Cost-efficient request mapping for large-scale live streaming services

YU TIAN[†§], ZHENYU LI[†§‡], MATTHEW YANG LIU[¶], QINGHUA WU[†§‡], ZHAOXUE ZHONG[¶], AO LI[¶], JIAXING ZHANG[†§], GERUI LV[†§], CHUANQING LIN[†§], XI WANG[¶], JIAN MAO[¶], GARETH TYSON[‖], JIE XIONG[£], ZHENHUA LI[•], GAOGANG XIE[✠§],

[†]Institute of Computing Technology, Chinese Academy of Sciences [¶]Bilibili [•] Tsinghua University
[£]Nanyang Technological University [‡]Purple Mountain Laboratories [✠] CNIC, CAS
[‖]Hong Kong University of Science and Technology (GZ) [§]University of Chinese Academy of Sciences

The live streaming landscape has shifted to a crowd-sourced paradigm, resulting in highly volatile and geographically diverse viewer demand. To handle growing traffic, Content Delivery Networks (CDNs) increasingly rely on a mix of dedicated infrastructure and lower-cost, heterogeneous edge resources. Our analysis of production data reveals two emerging characteristics in modern live delivery: dynamic regional supply-demand imbalance and per-stream heterogeneity in popularity and geography. However, existing request mapping solutions fall short in this new landscape, as they assume stable regional capacity and overlook stream-level heterogeneity. This paper proposes LiveMap, a cost-efficient and latency-aware request mapping system for live CDNs. LiveMap performs online bandwidth provisioning via dual-level coordination to resolve regional supply-demand imbalances and reduce bandwidth costs. Further, LiveMap introduces an adaptive stream mapping strategy that dynamically forms per-stream delivery groups based on real-time popularity and system load. Deployed in Bilibili CDN serving crowdsourced live streaming over a year, LiveMap reduces bandwidth costs by 42.18% and access latency by 20.26%, outperforming the state-of-the-art solutions.

CCS Concepts: • **Networks** → **Network design and planning algorithms**; **Network services**.

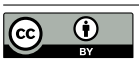
Additional Key Words and Phrases: Live streaming; CDN; Cost-efficient

ACM Reference Format:
Yu Tian[†§], Zhenyu Li[†§‡], Matthew Yang Liu[¶], Qinghua Wu[†§‡], Zhaoxue Zhong[¶], Ao Li[¶], Jiaxing Zhang[†§], Gerui Lv[†§], Chuanqing Lin[†§], Xi Wang[¶], Jian Mao[¶], Gareth Tyson[‖], Jie Xiong[£], Zhenhua Li[•], Gaogang Xie[✠§]. 2025. Cost-efficient request mapping for large-scale live streaming services. *Proc. ACM Netw.* 3, CoNEXT4, Article V3net031 (December 2025), 26 pages. https://doi.org/10.1145/3768978

1 Introduction

Live streaming services have undergone a paradigm shift from single-source broadcasts (e.g., live sports events) to crowdsourced content production, enabling any user to stream to a global audience [35, 48]. Modern platforms such as Twitch and TikTok incorporate algorithmic content recommendations, which can rapidly redirect user attention across streams. As a result, viewer traffic often exhibits sudden bursts and volatile patterns that are difficult to predict [11, 65, 68].

Despite these user-side changes, Content Delivery Networks (CDNs) remain the backbone of large-scale live content delivery [33, 34, 44]. To keep pace with increasing traffic, CDNs are evolving from using only dedicated infrastructure to also leveraging cheaper but less stable edge resources [54, 66]. At the same time, the interactive nature of live streaming imposes stricter latency requirements, while growing bandwidth costs push CDNs to focus more on cost efficiency.



ACM 2834-5509/2025/12-ARTV3net031
https://doi.org/10.1145/3768978

A key component in the CDN control plane is *request mapping*, which determines how edge nodes are assigned to serve individual viewer requests. Traditionally, request mapping can be formulated as a supply-demand matching problem between demand of user and edge node capacity. Recent studies have been focus on bandwidth cost optimization [24, 49, 50, 70], where the cost is computed as the product of each node's billable bandwidth and its unit price. Here, billable bandwidth refers to the charging threshold defined by pricing models such as the 95th percentile, which excludes a small fraction (*e.g.*, 5%) of the highest-usage intervals in a billing cycle.

We evaluated several state-of-the-art bandwidth optimization solutions using three-month traces from Bilibili live CDN [7]. However, their performance proved far from ideal: bandwidth cost was over 40% higher, and latency exceeded 45% on average—reaching up to 72% in regions with severe bandwidth shortages. Our further analysis identifies two key characteristics of modern live streaming workloads that underlie these inefficiencies. *First*, bandwidth supply and demand are highly dynamic and imbalanced across network regions. On the supply side, nearly half of the edge nodes exhibit substantial fluctuations in available capacity. On the demand side, content recommendation algorithms, which is opaque to the request mapping logic, can trigger sudden and unpredictable traffic spikes. *Second*, live streams exhibit highly uneven and diverse popularity trends across both time and geography. Even within the same time window, the viewership of thousands of concurrent streams varies significantly across regions. Unfortunately, existing bandwidth optimization solutions fall short under these new characteristics. They assume stable and sufficient capacity within each region and rely on static planning to configure billable bandwidth. They also overlook stream-level and user-group-level popularity differences, using coarse, precomputed mappings that cannot adapt to regional imbalances or real-time shifts in demand.

To fill these gaps, we propose two key insights for building a cost-efficient request mapping system in live streaming CDNs: (*i*) Dynamically reassign traffic from overloaded regions to those with spare bandwidth to mitigate regional imbalances. (*ii*) Schedule viewer requests with stream-level awareness to account for popularity and distribution differences. These insights motivate the design of LiveMap, a request mapping system that is both cost-efficient and latency-aware. Yet, implementing LiveMap at scale entails several practical challenges:

First, cross-region coordination must respect latency constraints with the need to reallocate bandwidth across regions under dynamic and uneven demand. Intuitive methods that let each region independently borrow from nearby regions can lead to contention, where early borrowers exhaust capacity, forcing others to fall back on distant options and resulting in suboptimal latency.

Second, stream-aware mapping introduces a fundamental tradeoff between minimizing latency and maximizing distribution efficiency. Prioritizing proximity lead to inefficient resource utilization in large-scale CDNs hosting massive (~10k) concurrent streams, as it may cause unpopular streams to be distributed across multiple edge nodes. Balancing latency and distribution efficiency, while accounting for the spatial heterogeneity of stream popularity, is a non-trivial challenge.

Third, billing heterogeneity further complicates cost optimization within a region. Edge nodes may differ in billing granularity (e.g., monthly vs. sub-monthly) and pricing models (*e.g.*, 95th-percentile vs. fixed-rate) [70]. This heterogeneity complicates billable bandwidth configuration within each region to achieve cost-aware provisioning.

To address these challenges, LiveMap is composed of two major components: a **dual-level coordinated bandwidth provision** component responsible for configuring bandwidth budget online (§4) and an **adaptive stream mapping** component responsible for dynamically assigning edge nodes to streams (§5). The bandwidth provision component optimizes bandwidth allocation across regions (region-level coordination) to mitigate supply-demand imbalances, and it adjusts node provisioning (node-level coordination) according to heterogeneous billing constraints and real-time usage. The stream mapping component dynamically forms per-stream delivery groups

that adapt to global bandwidth load and per-stream popularity distribution, and updates node assignment to delivery groups accordingly.

We confirm LiveMap's superiority over state-of-the-art solutions through trace-driven simulations and deploy it in Bilibili CDN since September 2023, supporting live streaming services for tens of millions of daily users. Results show that LiveMap saves the bandwidth costs by up to 42.18% (equal to millions of dollars annually), and decreases access latency by 20.26%. These results demonstrate that LiveMap has successfully tamed the supply and demand dynamics and stream diversity for cost-efficient live content delivery. In summary, our contributions are as follows:

- We identify unique characteristics of modern live streaming services that challenge request mapping based on real-world data from Bilibili CDN.
- We design LiveMap, a cost-efficient request mapping system for live streaming CDNs. LiveMap optimizes the bandwidth cost with proximity awareness for the CDNs that involve less stable edge nodes to support crowdsourced live streaming services.
- We propose solutions to address the challenges arising from dynamic supply and demand, and diverse stream behavior, including online bandwidth provision with dual-level coordination, and dynamic stream-level mapping through popularity- and load-driven adaptation.
- We validate the effectiveness of LiveMap through trace-driven simulations and large-scale deployment over 1 year, highlighting significant cost savings and a reduction in access latency.

2 Background and Motivation

2.1 Request Mapping in Live Streaming CDNs

Modern Live CDNs with Heterogeneous Edge Nodes. Live streaming has evolved to crowdsourced content production, resulting in substantial and increasingly volatile traffic. While the underlying CDN architecture remains largely unchanged [33, 34, 44], typically consisting of a central streaming center and a set of geographically distributed CDN nodes, modern CDNs are increasingly integrating low-cost heterogeneous edge resources to meet scaling demands and manage bandwidth costs [20, 22, 54, 56, 60, 66] . These resources typically consist of low-cost, ISP-owned resources that leverage underutilized bandwidth, rather than traditional, fully managed CDN infrastructure.[1] While cost-effective and widely distributed, they tend to exhibit higher volatility due to factors such as partial hardware failures (*e.g.*, individual NICs going offline), transient performance degradation, or limited service guarantees. These CDN nodes are usually organized in a two-layer hierarchy: viewers pull content from *edge nodes* (L1), which are deployed near end users and, in turn, pull streams from upstream *reflectors* (L2) that offer higher bandwidth and storage capacity.

Bandwidth Cost. CDNs incur bandwidth costs by paying Internet Service Providers (ISPs) for outbound traffic generated at CDN nodes. These costs are typically calculated on a per-node basis according to two common billing mechanisms [70]: (*i*) *Usage-based Billing*, where the billable bandwidth represents the charging threshold derived from bandwidth usage over a billing cycle (typically composed of fixed-length intervals, *e.g.*, every 5 minutes [57]). Under different pricing models, billable bandwidth is computed differently: in percentile-based billing, it corresponds to the 95th-percentile usage across all intervals; in average-based billing, it equals the mean bandwidth usage over the cycle. (*ii*) *Fixed billing*, where a predetermined bandwidth cap is assigned to each node, and CDN vendors are charged based on this limit regardless of actual usage. In the rest of this paper, we also refer to this fixed limit as the billable bandwidth, with the distinction that it is a known constant rather than a usage-derived value.

[1] In our system, these low-cost edge nodes are ISP-owned and do not involve user devices, so there are no privacy or ethical concerns.

In both models, the bandwidth cost of a node is calculated as the product of its billable bandwidth and the unit price charged by the ISP. Note that the unit price may vary across nodes. As live streaming traffic continues to grow with the increasing number of broadcasters and viewers, bandwidth cost has become a dominant operational concern, amounting to millions of dollars per year in large-scale CDNs.

Request Mapping. The bandwidth usage of a CDN node (either edge node or reflector) ultimately depends on request mapping, which determines which edge node serve each live stream. To reduce the overhead of mapping individual clients, users in the same autonomous system (AS) and metropolitan area are usually grouped into one user group[2], as in [28, 32]. Thus, the output of request mapping specifies, for each `(stream, user group)` pair, a list of edge nodes assigned to serve its requests during the current configuration cycle.

This mapping can be configured by one of the following approaches. *Proximity-first mapping* selects edge nodes in the same network region, where a network region refers to a group of edge nodes that reside within the same AS and geographical metropolitan area (*e.g.*, a province or a cluster of adjacent provinces), to minimize the latency between the users and edge nodes, but may concentrate traffic on geographically preferred nodes, leading to bandwidth spikes at high-demand nodes and increasing overall bandwidth costs. In contrast, *Cost-first mapping* prioritizes low-cost edge nodes with spare bandwidth [69], ignoring user proximity, which can inflate the latency. To strike a balance, *Planning-based mapping* formulates mapping as an optimization task, taking the CDN nodes' information, the requests information, and network latency as inputs. Formally, the problem can be summarized as:

$$\min\{\sum_{k\in N} cost_k + \beta * \sum_{s\in S}\sum_{v\in V(s)} lat_v\}, \tag{1}$$

where N is the set of CDN nodes, S is the set of streams and $V(s)$ is the set of viewing sessions of stream s. $cost_i$ is the bandwidth cost of node i, lat_v is the latency between the viewer and the edge node of the session v, and β represents how much we weigh the latency. Setting β to the maximum leads to the proximity-first mapping, while setting β to 0 equals the cost-first mapping.

