Oceanus: Scheduling Traffic Flows to Achieve Cost-Efficiency under Uncertainties in Large-Scale Edge CDNs

CHUANQING LIN[*] and GERUI LV[*], University of Chinese Academy of Sciences, China
FUHUA ZENG[†], HANLIN YANG, JUNWEI LI, XIAODONG LI, and JINGYU YANG, Alibaba Cloud, China
YU TIAN, University of Chinese Academy of Sciences, China
QINGHUA WU[†] and ZHENYU LI, University of Chinese Academy of Sciences, China and Purple Mountain Laboratories, China
GAOGANG XIE[†], University of Chinese Academy of Sciences, China and CNIC, CAS, China

Large-scale edge Content Delivery Networks (CDNs) provide low-latency content access services and suffer from high bandwidth costs. While previous studies have sought to optimize bandwidth costs under percentile billing, the efficacy is compromised due to the pervasive uncertainty inherent in practical systems, including traffic demand dynamics, performance-constrained scheduling bias, and systemic scheduling deviations. Such uncertainties can result in large gaps among optimal, expected, and actual utilization of massive vulnerable and heterogeneous edge nodes. To address these uncertainties, we propose Oceanus, a cost-effective traffic scheduling system for large-scale edge CDN systems. Oceanus decouples the bandwidth planning problem and performs on multiple timescales. In addition, Oceanus coordinates bandwidth planning with flow scheduling through the bidirectional feedback scheme. Oceanus further utilizes nodes with minimal marginal cost to reduce additional bandwidth cost. Extensive experiments in a trace-driven testbed and real-world deployment confirm the effectiveness of Oceanus. Compared to the state-of-the-art scheduling method, Oceanus achieves 79.4% (vs. 51.5%) of optimum cost reduction and reduces 21.4% (vs. 8.1%) bandwidth costs.

CCS Concepts: • **Networks** → **Network management**; **Traffic engineering algorithms**; **Network resources allocation**.

Additional Key Words and Phrases: CDN, Traffic Engineering, Flow Scheduling

ACM Reference Format:
Chuanqing Lin, Gerui Lv, Fuhua Zeng, Hanlin Yang, Junwei Li, Xiaodong Li, Jingyu Yang, Yu Tian, Qinghua Wu, Zhenyu Li, and Gaogang Xie. 2025. Oceanus: Scheduling Traffic Flows to Achieve Cost-Efficiency under Uncertainties in Large-Scale Edge CDNs. *Proc. ACM Netw.* 3, CoNEXT4, Article 36 (December 2025), 25 pages. https://doi.org/10.1145/3768983

1 Introduction

Content Delivery Network (CDN) aims to provide low-latency data access services to geo-distributed client users. Recently, emerging new applications (e.g., live video streaming [24, 59]) have posed various and stringent latency requirements (e.g., ensuring that end-to-end latency is less than 150 ms [33, 40]) to modern CDNs. To ensure the transmission performance meets the service level agreements (SLAs), CDNs tend to deploy massive edge nodes that are much closer to users (i.e., at each city). However, these edge nodes are vulnerable and heterogeneous, with limited bandwidth capacity and hardware performance, thus requiring fine-grained and careful management.

[*]Co-first authors.
[†]Corresponding authors.

ACM 2834-5509/2025/12-ART36
https://doi.org/10.1145/3768983

Concurrently, the data transmission from emerging applications grows rapidly [1], posing higher pressure on bandwidth costs for CDN vendors. In this context, the *scheduling center*, which manages the utilization of edge nodes by assigning user requests to specific nodes, is carrying increasing responsibility for *(i)* ensuring compliance with performance SLAs while *(ii)* minimizing bandwidth costs. Due to the computational complexity of joint optimization, most studies decouple and optimize the two goals sequentially. Among them, many works [7, 14, 45, 52, 58, 64] have focused on reducing bandwidth costs, especially under the percentile billing scheme that serves as the industry standard. This billing scheme typically takes the 95th percentile (P95) of node bandwidth utilization samples (corresponding to 5-minute time slots) as the final billable bandwidth in a billing cycle (e.g., a month).

The key to reducing overall bandwidth costs is to minimize billable (P95) bandwidth peaks and maximize the utilization of the remaining 5% of uncharged time slots (referred to as augmentation) for each node [14, 45](Fig. 1). To achieve this, existing scheduling methods follow a general workflow: *(i) Bandwidth planning:* Initialize the target billable bandwidth for each node based on estimated traffic demands at the beginning of the billing cycle, then heuristically select nodes to augment to meet actual traffic demands in real-time. *(ii) Flow scheduling:* Adjust the scheduling mapping that assigns traffic to nodes that have sufficient bandwidth and meet SLA requirements.

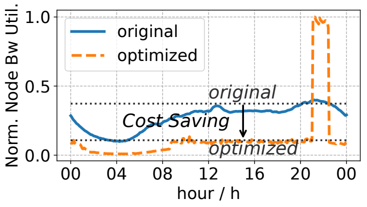


Fig. 1. Example for the 95th billable bandwidth optimization.

