.org/10.1109/TNET.2023.3329357.

[58] Y. Zhang, R. Isaacs, Y. Yue, J. Yang, L. Zhang, Y. Vigfusson, LatenSeer: Causal modeling of end-to-end latency distributions by harnessing distributed tracing, in: Proceedings of the 2023 ACM Symposium on Cloud Computing, SoCC '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 502–519, http://dx.doi.org/10.1145/3620678.3624787.

[59] Y. Jiang, L.R. Sivalingam, S. Nath, R. Govindan, WebPerf: Evaluating "What-If" Scenarios for Cloud-hosted Web Applications, Technical Report MSR-TR-2016-16, 2016.

[60] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, M. Ammar, Answering what-if deployment and configuration questions with wise, in: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication, 2008, pp. 99–110.

[61] V. Valancius, B. Ravi, N. Feamster, A.C. Snoeren, Quantifying the benefits of joint content and network routing, in: Proceedings of the ACM SIGMETRICS, 2013, pp. 243–254.