2.2 Live Streaming Service Characteristics

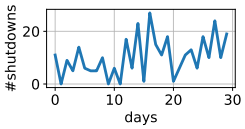
To investigate emerging characteristics of modern live streaming services, we collect a comprehensive dataset from the production Bilibili CDN, comprising request-level, stream-level, node-level, and user-group–level records. To the best of our knowledge, these characteristics have not been systematically studied in prior work.

The request-level dataset contains 4.86 billion requests collected over three months (April—June 2023). Each record includes the request timestamp, stream ID, bitrate, user group, the actual edge node that served the request, the region of that node, and the RTT from client to node. These attributes enable both latency estimation and faithful replay in simulation. Requests to streams with fewer than 10 viewers are excluded, both to reduce computational overhead and to align with real-world deployment practices, where such "cold" streams are directly served by L2 nodes without traversing LiveMap's two-layer architecture.

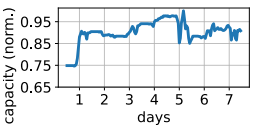
The stream-level dataset captures stream-level information from July 2023, including each stream's start and end times and per-minute viewer counts across regions. It is used to analyze stream dynamics and concurrency patterns, with the peak number of concurrent live streams during this period reaching 52.13k.

The node-level dataset collected in July 2023 records each node's region, billing mode, unit bandwidth cost, dynamic bandwidth capacity (which may fluctuate), and bandwidth usage at

[2]In practice, users are grouped based on their IP prefix.

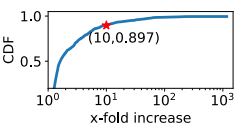


(a) Number of shutdowns.

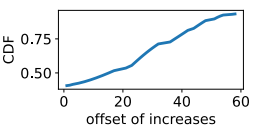


(b) Capacity variations.

Fig. 1. Dynamic node supply (entire CDN).



(a) Bursty increases.



(b) Bursty increase timing.

Fig. 2. Extreme demand dynamics.

5-minute granularity. This dataset forms the basis of our simulation evaluations, providing inputs for the baseline approaches and data for bandwidth cost estimation. It is also used to compute the temporal evolution of regional supply volume. Meanwhile, the user-group–level dataset provides per-5-minute traffic demand for each user group throughout July 2023, supporting the calculation of regional demand dynamics.

We further extract daily shutdown events from device logs across all edge nodes to characterize the volatility of heterogeneous resources. A "shutdown" refers to a node being temporarily excluded from request serving due to degraded QoS (*e.g.*, high CPU usage). These nodes are not permanently removed and may rejoin once their quality recovers. To ensure robustness, our system maintains a pool of backup resources; when a node shuts down, spare nodes are promoted to replace it.[3]

Unstable Bandwidth Supply. Unlike traditional CDNs with stable provisioning [44], modern platforms increasingly depend on heterogeneous edge resources [66], leading to volatile capacity. As shown in Figure 1(a), dozens of nodes are excluded from service daily due to QoS (quality of service) degradation. Notably, all excluded nodes belong to the pool of heterogeneous low-cost edge nodes, while none of the dedicated CDN nodes experienced any shutdown events during the observation period. Their bandwidth provision also fluctuates significantly, *i.e.*, by as much as 2—25% over a week (as shown in Figure 1(b)). This volatility stems from the heterogeneous and dynamic nature of the low-cost edge nodes[54], which often consist of multiple physical servers or network interfaces grouped under a single logical node ID. As a result, even partial hardware issues (*e.g.*, one NIC or server going offline) can cause noticeable capacity fluctuations.

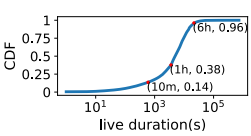
Extremely Dynamic Demand. We analyze the temporal variations of demands by identifying bursty increases, defined as a >10× increase in viewership within one minute. Figure 2(a) shows that 10.3% of the streams that attract over 100 viewers, experience at least one such bursty increase. Additionally, these bursts can occur at any point during the stream's lifetime, as illustrated in Figure 2(b), which plots the distribution of their time offsets relative to the stream start time. These surges are often triggered by algorithmic recommendations that suddenly redirect user attention, or by real-time viewer–broadcaster interactions, both of which are opaque to traditional request-mapping systems. Compared to previously studied dynamics–driven by unpredictable popular streams and short viewing durations [37]–this form of burstiness introduces even greater volatility and unpredictability into live streaming traffic patterns.

Observation 1: **Burst demand and unreliable edge nodes lead to dynamic regional imbalances between bandwidth supply and demand.**

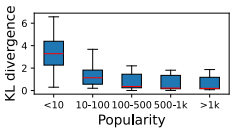
Table 1. The ratios of supply to demand across regions (red: supply < demand; green: supply > demand).

Regions	R1	R2	R3	R4	R5	R6	R7
Ratio	2.75	**0.40**	2.71	2.51	**0.91**	1.62	1.47

[3]The bandwidth provisioning is computed based on the resource pool excluding backup nodes. The activation of backup nodes in response to unexpected churn may cause deviations between planned and actual bandwidth usage, motivating the need for online adjustments and coordinated updates as discussed in § 4.

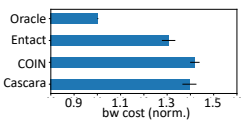


(a) Live duration.

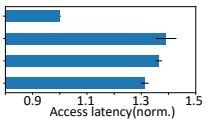


(b) Geo-popularity.

Fig. 3. Heterogeneous stream behavior.



(a) Bandwidth Cost.



(b) Mean access latency.

Fig. 4. Performance compared with Oracle.

Table 1 presents the supply-demand ratio of each region. Here, supply refers to the aggregate bandwidth capacity of nodes within a region, while demand denotes the actual bandwidth consumed by clients over the same measurement window, aggregated at a 5-minute interval. At the global level, we ensure that total supply exceeds total demand to guarantee all user requests are served. However, regional mismatches may still arise. For example, while some regions provide more than twice the capacity needed, others (*e.g.*, R2) suffer from significant shortages, with their excess demand implicitly satisfied by spare capacity in other regions.

Diverse Stream Behaviors. Live streams vary not only in popularity (*i.e.*, the total number of concurrent viewers) but also in fundamental characteristics such as lifespan and geographic demand patterns, driven by content diversity and audience behavior. As shown in Figure 3(a), 14% of streams last less than 10 minutes, while over 62% persist for more than an hour. This wide range reflects a mix of transient streams (*e.g.*, quick vlogs or test broadcasts) and long-running sessions (*e.g.*, gaming or event coverage). These temporal variations affect how delivery resources should be provisioned and maintained over time. Moreover, we quantify the geo-popularity distribution by computing the KL divergence between each stream's viewer distribution and the global viewer distribution across all streams (Figure 3(b)). Higher KL values indicate stronger deviation from the global pattern, *i.e.*, more geographically concentrated interest. While popular streams tend to follow the global distribution more closely, many high-traffic streams exhibit strong regional skew.

Observation 2: **The popularity of live video streams is not only skewed, but also regionally and temporally diverse.**

These per-stream differences, especially in geographic demand and lifespan, highlight the limitations of uniform request mapping strategies. Efficient delivery requires stream-aware mapping that adapts to per-stream locality and duration, ensuring low latency and cost-efficiency without overprovisioning.

2.3 Taming the Imbalances and Diversity

Pitfalls of Prior Arts. We apply three representative bandwidth optimization solutions in live streaming CDNs and evaluate them with 3-month real-world request logs from a large live streaming CDN (see Section 6.1 for trace details). These methods adopt different planning models: Entact [69] applies real-time cost-first mapping; COIN [70] performs offline planning at the beginning of each billing cycle, which is typically monthly ; CASCARA [49] combines offline planning with reactive updates when the supply is not sufficient. All three operate at the user group level, without accounting for per-stream characteristics. We also include an idealized Oracle with perfect future knowledge as the theoretical lower bound.

As shown in Figure 4, all three solutions incur significantly higher bandwidth cost and latency compared to Oracle. While Entact minimizes cost, it increases latency by 38%; COIN suffers from poor adaptability due to its static nature; and CASCARA, though it reactively updates configurations, still performs suboptimally (1.4× cost and 1.4× latency).

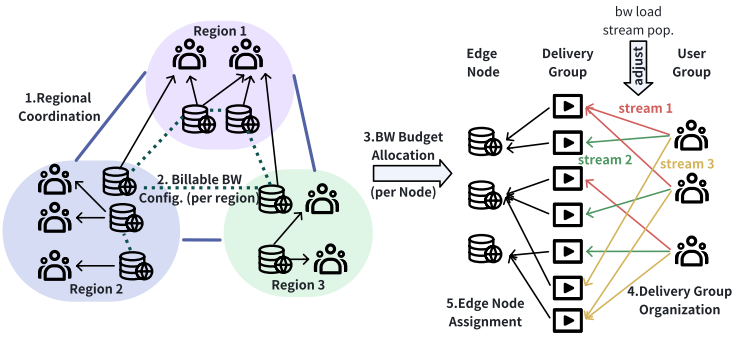


Fig. 5. LiveMap system overview. Left: regional and node-level coordination for bandwidth provisioning. Right: delivery groups, dynamically formed from user groups per stream, are mapped to edge nodes under bandwidth and load constraints.

These performance gaps stem from two fundamental issues: (*i*) These solutions assume stable and sufficient regional capacity and rely on static or region-specific planning. Under supply–demand imbalances, they frequently assign requests to remote regions–up to 69% for CASCARA, up to 78% for Entact, and up to 76% for COIN–leading to inflated latency (detailed in Appendix A). (*ii*) They overlook stream-level geo-distribution heterogeneity. Since mapping is performed at the user-group level, edge nodes may pull a stream from reflectors even if only a single viewer is assigned, leading to wasted reflector bandwidth and unnecessary overhead at edge nodes. Under all three baselines, streams with fewer than 500 viewers used over 35% more edge nodes than Oracle, whereas no significant difference was observed for more popular streams. This result confirms that ignoring per-stream geo-distribution leads to inefficient edge resource utilization, particularly for less popular streams.

Key Insights. The above analysis motivates us to propose two key insights to build an effective request mapping system in live streaming CDNs: (*i*) Cross-regional bandwidth coordination is essential to absorb regional supply-demand imbalances. Beyond reassigning capacity across regions, the system must also reconfigure bandwidth budgets at the node level accordingly. (*ii*) Stream-aware mapping is necessary to handle the geo-temporal diversity in stream popularity while maintaining low latency. This requires dynamically adjusting both the number and assignment of edge nodes for each stream, going beyond static, user-group–level planning.

3 LiveMap Design Overview

Design Challenges. While the high-level idea behind LiveMap is conceptually simple, its realization involves three key design challenges. *First*, how to coordinate bandwidth provisioning across regions while balancing latency and cost? When each region independently borrows bandwidth from neighbors based on local views, it may lead to contention and inefficient cross-region traffic.

Second, how to assign edge nodes to streams while accounting for stream-level heterogeneity? Proximity-first mapping scatters unpopular streams across multiple nodes, inflating reflector load, whereas efficiency-first mapping may route viewers to distant nodes, increasing latency.

Third, how to optimize bandwidth cost per region under heterogeneous billing models? Edge nodes may differ in billing granularity (*e.g.*, monthly vs. sub-monthly) and pricing models (*e.g.*, 95th-percentile vs. fixed-rate) [70]. Moreover, current-cycle usage history affects the remaining billable budget, making forward-looking cost planning per region more complex.

Main Components. The above challenges lead to the design of LiveMap in Figure 5 that consists of two main components: *Bandwidth Provision* and *Stream Mapping*. The Bandwidth Provision component performs *dual-level coordination* to configure the bandwidth budget. Specially, it ensures

Table 2. Symbol Table for Bandwidth Provision

Symbol	Description
r_i	i-th network region
$S(r_j)$	Bandwidth supply of region r_j
$D(r_j)$	Traffic demand of region r_j
$d(r_i, r_j)$	Average latency from r_i to r_j
θ	Threshold for CRS latency tolerance
$CRS(r_i)$	Candidate region set for r_i
$f(i, j)$	Bandwidth borrowed from r_i to r_j
N_p	Set of percentile-billing edge nodes
N_f	Set of fixed-billing edge nodes
z_k^t	Billable bandwidth of node k on day t
u_k^t	95th percentile usage of node k till day t
α_k^t	Remaining free time ratio of node k
d^t	Peak demand of the region at day t
δ_k^t	Bandwidth change for node k at day t
UN, DN	Sets of nodes needing bandwidth up/down
b_{kj}	Bandwidth budget from node k to UG j

Table 3. Symbol Table for Stream Mapping

Symbol	Description
e_{ijk}	equals 1 if requests of user group i for stream j are assigned to edge node k, and 0 otherwise.
p_i	Popularity of stream i (viewer count)
ϕ_j	unit distribution efficiency for stream j, *i.e.*, $p_j / \sum_{i,k} e_{i,j,k}$
u	Reflector (L2) bandwidth utilization
$[\gamma_l, \gamma_u]$	Threshold range of utilization
m	Number of streams adjusted (scale-in/out)
$loss_j(k_1, k_2)$	Latency loss for merging two delivery groups of stream j served by edge nodes k_1 and k_2
$gain_j(i)$	Latency gain for reassigning UG i of stream j
$G_{j,k}$	Set of UGs for stream j served by node k
n_i^{init}	Initial number of nodes assigned to stream i
$p_{\max}$	Maximum normalized popularity
n	Total number of user groups

cost-efficiency and smooth update by node-level coordination (§ 4.2) while maintaining proximity for most user requests by regional coordination (§ 4.1). Then it configures the bandwidth provision plan to each user group at a smaller timescale (5 minutes to align with the bandwidth usage gauging interval) to accommodate the real-time dynamics (§4.3). The Stream Mapping component organizes user groups into per-stream delivery groups based on stream popularity distribution and reflector load through latency-aware scale-in/scale-out operations (§5.2), and then selects appropriate edge nodes for each group according to bandwidth budget per node(§5.3).

4 Bandwidth Provision

This section details the Bandwidth Provision component, starting with the global supply-demand matching by regional coordination in §4.1, then going down into regions to configure billable bandwidth for edge nodes online in §4.2. It finally dynamically allocates edge nodes' bandwidth to user groups every 5 minutes in §4.3.

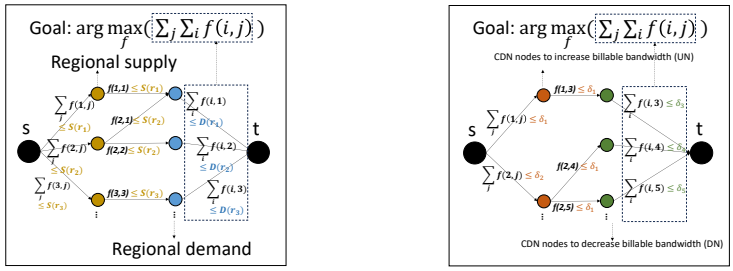
4.1 Regional Coordination: Supply-Demand Matching

The global supply-demand matching module addresses the regional imbalance between supply and demand by determining, for each region with insufficient capacity, which other regions should contribute excess bandwidth to help meet its demand. To keep latency low, the ideal candidate regions for a shortage region r_i are those having excess bandwidth and low latency to r_i. Formally, the candidate region set (CRS) for a region r_i is:

$$CRS(r_i) = \{r_j | S(r_j) > D(r_j) \& d(r_i, r_j) < (1 + \theta) d(r_i, r_i)\},$$

where $S(r_j)$ and $D(r_j)$ are the bandwidth supply and estimated traffic demand of the region r_j in the following day, and $d(r_i, r_j)$, is the average latency from user groups in r_i to the edge nodes in r_j. The parameter θ controls the trade-off between the number of candidate regions and the increased latency from cross-regional borrowing.[4] Each user group measures the latency to every edge node periodically and reports the results to the control plane. Beyond this latency-based filtering, our production system also incorporates policy restrictions, such as ISP-level constraints,

[4]In our implementation, we set $\theta = 0.1$, based on the latency distribution of cross-regional scheduling. Specifically, we experimented with several values of $\theta \in \{0, 0.05, 0.10, 0.15, 0.20\}$ and found that when θ is 0 or 0.05, the solution space is often too constrained to provide sufficient supply coverage across regions. Setting $\theta = 0.1$ strikes a good balance between latency tolerance and the feasibility of cross-regional coordination.



(a) regional S-D match. (b) Billable bw updates

Fig. 6. Maximum flow problems for global supply and demand matching (a) and billable bandwidth updates.

when determining the CRS. Importantly, the CRS is not statically assigned or hardcoded. Instead, it is dynamically maintained, allowing the system to adapt to changing network conditions and policy requirements over time.

As shown in our evaluation (§2), baseline approaches that greedily borrow bandwidth from low-latency regions without global coordination can result in inflated latency. This happens because excessive bandwidth of a candidate region for r_i has been lent to other regions, forcing r_i to borrow bandwidth from those further from it (*i.e.* not in its CRS). To address this issue, we propose a global matching approach that aims to maximize the demand served by CRS for *all regions*, as opposed to individual regions.

Problem Formulation. Let $f(i,j)$ represent the bandwidth borrowed from region r_i to region r_j. The demand of r_j that can be served by $CRS(r_j)$ is thus $\sum_{r_i \in CRS(r_j)} f(i,j)$. Consequently, the total demand that can be served by CRS of all regions is $\sum_{r_j} \sum_{r_i \in CRS(r_j)} f(i,j)$. Our objective is:

$$\arg\max_f \sum_{r_j} \sum_{r_i \in CRS(r_j)} f(i,j). \tag{2}$$

This is illustrated in Figure 6(a), where yellow nodes represent the supply of each region and blue nodes represent the demand. There is an edge from a yellow node i to a blue node j if region r_i is a candidate region for r_j; the edge weight is $f(i,j) < S(r_i)$. To solve this optimization problem, we introduce a source node s connecting to every yellow node, where $f(s,i) = \sum_j f(i,j)$ represents the total demand served by the edge nodes in region i. We enforce the constraint $f(s,i) \le S(r_i)$ to ensure that the total serving demand does not exceed the available supply. Similarly, a sink node t is connected to every blue node, with $f(j,t) = \sum_i f(i,j)$ representing the total demand in region j served by edge nodes in $CRS(r_j)$, constrained by $f(j,t) \le D(r_j)$, ensuring that the total served demand does not exceed the actual demand. Solving Eq. 2 is then equivalent to solving:

$$\arg\max_f \sum_{(j,t)} f(j,t). \tag{3}$$

This objective aligns naturally with the goal of finding the maximum flow in the graph. Appendix B.1 and B.2 detail the conditions and the problem.

Solution. We use the Ford-Fulkerson algorithm [30] to solve the above maximum-flow problem [30]. The solution determines how much demand each region serves locally and, in the event of regional bandwidth shortages, how much is served to other regions. In practice, we perform this regional coordination once per day, aligned with the minimum billing granularity of edge nodes.

4.2 Node-level Coordination: Billable Bandwidth Configuration

Given that regional coordination may reassign the regions served by each node, and that actual usage may deviate from expectation due to demand dynamics, we next update the billable bandwidth of edge nodes within each region. In this way, the regional-level coordination determines how

much capacity should be shifted across regions, while the node-level formulation refines this plan into concrete per-node bandwidth updates that are consistent with both the global reallocation decisions and the real-time bandwidth usage within its region. For clarity, Table 2 summarizes the symbols used in this section.

Problem Formulation. We first model the bandwidth cost optimization per region as a Linear Programming (LP) problem, while taking the historical usage into account:

$$\min \sum_{k \in N_p} c_k \cdot z_k^t + \sum_{k \in N_f} c_k \cdot C_k, \tag{4}$$

$$\text{s.t.} \sum_{k \in N_p} \alpha_k^t \cdot C_k + \sum_{k \in N_p} (1 - \alpha_k^t) \cdot z_k^t + \sum_{k \in N_f} C_k \geq \bar{d}_t, \quad z_k^t \geq u_k^t, \forall k. \tag{5}$$

Here, N_p and N_f represent the edge node sets using the percentile usage-based billing method and the fixed billing method, respectively. z_k^t is node k's billable bandwidth on day t, u_k^t is its actual 95th percentile usage till t, and α_k^t estimates the remaining free time (τ_k^t/γ^t, where τ_k^t represents the remaining free time of k estimated on day t, and γ^t is the aggregate remaining peak time periods in the billing cycle). This ensures that provision suffices for peak demand ($\bar{d}_t$) and that the billable bandwidth is not underprovisioned based on historical usage (u_k^t). Specifically, we estimate τ_k^t as the time intervals in which the bandwidth usage exceeds u_k^t. This indeed is an underestimation of free time as $z_k^t \geq u_k^t$, but it ensures sufficient provision.

This formulation is different from the offline plannings in existing studies [13, 49, 70] for two reasons: (*i*) We relax the original mixed integer linear programming (MILP) formulation to a linear program (LP), reducing the computational complexity from *exponential* (in the number of integer variables) to *polynomial time*. (*ii*) The offline plannings ignore the historical usage of edge nodes as they are performed at the beginning, while we focus on online cost optimization in the middle of a billing cycle and manage to update billable bandwidth configuration, though deviations may exist between the historical configurations and actual bandwidth usage. In practice, we solve the LP problem using the SCIP solver [9] for each region in parallel, and the output is the configured billable bandwidth z_k^t for each edge node k on day t. However, directly applying the output z_k^t can cause large day-to-day fluctuations, triggering fluctuations in bandwidth usage and leading to bandwidth wastage.

Coordinated Updates. To smooth such fluctuations, we introduce a node-level coordinated update mechanism that redistributes changes across nodes. This design aims to smooth out abrupt per-node changes on billable bandwidth due to the non-continuity of ILP solutions. For example, suppose the optimizer suggests increasing the billable bandwidth of Node A by 100 Gbps and decreasing Node B by 80 Gbps. Naively applying both changes would incur a total reconfiguration magnitude of 180 Gbps. However, we can instead leave Node B unchanged and only apply a 20 Gbps increase to Node A. This satisfies the system-wide provisioning requirement while reducing the total update cost from 180 Gbps to 20 Gbps.

Formally, our goal is to minimize day-to-day billable bandwidth configuration changes across edge nodes, that is to minimize $\sum_{k \in N_p} \delta_k$, where $\delta_k^t = |z_k^t - z_k^{t-1}|$. Let $UN = \{k | z_k^t - z_k^{t-1} > 0\}$ and $DN = \{k | z_k^t - z_k^{t-1} < 0\}$ represent node sets where bandwidth should be raised or lowered, respectively. The correction factor $f(i,j) \geq 0$ between node $i \in UN$ and node $j \in DN$ adjusts their final billable bandwidths to $z_i^{t-1} + \delta_i^t - f(i,j)$ for i and $z_j^{t-1} - \delta_j^t + f(i,j)$ for j. Accordingly, the day-to-day billable bandwidth changes equals to $\sum_{k \in N_p} \delta_k - 2 * \sum_{i \in UN \& j \in DN} f(i,j)$. Minimizing total changes then becomes a matter of finding $f(i,j)$ that maximizes $\sum_{i \in UN, j \in DN} f(i,j)$, leading to the following optimization problem:

$$\arg\max_{f} \sum_{i \in UN, j \in DN} f(i,j). \tag{6}$$

Similar to Section 4.1, this optimization can also be regarded as a Maximum-Flow problem as shown in Figure 6(b), where an edge exists from $i \in UN$ to $j \in DN$ if j has more free time remaining in the billing cycle. Source node s connects to UN nodes, and sink node t connects from DN nodes. We also ensure that no node's bandwidth correction exceeds its initial change (δ_k^t).

Solution. We again use the Ford-Fulkerson algorithm to find the optimal corrections $f(i, j)$ for each edge. Then the configured billable bandwidth of node k is updated on day t to:

$$z_k^{t*} = \begin{cases} z_k^{t-1} + \delta_k^t - f(s,k), & k \in UN, \\ z_k^{t-1} + \delta_k^t - f(s,k), & k \in DN, \\ z_k^{t-1} & otherwise, \end{cases}$$

where s is the source node in Figure 6(b).

Algorithm 1: Online Allocation

Input : The demand of each UG d_i, including pseudo UGs
Output: The bandwidth budget b_{kj} for node k and user group j

1 $d \leftarrow \sum_i d_i$, $T_k \leftarrow 0$
2 `AllocateDemand`(d, N_f) //z_k^{t*} equals to C_k
3 `AllocateDemand`(d, N_p)
4 $N_p \leftarrow$ sorted by remaining free time
5 **foreach** *node* $k \in N_p$ **do**
6 **if** $d \leq 0$ **then**
7 return
8 $T_k \leftarrow C_k$ // add extra $C_k - z_k^{t*}$ to node k
9 $d \leftarrow d - (C_k - z_k^{t*})$ // update unfulfilled demand
10 **foreach** *node* k *and user group* j **do**
11 $b_{kj} \leftarrow T_k * \frac{d_j}{\sum_i d_i}$
12 **Function** `AllocateDemand`(*demand* d, *nodes* N)
13 **foreach** *node* $k \in N$ **do**
14 $T_k \leftarrow T_k + z_k^{t*} \cdot \min(1, \frac{d}{\sum_k z_k^{t*}})$
15 $d \leftarrow d - \sum_k T_k$ // update unfulfilled demand

Algorithm 2: Edge Node Mapping

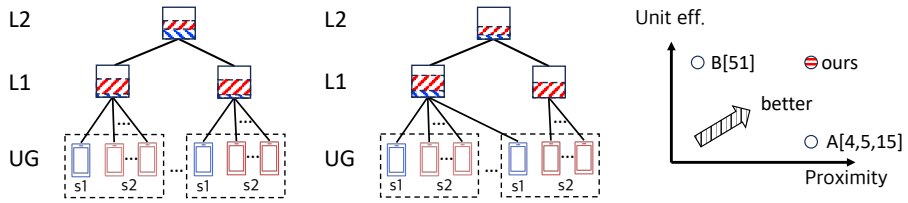
Input : b_{kj}^t: Remaining bandwidth budget for UG j on edge node k in the t-th interval
$UpdS^t(< G, i >)$: Delivery groups requiring new edge node mappings in the t-th interval
Output: Updated stream-edge node mappings

1 $UpdS^t \leftarrow$ sorted by $|G|$ // Larger delivery groups prioritized
2 **foreach** $< G, i > \in UpdS^t$ **do**
3 $Q_j^t \leftarrow \{k \mid b_{kj}^t \geq d_{ij}^t, \forall j \in G\}$ // candidate nodes with enough budget
4 Select K nodes from Q_j^t with probability $p_{jk}^t = \frac{b_{kj}^t}{\sum_{r \in Q_j^t} b_{rj}^t}$
5 **foreach** $j \in G$ **do**
6
7 set $e_{ijk} = 1$, // update node assignment for stream i in UG j
8 $b_{kj}^t \leftarrow b_{kj}^t - d_{ij}^t$ // update remaining budget

4.3 Online Budget Allocation

This module matches the demand of each user group to the bandwidth supply of edge nodes, constrained by the billable bandwidth configured by the module in the previous section. Given the extreme demand dynamics even at the timescale of minutes (§ 2, this module runs every 5 minutes to accommodate the dynamics. The basic idea is to follow the billable bandwidth configuration of each edge node as much as possible, but during demand surges, we exploit the free intervals of 95-percentile billing nodes to handle the extra demand. The output is a bandwidth budget matrix where each element b_{kj} represents the bandwidth allocated from node k to user group j in the following 5 minutes, which also aligns with the billing interval.

Algorithm 1 outlines this process for a region, where z_k^{t*} is the configured billable bandwidth of node k on day t obtained from § 4.2. To manage bandwidth borrowing between regions, we introduce pseudo user groups representing the demand from regions that need to borrow bandwidth. We sum the total (estimated) demand for all user groups in the following time interval, including pseudo ones, and allocate it to fixed billing nodes first, then percentile billing nodes, where the demand is allocated proportionally to the nodes' billable bandwidth (lines 1-3). The remaining demand is distributed to 95-percentile billing nodes based on available free time (lines 4-9). Finally, bandwidth b_{kj} is assigned proportionally to each user group's demand share (lines 10-11).



(a) Proximity first (b) Unit eff. first (c) LiveMap

Fig. 7. Design space analysis for stream mapping.

5 Adaptive Stream Mapping

We next describe the Stream Mapping component, which is responsible for assigning stream requests to edge nodes under the constraints of the bandwidth budget allocated from edge nodes to individual user groups. This component runs every 5 minutes to align with the online budget allocation. We begin with the design space analysis and then proceed to the detailed design. For clarity, Table 3 summarizes the symbols used in this section.

5.1 Design Space Analysis

The stream mapping determines $e_{ijk} = \{0, 1\}$, with $e_{i,j,k} = 1$ signifies that the request for stream j from user group i is mapped to edge node k in the following 5-minute time interval; $e_{ijk} = 0$ otherwise. The mapping is constrained by b_{kj} (the bandwidth allocated from k to j in the interval).

A strawman solution to decide e_{ijk} is to select the nearest edge nodes for requests (called proximity first), provided the node has sufficient available capacity. While this prioritizes the latency, it inadvertently results in unpopular streams being distributed across multiple edge nodes. Given the huge number of unpopular streams, the load on reflectors would be greatly inflated. Figure 7(a) illustrates an example where stream $s1$ has much fewer viewers than stream $s2$ but imposes the same load on the reflectors. The issue behind is that unpopular streams would have low *unit distribution efficiency*, which is computed by $\phi_j = \frac{p_i}{\sum_{i,k} e_{i,j,k}}$ for the stream i with popularity of p_i (measured by the number of views).

An alternative mapping approach is thus unit distribution efficiency first, which aims to optimize the following objective function:

$$\max \min_{stream\ j} \frac{p_j}{\sum_{i,k} e_{i,j,k}}. \tag{7}$$

Nevertheless, this approach potentially leads to latency inflation for unpopular streams as illustrated in Figure 7(b), where some requests of unpopular stream $s1$ are served by remote edge nodes. This is undesired during periods of low reflector load (*e.g.*, off-peak hours).

In response, we design an *adaptive* mapping approach that dynamically scales in or scales out edge nodes for individual streams according to the stream popularity and L2 nodes' load. This adaptive mapping provides a good balance between the unit distribution efficiency and the viewing latency as shown in Figure 7(c). At its core, the stream mapping organizes user groups into **delivery groups**, each of which consists of one or more user groups watching the same stream. Delivery groups are dynamically formed and adjusted to reflect popularity skew and bandwidth constraints (§5.2), and then assigned to edge nodes according to real-time provisioning and load (§5.3).

5.2 Delivery Group Organization

Stream Selection. We use the bandwidth utilization of reflectors as a load indicator, aiming to keep the utilization u within a predefined range $[\gamma_l, \gamma_u]$. When $u > \gamma_u$, we reduce reflector load by consolidating delivery for low-efficiency streams. When $u < \gamma_l$, we enhance proximity by allowing

more localized delivery for high-efficiency streams. Otherwise, no changes are made. The number of streams m to adjust is proportional to the deviation of u from the target range:

$$m = \begin{cases} N_s \cdot \left(1 - \frac{\gamma_u}{u}\right), & u > \gamma_u \quad \texttt{(scale-in)}, \\ N_s \cdot \left(\frac{\gamma_l}{u} - 1\right), & u < \gamma_l \quad \texttt{(scale-out)}, \\ 0, & \text{otherwise}, \end{cases}$$

where N_s is the number of ongoing streams.

Depending on the direction of adjustment, we select m streams with the lowest (for scale-in) or highest (for scale-out) unit efficiency. For each selected stream j, we identify user groups to be reassigned, which collectively form new or modified delivery groups.

Proximity-aware Scale-In/Out. For stream j, to reduce the number of edge nodes (scale-in), we merge two delivery groups served by edge nodes k_1 and k_2, where the remapping from k_1 to k_2 incurs minimal quality loss. Note that new requests are directed to the newly assigned edge node while existing connections remain on their original nodes. The quality loss (*i.e.* latency inflation) caused by this re-mapping is estimated as:

$$\text{loss}_j(k_1, k_2) = \sum_{ug \in G_{j,k_1}} \omega_{ug} \cdot \max\left(0, \frac{l(ug, k_2) - l(ug, k_1)}{l(ug, k_1)}\right),$$

where $l(ug, k)$ is the average latency between user group ug and edge node k, and ω_{ug} is the request volume from ug for stream j. To minimize latency loss, we select the user groups G_{j,k_m} and edge node k_n satisfying:

$$m, n = \arg \min_{m,n \in E_j, m \neq n} \text{loss}_j(k_m, k_n),$$

where E_j represents the edge node set currently serving stream j. Accordingly, the merged delivery group is $Gj, k_m \cup G_{j,k_n}$. As this calculation is done per stream and in parallel, the operation completes within 1 second.

To scale out, we identify user group i for stream j where quality gain is maximized by remapping to a new edge node k. The quality gain is calculated as:

$$\text{gain}_j(i) = \max_{k \in \{k | b_{ki} > 0\}} \frac{l(ug, k_1) - l(ug, k)}{l(ug, k_1)}.$$

The user group i with the highest gain is moved to form a new delivery group to enhance proximity:

$$i = \arg \max_{i \in UG} \text{gain}_j(i).$$

Handling Cold Start. For a new stream i, we determine the number of edge nodes n_i^{init} that serve it based on its estimated popularity p_i (normalized to [0, 100] here) and the real-time reflectors' bandwidth utilization u as:

$$n_i^{\text{init}} = \min\left\{1, \frac{n}{1 + e^{p_{\max} \cdot u - p_i}}\right\}, \tag{8}$$

where n is the total number of user groups, and $p_{\max} = 100$. This calculation follows an S-shaped curve controlled by u, allowing fewer edge nodes for less popular streams under high utilization. In case $n_i^{\text{init}} < n$, we simulate $(n - n_i^{\text{init}})$ scale-in operations to identify n_i^{init} delivery groups.

5.3 Edge Node Assignment

We finally assign edge nodes to delivery groups every 5 minutes based on: (*i*) updated delivery groups generated by the adaptation module; (*ii*) available bandwidth budget from each edge node. If a node's provisioned bandwidth for a delivery group is exceeded, we randomly select streams and remap them to other edge nodes that have available bandwidth to this delivery group.

Algorithm 2 describes the assignment logic. Each element in $UpdS^t$ represents a delivery group for stream i needing reassignment. We first sort the elements in $UpdS^t$ in descending order of the size of G (line 1) in order to prioritize larger ones. For each stream i and unmapped delivery group

Table 4. Performance (relative to Oracle) comparison.

	Entact	COIN	CASCARA	LiveMap
BW Cost(%)(↓)	31.5 (20.1%↑)	42.1 (30.7%↑)	40.5 (29.1%↑)	**11.4**
Latency(%)(↓)	46.7 (29.9%↑)	46.3 (29.5%↑)	37.5 (20.7%↑)	**16.8**

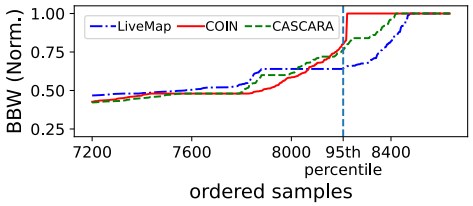


Fig. 8. Comparison of billable bandwidth.

Table 5. Live streaming service overview.

	#streams	#viewers	Traffic Volumn
12 p.m.	17.85k	258.99k	0.78 Tbps
8 p.m.	41.78k	468.45k	1.90 Tbps

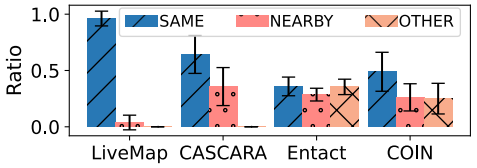


Fig. 9. Ratio of requests served by same/near/other regions.

G, we choose edge nodes from the node-set Q_j^t, *i.e.*, edge nodes that have abundant bandwidth budget for user groups in G (line 3). Candidate nodes are sampled proportionally to their remaining bandwidth budget, ensuring edge nodes with more available bandwidth are more likely to be selected (line 4). Finally, mappings are updated for each user group in the delivery group (lines 5-7).

6 Performance Evaluation

We implement LiveMap and evaluate it via trace-driven simulation (§ 6.1) to contrast it with state-of-the-art solutions and large scale deployment (§ 6.2) to show its performance in the wild.

6.1 Data-driven Simulation

We first conduct data-driven simulations to compare LiveMap with the state-of-the-art solutions mentioned in § 2, including: Entact [69], COIN [70], and CASCARA [49]. We also include an Oracle baseline with perfect future knowledge, as described in §2. The dataset was collected from our private CDN and includes detailed information on each CDN node (capacity, unit price, and billing method), each user request (start time, duration, requested stream, and origin), and network measurements (round-trip time (RTT) between each user group and edge node). We collect and replay *4.86 billion* requests based on the trace logs for simulation[5].

For evaluation, we focus on two primary metrics: (*i*) *Bandwidth cost*, calculated as the total cost across all CDN nodes under their respective billing models; and (*ii*) *Access latency*, estimated as the RTT between the user origin and the assigned edge node, based on measured network statistics.

Overall Performance. Table 4 compares LiveMap with the baselines, reporting the cost and latency increases relative to the Oracle. LiveMap achieves significant cost savings, ranging from 20% to 31% compared with the baselines. The relatively high cost in CASCARA and COIN stems from their heavy reliance on offline planning and reactive local-greedy updates, leading to huge gaps between the actual billable bandwidth and planned billable bandwidth. Notably, although Entact assigns requests greedily to the cheapest edge nodes, it incurs high cost due to under-utilization of nodes during their free periods (*i.e.* the 5% of the time).

Unlike above, LiveMap not only excels in cost efficiency by dual-level coordination, but also achieves the lowest access latency due to proximity awareness throughout bandwidth optimization.

Bandwidth Usage. LiveMap's reduced bandwidth cost can be explained by the 95th percentile of bandwidth usage (*i.e.* billable bandwidth) on a specific node in Figure 8. Here, the bandwidth usage is normalized by the node's capacity. We sorted the node bandwidth usage sampled every 5 minutes (8,640 data points in a month billing cycle). The 8,208th value is thus the billable bandwidth. The results show a much lower billable bandwidth of LiveMap in comparison with the two baselines. It

[5]To reduce simulation overhead, we exclude requests for extremely unpopular streams with fewer than 10 viewers.

Table 6. Bandwidth cost saving by LiveMap

	BwSaving (edges)	BwSaving (reflectors)	CostSaving (edges)	CostSaving (reflectors)	CostSaving (all)
Only BP	15.3%	5.2%	38.5%	7.8%	30.6%
BP+SM	11.7%	32.8%	33.7%	18.9%	41.2%

Table 7. Reduction of latency and buffering rate

	BufRate (P90)	BufRate (P95)	BufRate (avg)	AccLat (P90)	AccLat (P95)	AccLat (avg)
Only BP	10.3%	8.3%	13.8%	19.8%	16.7%	17.6%
BP+SM	9.2%	7.5%	12.7%	18.8%	15.2%	20.3%

is also interesting to see that there are more intervals (*i.e.*, 327 intervals) in LiveMap during which the usage matches the billable bandwidth. The results demonstrate LiveMap's better bandwidth utilization planning.

Supply Provision Regions. We calculate the distribution of traffic fulfillment based on proximity. Specifically, requests served by edge nodes within the same region are categorized as `same`, those served by nearby regions in the CRS (see § 4.1) as `nearby`, and others as `other`. Figure 9 compares the distribution. LiveMap generates the least `other` traffic, avoiding cross-region delivering. CASCARA, while also involving less `other` traffic, its `nearby` traffic is much higher than LiveMap because of its local-greedy matching. COIN suffers from both high `nearby` and `other` traffic, because of proximity ignorance. Entact prioritizes cheaper nodes, resulting in higher cross-region transferring. The results explain the reduced access latency by LiveMap.

6.2 Large Scale Deployment

We further deploy LiveMap in our private CDN, with the detailed stream delivery workflow provided in Appendix C. We report deployment statistics from a typical day (November 3rd, 2023) in Table 5, including the number of concurrent streams, viewers, and total traffic volume observed at two peak hours (12 p.m. and 8 p.m.). On that day, the system operated with 44 dedicated CDN nodes and 79 heterogeneous edge nodes. Both the viewership scale and node scale are consistent with our daily operating conditions. The large-scale evaluation consists of two stages: first deploying only the Bandwidth Provision (BP) component, followed by the full deployment of LiveMap with both Bandwidth Provision and Stream Mapping (BP+SM). This enables us to examine the performance of each component in isolation.

For comparison, we also report the results from OffPlan, which is our request mapping system before the deployment of LiveMap. OffPlan employs an offline bandwidth provisioning algorithm similar to COIN [70] and a popularity-agnostic stream mapping strategy, representative of common practice in live streaming CDNs.

We collect real-time bandwidth usage recorded every 5 minutes and QoE metrics (such as RTT and buffering times) per request over a month for each of the three deploying stages: OffPlan, Only BP, and LiveMap (BP+SM). Notably, the supply and demand of the three months are similar (with a difference of less than 10%), ensuring a fair comparison of bandwidth costs across deployment stages under comparable operating conditions.

Bandwidth Cost. We focus on two cost metrics (compared with OffPlan): (*i*) Bandwidth saving (BwSaving), which measures how much the total billable bandwidth is reduced; (*ii*) Cost saving (CostSaving), which measures how much the final cost is saved. The cost is calculated based on the money paid to the network operators that provide the links. Table 6 reports the results. Overall, LiveMap saves cost by as much as 41.2%. Only BP saves 30.6% bandwidth cost, primarily due to the significant reduction (15.3%) in billable bandwidth for edge nodes. Additionally, the SM component leads to 10.6% more cost savings, mostly because of the reduced bandwidth usage (32.8%) of reflectors. This is attributed to the dynamic organization of delivery groups to optimize reflectors' bandwidth usage. While BP alone saves bandwidth cost, the combination of BP and SM incurs greater savings by jointly optimizing edge- and reflector-level bandwidth usage.

Access Latency and QoE. We next evaluate the performance in terms of user experience, including two metrics: (*i*) Buffering rate (BufRate), which is a critical QoE metric for video streaming [16],

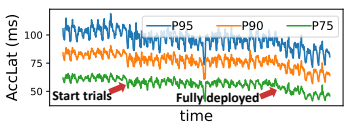


Fig. 10. The access delay before and after LiveMap.

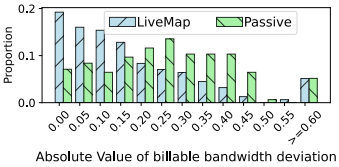


Fig. 11. Billable bandwidth deviation from optimum.

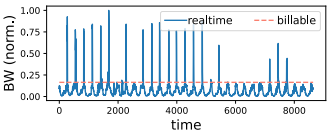


Fig. 12. Example for real-time bandwidth usage of a node.

computed by dividing the number of buffering events by the viewing duration across all views; (*ii*) Access latency (AccLat), which is the RTT between the viewing origin and the edge node.

We present the tail distributions (90th and 95th percentiles) and mean values for these two metrics in Table 7. Deploying only the BP component results in a 13.8% reduction in average buffering rate and a 17.6% reduction in average access delay compared with OffPlan. We also present a detailed report for performance before and after LiveMap's deployment in Figure 10, including an intermediate phase during which the deployment scope was gradually expanded. Notably, as LiveMap was rolled out to more regions and nodes, access latency consistently decreased, demonstrating its effectiveness at scale. These improvements are attributed to several factors, including proximity-aware supply-demand matching, coordinated billable bandwidth updates allowing more demand to be met locally, and the better utilization of free time.

As expected, after combining the SM component with the BP component, the tail metrics slightly increase (within 2%), because of the dynamic scale-in operations to control the reflectors' bandwidth cost. Nevertheless, in comparison with OffPlan, LiveMap still manages to reduce the 95th percentile of these two metrics by 7.5% and 15.2%, respectively.

we next dissect the effectiveness of major subcomponents in LiveMap, with a focus on their performance relative to the theoretical optimum.

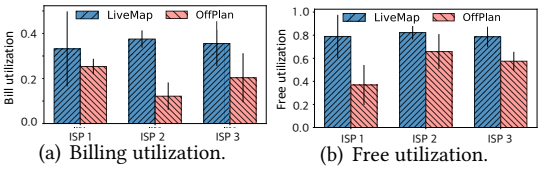
Billable Bandwidth Configuration. One of the major cost-saving contributors is the billable bandwidth configuration in § 4.2. We compare LiveMap's configuration for each node with the Oracle that generates theoretical optimum as it has perfect knowledge of future demand and supply. We also include the approach that only updates the bandwidth configuration with *Passively*, *i.e.* only when the current configuration cannot meet demand requirements, similar to Cascara [49].

Figure 11 reports the distribution of deviation from the Oracle for LiveMap and Passive. The smaller the deviation is, the closer the actual bandwidth cost is to the optimum. Notably, only 21.3% of edge nodes in LiveMap exhibit a difference greater than 0.25. In contrast, this percentage is as high as 51.8% for *Passive*, highlighting the benefits of dynamic billable bandwidth configuration.

Online Bandwidth Allocation. Next, we examine the bandwidth usage patterns of edge nodes over time and explain how online bandwidth allocation (§4.3) reduces costs. Figure 12 illustrates a node's bandwidth usage over a month after deploying LiveMap, showing a daily pattern with peak usage in the evenings. Usage surges close to capacity during free intervals (scheduled to serve peak-hour traffic) and stays near the billable bandwidth otherwise, resulting in the node's billable bandwidth being only 20% of its capacity.

Formally, we define the *billing utilization* of a node in a billing cycle as the average ratio of its bandwidth usage to its billable bandwidth during non-free time periods. Similarly, *free utilization* is defined as the average ratio of bandwidth usage to capacity during scheduled free time intervals. Figure 13 compares these two metrics between LiveMap and OffPlan. We see that billing utilization increases by 8%, 25%, and 15% for the three major ISPs, respectively, while free utilization increases by 42%, 17%, and 21%, respectively. This increased utilization directly contributes to significant cost reductions. It is important to note that the free time intervals are strategically scheduled to

coincide with peak hours, resulting in much higher utilization during these intervals compared with non-free time intervals (80% vs. 35% on average) when traffic demand is lower.



(a) Billing utilization. (b) Free utilization.

Fig. 13. Bandwidth utilization comparison.

Computational Overhead. The overhead of LiveMap is evaluated by the time spent on solving the optimizations. LiveMap is run on a moderate server with 16-core processors and 32GB of RAM. Specifically, the time spent in solving the maximum-flow problem in § 4 is less than 1 second using the Ford-Fulkerson algorithm [30]. The LP problem for optimizing bandwidth cost is solved for each network region in parallel, which takes a few seconds with a median of around 8 seconds. The streaming mapping component takes only around 300ms credited to the parallel execution of scale-in and scale-out operations for each stream. In summary, these execution times are at the second level and can be considered negligible relative to their execution cycles (5 minutes).

Prediction and Robustness. LiveMap relies on two prediction tasks—traffic demand and stream popularity for guiding bandwidth provisioning and stream mapping. While traffic patterns in live streaming are highly dynamic and often unpredictable, our system combines lightweight prediction with adaptive reconfiguration to maintain robust performance. We detail our prediction models and evaluate LiveMap's sensitivity to prediction inaccuracies in Appendix D, showing that its cost efficiency remains stable even under vaiours deviations in bandwidth forecasts.

7 Lesson Learned

Reflector Bandwidth Cost. We found that optimizing bandwidth cost for reflectors is complex but not profitable. The reason is that the traffic demand of reflectors, which is the key input for optimization, is influenced by both the traffic volume at the edge nodes and the unit bandwidth efficiency. Both factors constantly vary as the demand dynamics as well as the dynamic scale-in and scale-out operations every 5 minutes (§ 5). Consequently, we adopt a simple yet effective approach: we regulate the bandwidth of reflectors to remain within an acceptable range, thereby controlling their bandwidth costs.

Surges Beyond Mapping Frequency. We observe that platform promotions can add hundreds of thousands of viewers to a stream in under five minutes, faster than our mapping update cycle. To absorb these bursts, we maintain a rotating pool of L1 nodes with remaining free time. When a delivery group's actual viewership significantly exceeds its expected popularity, incoming requests are redirected to this pool. The burst rides on the free intervals, preserving QoS and containing cost without complex prediction.

Vendor-Driven Constraints. We have observed that vendors (*e.g.*, ISPs) may impose countermeasures to limit overuse of inexpensive bandwidth. For example, some enforce minimum utilization thresholds on ultra-cheap nodes, requiring baseline usage to retain favorable billing terms. Yet our measurements show that high utilization degrades QoE (*e.g.*, increased rebufferings). We therefore cap their usage and treat them as fixed-rate nodes, ignoring percentile billing headroom since they are already cost-effective. Moreover, some vendors restrict the usage of low-cost nodes to serve only local traffic, discouraging overuse of inexpensive, remote bandwidth resources. Thus, our system prioritizes in-province edge nodes for request mapping whenever possible. At the same time, our dual-level coordination decouples regional bandwidth provisioning and node-level planning, allowing LiveMap to quickly adapt to ISP-imposed locality policies.

Hybrid Live-VoD Trials. At the time of writing this paper, LiveMap has been tested with several trials to support hybrid live-VoD CDNs. We piloted LiveMap on a mixed live-and-VoD CDN. Billable bandwidth is first optimized globally across all traffic, then split proportionally between services (as in § 4). The stream mapping (§ 5) runs independently inside each service as they exhibit different dynamic patterns. Early trials show the same cost and latency gains, suggesting LiveMap 's techniques extend beyond pure live streaming.

8 Related Work

CDN Request Mapping. CDN request mapping can be broadly categorized into inter-CDN selection and intra-CDN mapping. Existing studies on inter-CDN selection [19, 27, 38] focus on choosing which external CDN a client should use at the chunk level or stream level. In contrast, intra-CDN mapping has been widely explored to improve user service quality within a single CDN, offering finer-grained control over request distribution and performance optimization. DNS-based user mapping and anycast-based mapping are prevalent for providing proximity-aware routing [2, 3, 18]; some studies also assess anycast performance [31, 45, 71, 72]. In addition, numerous studies present solutions to monitor and examine the network states for improved user experience [12, 17, 26, 47, 58, 59]. LiveMap can be integrated with these systems as they are complementary.

Bandwidth Cost Optimization. A large body of work models traffic scheduling and resource allocation problems as LP or MILP with different optimization objectives, such as cost, throughput, delay, or fairness [4–6, 8, 14, 15, 21, 23, 25, 39–42, 46, 51, 55, 61, 63, 64, 67]. In the context of cost optimization, linear billing methods have been examined [1, 29, 36, 43, 62, 69], while non-linear billing schemes have been proven to be NP-Hard [1]. Recent efforts have explored the opportunities of cost-saving under the 95th-percentile billing method in WANs [24, 50, 57] and cloud networks [49, 70]. In contrast, we focus on live CDNs that exhibit higher dynamics and extremely skewed popularity distribution across streams. While TrafAda [59] focuses on live CDNs, its focus is on bitrate selection according to the billable bandwidth, rather than request mapping.

9 Conclusion

This paper has introduced LiveMap, a cost-efficient request mapping system for large-scale live streaming CDNs. LiveMap addresses two emerging challenges in modern live delivery: dynamic regional supply-demand imbalance and per-stream heterogeneity in popularity and geography. It achieves cost-effective delivery through dual-level bandwidth provisioning and adapts to stream diversity via dynamic, popularity-driven stream mapping. By incorporating proximity awareness throughout its design, LiveMap strikes a balance between minimizing bandwidth cost and maintaining low access latency. Extensive evaluations through trace-driven simulations and large-scale deployment have demonstrated the superior performance of LiveMap.

10 Acknowledgement

We sincerely thank our shepherd and the anonymous reviewers for their valuable comments. We also thank Chao Geng from Bilibili for his valuable contributions during the deployment of LiveMap in production. This work was supported in part by the National Key R&D Program of China (2022YFB2901800). Corresponding author: Zhenyu Li.

References

[1] Micah Adler, Ramesh K Sitaraman, and Harish Venkataramani. 2011. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks* 55, 18 (2011), 4007–4020.

[2] Zakaria Al-Qudah, Seungjoon Lee, Michael Rabinovich, Oliver Spatscheck, and Jacobus Van der Merwe. 2009. Anycast-aware transport for content delivery networks. In *Proceedings of the 18th international conference on World wide web*. 301–310.

[3] Hussein A Alzoubi, Seungjoon Lee, Michael Rabinovich, Oliver Spatscheck, and Jacobus Van Der Merwe. 2011. A practical architecture for an anycast CDN. *ACM Transactions on the Web (TWEB)* 5, 4 (2011), 1–29.

[4] Congkai An, Huanhuan Zhang, Shibo Wang, Jingyang Kang, Anfu Zhou, Liang Liu, Huadong Ma, Zili Meng, Delei Ma, Yusheng Dong, et al. 2025. Tooth: Toward Optimal Balance of Video {QoE} and Redundancy Cost by {Fine-Grained}{FEC} in Cloud Gaming Streaming. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 635–651.

[5] Konstantin Andreev, Bruce M Maggs, Adam Meyerson, Jevan Saks, and Ramesh K Sitaraman. 2011. Algorithms for constructing overlay networks for live streaming. *arXiv preprint arXiv:1109.4114* (2011).

[6] Konstantin Andreev, Bruce M Maggs, Adam Meyerson, and Ramesh K Sitaraman. 2003. Designing overlay multicast networks for streaming. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*. 149–158.

[7] Bilibili. 2025. *Bilibili homepage*. https://www.bilibili.com/

[8] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. 2019. TEAVAR: striking the right utilization-availability balance in WAN traffic engineering. In *Proceedings of the ACM Special Interest Group on Data Communication*. 29–43.

[9] Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. 2024. *The SCIP Optimization Suite 9.0*. Technical Report. Optimization Online. https://optimization-online.org/2024/02/the-scip-optimization-suite-9-0/

[10] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[11] Qingpeng Cai, Zhenghai Xue, Chi Zhang, Wanqi Xue, Shuchang Liu, Ruohan Zhan, Xueliang Wang, Tianyou Zuo, Wentao Xie, Dong Zheng, et al. 2023. Two-stage constrained actor-critic for short video recommendation. In *Proceedings of the ACM Web Conference 2023*. 865–875.

[12] Matt Calder, Ryan Gao, Manuel Schröder, Ryan Stewart, Jitendra Padhye, Ratul Mahajan, Ganesh Ananthanarayanan, and Ethan Katz-Bassett. 2018. Odin:{Microsoft's} scalable {Fault-Tolerant}{CDN} measurement system. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 501–517.

[13] Huan Chen, Huiyou Zhan, Haisheng Tan, Huang Xu, Weihua Shan, Shiteng Chen, and Xiang-Yang Li. 2022. Online Traffic Allocation Based on Percentile Charging for Practical CDNs. In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[14] Shawn Shuoshuo Chen, Keqiang He, Rui Wang, Srinivasan Seshan, and Peter Steenkiste. 2024. Precise Data Center Traffic Engineering with Constrained Hardware Resources. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 669–690.

[15] Tianyu Chen, Yiheng Lin, Nicolas Christianson, Zahaib Akhtar, Sharath Dharmaji, Mohammad Hajiesmaili, Adam Wierman, and Ramesh K Sitaraman. 2024. SODA: An adaptive bitrate controller for consistent high-quality video streaming. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 613–644.

[16] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the impact of video quality on user engagement. *ACM SIGCOMM computer communication review* 41, 4 (2011), 362–373.

[17] Xun Fan, Ethan Katz-Bassett, and John Heidemann. 2015. Assessing affinity between users and CDN sites. In *Traffic Monitoring and Analysis: 7th International Workshop, TMA 2015, Barcelona, Spain, April 21-24, 2015. Proceedings 7*. Springer, 95–110.

[18] 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}. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. 381–394.

[19] Aditya Ganjam, Faisal Siddiqui, Jibin Zhan, Xi Liu, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. 2015. C3:{internet-scale} control plane for video quality optimization. In *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. 131–144.

[20] Peisheng Guo, Jiao Zhang, Yuqing Wang, Haozhe Li, Zhichen Xue, Yajie Peng, Rui Han, Xiaofei Pang, Tao Huang, Ruili Fang, et al. 2025. HELDR: Packet Loss Detection and Retransmission for Live Streaming Hyper-Edge Network. In

2025 IEEE/ACM 33rd International Symposium on Quality of Service (IWQoS). IEEE, 1–10.

[21] Kathrin Hanauer, Monika Henzinger, Lara Ost, and Stefan Schmid. 2023. Dynamic Demand-Aware Link Scheduling for Reconfigurable Datacenters. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*.

[22] iCDN. 2024. *iCDN Prime, Hybrid CDN-node as a service.* https://www.icdntele.com/index.php/icdn-prime

[23] Paras Jain, Sam Kumar, Sarah Wooders, Shishir G Patil, Joseph E Gonzalez, and Ion Stoica. 2023. Skyplane: Optimizing Transfer Cost and Throughput Using {Cloud-Aware} Overlays. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1375–1389.

[24] Virajith Jalaparti, Ivan Bliznets, Srikanth Kandula, Brendan Lucier, and Ishai Menache. 2016. Dynamic pricing and traffic engineering for timely inter-datacenter transfers. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 73–86.

[25] Virajith Jalaparti, Peter Bodik, Ishai Menache, Sriram Rao, Konstantin Makarychev, and Matthew Caesar. 2015. Network-aware scheduling for data-parallel jobs: Plan when you can. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 407–420.

[26] Umar Javed, Italo Cunha, David Choffnes, Ethan Katz-Bassett, Thomas Anderson, and Arvind Krishnamurthy. 2013. PoiRoot: Investigating the root cause of interdomain path changes. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 183–194.

[27] Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. 2013. Shedding light on the structure of internet video quality problems in the wild. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 357–368.

[28] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. 2017. Pytheas: Enabling Data-Driven Quality of Experience Optimization Using Group-Based Exploration-Exploitation. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 393–406. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/jiang

[29] Bhaskar Kataria, Palak LNU, Rahul Bothra, Rohan Gandhi, Debopam Bhattacherjee, Venkata N Padmanabhan, Irena Atov, Sriraam Ramakrishnan, Somesh Chaturmohta, Chakri Kotipalli, et al. 2024. Saving Private WAN: Using Internet Paths to Offload WAN Traffic in Conferencing Services. *Proceedings of the ACM on Networking* 2, CoNEXT4 (2024), 1–22.

[30] Jon Kleinberg and Eva Tardos. 2006. *Algorithm design*. Pearson Education India.

[31] Thomas Koch, Ethan Katz-Bassett, John Heidemann, Matt Calder, Calvin Ardi, and Ke Li. 2021. Anycast in context: A tale of two systems. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 398–417.

[32] Thomas Koch, Shuyue Yu, Sharad Agarwal, Ethan Katz-Bassett, and Ryan Beckett. 2023. PAINTER: Ingress Traffic Engineering and Routing for Enterprise Cloud Networks. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 360–377. doi:10.1145/3603269.3604868

[33] Leonidas Kontothanassis, Ramesh Sitaraman, Joel Wein, Duke Hong, Robert Kleinberg, Brian Mancuso, David Shaw, and Daniel Stodolsky. 2004. A transport layer for live streaming in a content delivery network. *Proc. IEEE* 92, 9 (2004), 1408–1419.

[34] Federico Larumbe and Abhishek Mathur. 2015. Under the hood: Broadcasting live video to millions. https://engineering.fb.com/2015/12/03/ios/under-the-hood-broadcasting-live-video-to-millions/.

[35] Jinyang Li, Zhenyu Li, Ri Lu, Kai Xiao, Songlin Li, Jufeng Chen, Jingyu Yang, Chunli Zong, Aiyun Chen, Qinghua Wu, Chen Sun, Gareth Tyson, and Hongqiang Harry Liu. 2022. LiveNet: a low-latency video transport network for large-scale live streaming. In *Proceedings of the ACM SIGCOMM 2022 Conference* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 812–825. doi:10.1145/3544216.3544236

[36] Wenxin Li, Keqiu Li, Deke Guo, Geyong Min, Heng Qi, and Jianhui Zhang. 2016. Cost-minimizing bandwidth guarantee for inter-datacenter traffic. *IEEE Transactions on Cloud Computing* 7, 2 (2016), 483–494.

[37] Zhenyu Li, Jinyang Li, Qinghua Wu, Gareth Tyson, and Gaogang Xie. 2023. A Large-Scale Measurement and Optimization of Mobile Live Streaming Services. *IEEE Transactions on Mobile Computing* 22, 12 (2023), 7294–7309. doi:10.1109/TMC.2022.3208094

[38] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. 2012. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (Helsinki, Finland) *(SIGCOMM '12)*. Association for Computing Machinery, New York, NY, USA, 359–370. doi:10.1145/2342356.2342431

[39] Ximeng Liu, Shizhen Zhao, Yong Cui, and Xinbing Wang. 2024. FIGRET: Fine-Grained Robustness-Enhanced Traffic Engineering. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 117–135.

[40] Yingling Mao, Xiaojun Shang, and Yuanyuan Yang. 2022. Joint resource management and flow scheduling for sfc deployment in hybrid edge-and-cloud network. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 170–179.

[41] Congcong Miao, Zhizhen Zhong, Yunming Xiao, Feng Yang, Senkuo Zhang, Yinan Jiang, Zizhuo Bai, Chaodong Lu, Jingyi Geng, Zekun He, et al. 2024. MegaTE: Extending WAN Traffic Engineering to Millions of Endpoints in Virtualized Cloud. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 103–116.

[42] Congcong Miao, Zhizhen Zhong, Yiren Zhao, Arpit Gupta, Ying Zhang, Sirui Li, Zekun He, Xianneng Zou, and Jilong Wang. 2025. PreTE: Traffic Engineering with Predictive Failures. In *Proceedings of the ACM SIGCOMM 2025 Conference*. 780–795.

[43] Murtaza Motiwala, Amogh Dhamdhere, Nick Feamster, and Anukool Lakhina. 2012. Towards a cost model for network traffic. *ACM SIGCOMM Computer Communication Review* 42, 1 (2012), 54–60.

[44] Matthew K. Mukerjee, David Naylor, Junchen Jiang, Dongsu Han, Srinivasan Seshan, and Hui Zhang. 2015. Practical, Real-time Centralized Control for CDN-based Live Video Delivery. *SIGCOMM Comput. Commun. Rev.* 45, 4 (aug 2015), 311–324. doi:10.1145/2829988.2787475

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

[46] Hojin Park, Ziyue Qiu, Gregory R Ganger, and George Amvrosiadis. 2024. Reducing Cross-Cloud/Region Costs with the Auto-Configuring MACARON Cache. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*. 347–368.

[47] Ingmar Poese, Benjamin Frank, Bernhard Ager, Georgios Smaragdakis, Steve Uhlig, and Anja Feldmann. 2011. Improving content delivery with PaDIS. *IEEE Internet Computing* 16, 3 (2011), 46–52.

[48] Aravindh Raman, Gareth Tyson, and Nishanth Sastry. 2018. Facebook (A) live? Are live social broadcasts really broad casts?. In *Proceedings of the 2018 world wide web conference*. 1491–1500.

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

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

[51] Yu Sun, Chi Lin, Jiankang Ren, Pengfei Wang, Lei Wang, Guowei Wu, and Qiang Zhang. 2022. Subset selection for hybrid task scheduling with general cost constraints. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 790–799.

[52] Linpeng Tang, Qi Huang, Amit Puntambekar, Ymir Vigfusson, Wyatt Lloyd, and Kai Li. 2017. Popularity prediction of facebook videos for higher quality streaming. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. 111–123.

[53] Shisong Tang, Qing Li, Xiaoteng Ma, Ci Gao, Dingmin Wang, Yong Jiang, Qian Ma, Aoyang Zhang, and Hechang Chen. 2022. Knowledge-based temporal fusion network for interpretable online video popularity prediction. In *Proceedings of the ACM Web Conference 2022*. 2879–2887.

[54] Yu Tian, Zhenyu Li, Matthew Yang Liu, Jian Mao, Gareth Tyson, and Gaogang Xie. 2024. Cost-Saving Streaming: Unlocking the Potential of Alternative Edge Node Resources. In *Proceedings of the 2024 ACM on Internet Measurement Conference*. 580–587.

[55] Midhul Vuppalapati, Giannis Fikioris, Rachit Agarwal, Asaf Cidon, Anurag Khandelwal, and Eva Tardos. 2023. Karma: Resource Allocation for Dynamic Demands. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*.

[56] Haiping Wang, Ruixiao Zhang, Chaojun Li, Zhichen Xue, Yajie Peng, Xiaofei Pang, Yixuan Zhang, Shaorui Ren, and Shu Shi. 2024. Twist: A Multi-site Transmission Solution for On-demand Video Streaming. *Proceedings of the ACM on Networking* 2, CoNEXT2 (2024), 1–19.

[57] Xiaoliang Wang, Penghui Mi, Yong Zhu, Baoyi An, Yinhua Wang, Lixiang Wang, Xuezhi Yu, Qiong Xie, Xiang Huang, Mingliang Yin, et al. 2024. EdgeCross: Cloud Scale Traffic Management at Peering Edges. *Proceedings of the ACM on Networking* 2, CoNEXT4 (2024), 1–23.

[58] Yifan Wang, Minzhao Lyu, and Vijay Sivaraman. 2024. Characterizing User Platforms for Video Streaming in Broadband Networks. In *Proceedings of the 2024 ACM on Internet Measurement Conference*. 563–579.

[59] Yizong Wang, Dong Zhao, Chenghao Huang, Fuyu Yang, Teng Gao, Anfu Zhou, Huanhuan Zhang, Huadong Ma, Yang Du, and Aiyun Chen. 2023. TrafAda: Cost-Aware Traffic Adaptation for Maximizing Bitrates in Live Streaming. *IEEE/ACM Transactions on Networking* (2023), 1–14. doi:10.1109/TNET.2023.3285812

[60] Dehui Wei, Jiao Zhang, Xiang Liu, Haozhe Li, Zhichen Xue, Tao Huang, Linshan Jiang, and Jialin Li. 2025. QoE-Optimized MultiPath Scheduling for Video Services in Large-Scale Peer-to-Peer CDNs. *IEEE Journal on Selected Areas in Communications* (2025).

[61] Sarah Wooders, Shu Liu, Paras Jain, Xiangxi Mo, Joseph E. Gonzalez, Vincent Liu, and Ion Stoica. 2024. Cloudcast: High-Throughput, Cost-Aware Overlay Multicast in the Cloud. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 281–296. https://www.usenix.org/conference/nsdi24/presentation/wooders

[62] Sarah Wooders, Shu Liu, Paras Jain, Xiangxi Mo, Joseph E Gonzalez, Vincent Liu, and Ion Stoica. 2024. Cloudcast:{High-Throughput},{Cost-Aware} Overlay Multicast in the Cloud. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 281–296.

[63] Bingyang Wu, Kun Qian, Bo Li, Yunfei Ma, Qi Zhang, Zhigang Jiang, Jiayu Zhao, Dennis Cai, Ennan Zhai, Xuanzhe Liu, et al. 2023. XRON: A Hybrid Elastic Cloud Overlay Network for Video Conferencing at Planetary Scale. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 696–709.

[64] Menglu Yu, Ye Tian, Bo Ji, Chuan Wu, Hridesh Rajan, and Jia Liu. 2022. Gadget: Online resource optimization for scheduling ring-all-reduce learning jobs. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 1569–1578.

[65] Sanshi Yu, Zhuoxuan Jiang, Dong-Dong Chen, Shanshan Feng, Dongsheng Li, Qi Liu, and Jinfeng Yi. 2021. Leveraging tripartite interaction information from live stream e-commerce for improving product recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3886–3894.

[66] Rui-Xiao Zhang, Haiping Wang, Shu Shi, Xiaofei Pang, Yajie Peng, Zhichen Xue, and Jiangchuan Liu. 2024. Enhancing Resource Management of the World's Largest {PCDN} System for {On-Demand} Video Streaming. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 951–965.

[67] Rui-Xiao Zhang, Changpeng Yang, Xiaochan Wang, Tianchi Huang, Chenglei Wu, Jiangchuan Liu, and Lifeng Sun. 2022. AggCast: Practical Cost-effective Scheduling for Large-scale Cloud-edge Crowdsourced Live Streaming. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3026–3034.

[68] Yixin Zhang, Yong Liu, Hao Xiong, Yi Liu, Fuqiang Yu, Wei He, Yonghui Xu, Lizhen Cui, and Chunyan Miao. 2023. Cross-domain disentangled learning for e-commerce live streaming recommendation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2955–2968.

[69] Zheng Zhang, Ming Zhang, Albert G Greenberg, Y Charlie Hu, Ratul Mahajan, and Blaine Christian. 2010. Optimizing Cost and Performance in Online Service Provider Networks.. In *NSDI*. 33–48.

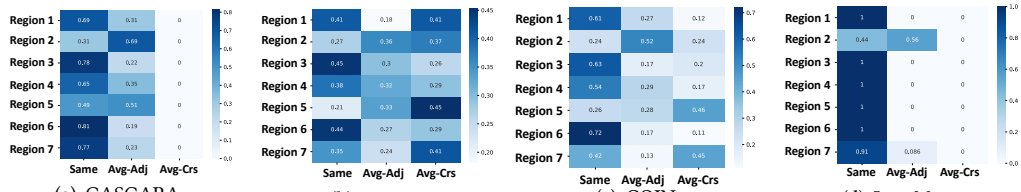
[70] Gongming Zhao, Jingzhou Wang, Hongli Xu, and Zhuolong Yu3 Chunming Qiao. 2023. COIN: Cost-Efficient Traffic Engineering with Various Pricing Schemes in Clouds. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE.

[71] Mingchen Zhao, Paarijaat Aditya, Ang Chen, Yin Lin, Andreas Haeberlen, Peter Druschel, Bruce Maggs, Bill Wishon, and Miroslav Ponec. 2013. Peer-assisted content distribution in akamai netsession. In *Proceedings of the 2013 conference on Internet measurement conference*. 31–42.

[72] Jiangchen Zhu, Kevin Vermeulen, Italo Cunha, Ethan Katz-Bassett, and Matt Calder. 2022. The best of both worlds: high availability CDN routing without compromising control. In *Proceedings of the 22nd ACM Internet Measurement Conference*. 655–663.

Appendix

A Trace-driven Evaluations under Imbalance Scenarios



(a) CASCARA (b) Entact (c) COIN (d) LiveMap

Fig. 14. Performance comparison in terms of Avg-Adj and Avg-Crs.

In this section, we evaluate the performance of baselines under imbalance supply-demand scenarios and we list the evaluation metrics used in the experiments as follows:

- Avg-Adj. We calculate the traffic ratio scheduled to the nodes in CRS regions every day and take the average monthly, which is denoted as Avg-Adj. The objective is to keep Avg-Adj as low as possible.
- Avg-Crs. We calculate the traffic ratio scheduled to the nodes not in CRS regions every day and take the average monthly, which is denoted as Avg-Crs. The objective is to avoid generating Avg-Crs.

Figure 14 shows the performance comparison in terms of Avg-Adj and Avg-Crs. We calculated the percentage of demand scheduled for each region to local nodes, adjacent nodes, and non-adjacent nodes under three baseline methods. LiveMap generates the least Avg-Adj and Avg-Crs traffic, effectively limiting the quality impact caused by cross-regional scheduling. As CASCARA is only allowed to select nodes from the nodes in the local region and adjacent regions in our experiments when allocating bandwidth, it generates smaller Avg-Crs traffic compared to the other two methods. However, the Avg-Adj traffic is much higher than Cesium. COIN overlooks quality considerations, resulting in both high Avg-Adj and Avg-Crs traffic. Entact favors cheaper nodes in non-adjacent regions over slightly more expensive nodes in adjacent regions, leading to a significant amount of cross-regional traffic.

B Details of Maximum-Flow Problem

B.1 Definition for Maximum-Flow Problem

In this section, we briefly introduce the definition of the Maximum-Flow Problem [30].

Flow Networks. Formally, we say that a flow network is a directed graph $G = (V, E)$ with the following features:

- Associated with each edge e is a capacity, which is a non-negative number that we denote c_e.
- There is a single source node $s \in V$.
- There is a single sink node $t \in V$.

Nodes other than s and t will be called internal nodes.

Defining Flow. Next we define what it means for our network to carry traffic, or flows. We say that an $s-t$ flow is a function f that maps each edge e to a non-negative real number, $f : E \leftarrow R^+$; the value $f(e)$ intuitively represents the amount of flow carried by edge e. A flow f must satisfy the following two properties:

- (Capacity conditions) For each $e \in E$, we have $0 \leq f(e) \leq c_e$.
- (Conservation conditions) For each node v other than s and t, we have $\sum_{e\ into\ v} f(e) = \sum_{e\ out\ of\ v} f(e)$

The value of a flow f , denoted $v(f)$, is defined to be the amount of flow generated at the source: $v(f) = \sum_{e\ out\ of\ s} f(e) = \sum_{e\ into\ of\ t} f(e)$

Given a flow network, a natural goal is to arrange the traffic so as to make as efficient use as possible of the available capacity. Thus, the basic algorithmic problem we consider is the following: Given a flow network, find a flow of maximum possible value.

B.2 Modeling the S-D Matching as a Maximum-Flow Problem

We model our S-D Matching in Section 4.1 as a Maximum-Flow Problem. Let R represent the number of regions and $D(r_i)$ and $S(r_i)$ represent the traffic demand and bandwidth supply of the region r_j respectively. We first construct a directed acyclic graph (DAG) to capture the relationship between regional supply and demand (see Figure 6(a)). This allows us to model the matching between the supply and demand. The graph comprises $2 + 2R$ nodes, *i.e.* two virtual nodes as sink node and source node and one demand node and one supply node for each region. There is an edge from supply node of r_i to the demand node of r_j if $r_i \in CRS(r_j)$. Every supply node is connected to the source node and every demand node is connected to the sink node. We then model the supply scheduling problem as follows:

- We define a mapping function f from each edge e_{ij} to a non-negative real number, $f : E \rightarrow R^+$. (*i*) When neither i nor j is the sink or source, $f(e_{ij})$ represents the demand of r_j served by edge nodes in region r_i. (*ii*) When i is the source node, $f(e_{si}) = \sum_{r_j \in OS(r_i)} f(e_{ij})$. Note, $OS(r_i) = \{r_j | r_i \in CRS(r_j)\}$ represents the set of regions having r_i in its candidate region set. (*iii*) When j is the sink node, $f(e_{jt}) = \sum_{r_i \in CRS(r_j)} f(e_{ij})$. According to the definition, the *conservation conditions* is satisfied.
- We further introduce the capacity c_{ij} for each edge e_{ij} and $f(e_{jt}) \leq c_{ij}$ is required to satisfy the *capacity conditions*. To ensure that the serving demand of edge nodes in region r_i not exceed the supply of r_i, we set $c_{si} = S(r_i), \forall i$ and $c_{ij} = S(r_i), \forall i, j \neq t$. To ensure that the total served demand does not exceed the actual demand, we set $c_{it} = D(r_i), \forall i$.

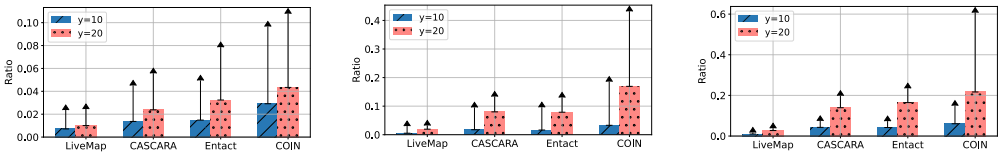
The above problem is essentially a Maximum-Flow problem [30]. Once we find a f that maximizes $\sum_i f(e_{si})$, we obtain the solution that maximize the demand severd within CRS.

C Details of the Live Streaming System

In this section, the live streaming process starting from the request initialized by the clients is detailed, along with the necessary supporting services within the live streaming system. This clarification aims to help understand the significance of LiveMap in such systems. Note that DNS is only used to resolve the returned domain (if any) to an IP address and does not influence the mapping decision. This design gives LiveMap precise control over edge node selection and enables fine-grained request mapping and bandwidth planning.

The primary components of a typical live streaming system is outlined in Figure 16. The workflow for users watching live streams can be summarized as follows:

(*i*) Request Initiation: The client initiates a gRPC request to the Access Gateway for the IP address of an edge node that will provide the live content.



(a) x=5 (b) x=10 (c) x=15

Fig. 15. Impact of prediction inaccuracies on cost.

(*ii*) Stream ID and User Group ID Formation: The Access Gateway derives the Stream ID from the Channel ID and the user's requested resolution. It also determines the User Group (UG) ID based on the client's IP prefix.

(*iii*) & (*iv*) Candidate Edge Nodes Retrieval: The Access Gateway passes the Stream ID and UG ID to the Service Gateway, which proceeds to communicate with the Streaming Center to request a list of candidate edge nodes that are capable of streaming the content. The Streaming Center queries the SIB, which is managed by LiveMap, to identify the most suitable Edge Node for streaming.

(*v*) Edge Node Selection: The Service Gateway selects an Edge Node ID from the list of candidates, based on predefined configurations.

(*vi*) & (*vii*) Stream Pulling: The Access Gateway retrieves the IP address of the chosen Edge Node from the Service Gateway and passes it to the client through gRPC response, which proceeds to pull the stream from the specified media server for viewing.

D Simulation with inaccurate predictions

LiveMap employs two prediction tasks: *traffic demand* in the Bandwidth Provision component and *stream popularity* in the Stream Mapping component. The traffic demand is estimated using a Gradient Boosting Decision Tree (GBDT) model trained on historical, seasonal, and contextual features. The stream popularity is predicted using a Random Forest model [10], leveraging inputs such as views, broadcaster stats, content details, and start time [52, 53]. We select these simple yet efficient models because of their relatively low processing time, less than 1ms for tens of thousands of streams in practice. We note that because of the extreme dynamics of traffic demand and stream popularity, it is difficult to obtain a higher prediction accuracy. While unpredictable traffic surges remain inevitable, reliable planning updates and online allocation mitigate their impact, leaving the pursuit of more accurate predictions for future work.

To assess the robustness of LiveMap against the inaccuracy of traffic demand predictions, we simulated various scenarios with random disturbances in the real-time bandwidth of nodes and analyzed the resulting cost variations. Specifically, we introduced disturbances by selecting $x\%$ of nodes at random and causing their real-time bandwidth to increase by $y\%$ of the scheduled bandwidth. We conducted six sets of experiments, varying x (5, 10, and 15) and y (10 and 20), with eighty simulations for each parameter configuration.

Figures 15(a) to 15(c) illustrate the cost increases relative to the oracle under different disturbance scenarios. The results show that when $x = 15$, the cost variation is notably higher than when $x = 5$ or $x = 10$, but it remains within 5%, highlighting LiveMap's strong resilience to deviations. This robustness stems from LiveMap's ability to online update the billable bandwidth of nodes by both regional coordination and node-level coordination. Similarly, CASCARA and Entact exhibit relatively low sensitivity to bandwidth deviations, as their heuristic online algorithms adaptively select nodes with lower costs in real time. In contrast, COIN shows greater sensitivity to disturbances, underscoring the importance of real-time adjustments for effective cost optimization. In summary, LiveMap 's design effectively absorbs prediction errors, keeping cost variation low by online coordination.

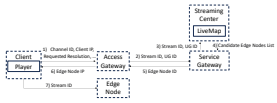


Fig. 16. Workflow of stream requests.

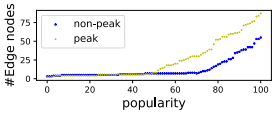
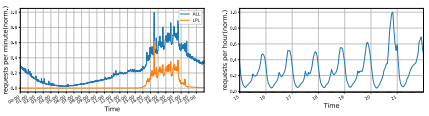


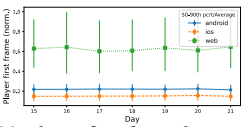
Fig. 17. The number of mapping nodes for different popularity streams.



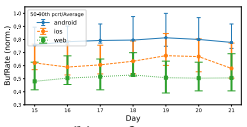
(a) Requests volume per minute on April 20.

(b) Requests volume per hour from April 15 to 21.

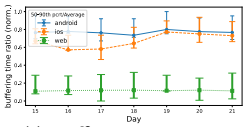
Fig. 18. Normalized requests volume



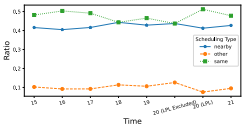
(a) Player first frame latency.



(b) BufRate.



(c) Buffering time ratio.



(d) The ratio of requests scheduled to the same/nearby/other region.

Fig. 19. Comparison results between the LPL Sping Final event and other normal days. The data is normalized by the corresponding maximum value.

E Additional Experiments under a Large Scale Deployment

E.1 Evaluation of the Stream Delivery Module.

We evaluate the effectiveness of the Stream Delivery Component by analyzing the number of edge nodes serving each stream. To better illustrate the results, streams with a maximum viewership of fewer than 10 concurrent viewers are placed in a separate group, while the remaining streams are grouped by their ascending popularity in deciles. For each group, we calculate the average number of edge nodes serving each stream.

Figure 17 presents the average number of edge nodes in each group during two time periods: peak hours (20-22 pm), and non-peak hours (9-11 am). During evening peaks, the Dynamic Resource Adaption component conduct scales-in operations on popular streams less frequently as the Online Allocation component expands node bandwidth beyond billable limits, resulting in more edge nodes serving individual streams compared to the other period. This adaptability ensures optimal resource utilization and a smooth viewing experience, even as stream numbers and bandwidth budgets fluctuate.

E.2 Case Study

We present a brief case study of the League of Legends Pro League (LPL) 2024 Spring Finals, a major live-streaming event held from 16:30 to 21:00 on April 20, 2024 that created a significant spike in requests (40%) on our platform (see Figure 18). The performance metrics include: (*i*) Startup delay that refers to time between the first frame that is displayed by the player after starting the playback; (*ii*) BufRate; (*iii*) Buffering time ratio: the fraction of the total session time spent in buffering. Figure 19 reports the results. We observe that despite the substantial increase in demand, there was no noticeable degradation in these metrics. Further, we find a slight increase of the ratio of requests served within the same region (from 43% to 50%) during this events (see Figure 19(d)). The limited impact of this large event stems from the fact that LiveMap can effectively schedules more free intervals of the edge nodes that use 95-th percentile pricing model for accommodating the traffic surge.