However, our empirical evidence shows that *even the state-of-the-art scheduling method can only achieve 51.5% of the theoretical optimum in terms of cost savings* (§2.3), resulting in millions of dollars of additional bandwidth expenditure. The root cause lies in **uncertainties** in real-world edge CDN systems, including: *(i) Traffic demand dynamics:* The traffic demand level can vary widely across billing cycles; *(ii) SLA-constrained scheduling bias:* The subsequent flow scheduling method may actively violate the preceding decision of bandwidth planning to meet SLA requirements; and *(iii) Systemic scheduling deviations:* The scheduled traffic amount may surge and differ from the expected due to the scheduling delay from the time a scheduling decision is made to the time it is executed [28, 48]. Consequently, these uncertainties cause traffic demand estimates to be inaccurate and actual node utilization to be far from expected. Nevertheless, lightweight edge CDN nodes are more vulnerable to the traffic fluctuations induced by uncertainty, and the massive number of edge nodes further amplifies the impact of these fluctuations. While such uncertainty has a substantial impact on merely 0.1% of time slots, it results in considerable cost-saving gaps between the optimal, expected, and actual performance of the scheduling method (§2.3).

To address the impact of uncertainties above, our main idea is to *proactively reduce future uncertainties and reactively adapt to existing uncertainties.* Specifically, we can manage to reduce uncertainty *(i)* and *(ii)*, and strategically adapt to uncertainty *(iii).* However, the practical implementation of this straightforward idea poses additional challenges. Firstly, to track traffic demand dynamics, bandwidth plans must be re-optimized frequently, but this is impractical due to the high computational hardness of Mixed Integer Linear Programming (MILP) with $O(10^7)$ parameters. Secondly, preceding efforts to reserve a fixed proportion of bandwidth buffers cannot efficiently reduce the SLA-constrained scheduling bias, because the buffers are insufficient in urgent 0.1% time slots, but wasteful in most cases. Thirdly, to adapt to the inevitable systemic scheduling deviation, free slots of nodes can be prematurely exhausted, which limits the future selection space of augmented nodes for mitigating additional bandwidth cost.

This paper presents Oceanus, a cost-effective scheduling system for practical edge CDNs. Oceanus decouples and simplifies the bandwidth planning problem, thereby performing it on **multiple**

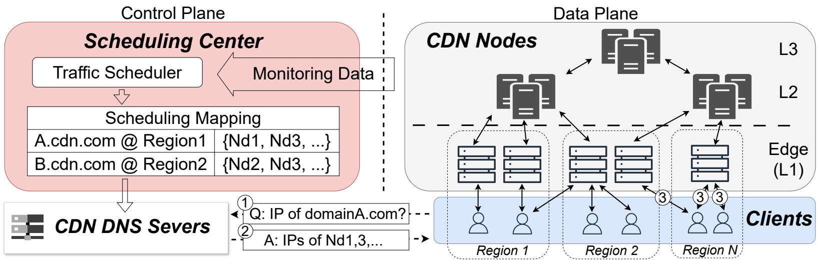


Fig. 2. The architecture and workflow of edge CDN systems.

timescales to track the dynamic *(i)* at a finer granularity. In addition, Oceanus coordinates bandwidth planning with flow scheduling based on a **bidirectional feedback scheme** (i.e., the planner's suggestion and the scheduler's report) to mitigate the uncertainty *(ii)*. In addition, Oceanus heuristically augments nodes with **minimal marginal cost** in future to reduce the additional bandwidth cost under the uncertainty *(iii)*.

Oceanus has been deployed in one of the leading large-scale edge CDN systems, serving traffic demands at the Tbps scale from thousands of edge nodes. The trace-driven evaluations show that compared to the state-of-the-art method (i.e., Cascara [45]), Oceanus achieves 79.4% (vs. 51.5%) of optimum cost reduction and saves 21.4% bandwidth costs (vs. 8.1%) while maintaining performance SLAs. In real-world evaluations, Oceanus updates scheduling mappings 40.3% faster and ensures 80.3% higher scheduling mapping stability compared to baselines. These results confirm that Oceanus effectively addresses the impact of the three types of uncertainties identified.

In summary, our contributions are as follows:

- We have identified three types of uncertainty in edge CDN systems that prevent existing solutions from achieving the expected cost savings (§2).
- We have designed and implemented Oceanus, a cost-effective traffic scheduling system for large-scale edge CDNs (§3-§5).
- We have verified that Oceanus is effective in achieving significant cost savings under uncertainties, both in trace-driven testbeds and in real-world deployments (§6).

2 Background and Motivation

2.1 Edge CDN Systems

Edge CDN system architecture. CDNs aim to provide low-latency content access services for client users by pre-caching frequently requested content from the original server, thereby shortening the end-to-end transmission distance. As shown in Fig. 2, a CDN system contains three components: *(i)* a centralized scheduling center, *(ii)* Domain Name System (DNS) servers, and *(iii)* distributed CDN nodes. At the control plane, the scheduling center determines scheduling mappings that assign user requests to specific CDN nodes, in order to reduce latency and bandwidth costs. These mapping strategies are subsequently performed by DNS servers. At the data plane, geo-distributed CDN nodes form a hierarchical caching architecture (i.e., L1, L2, and L3 layers) to serve client requests efficiently. All client requests are first processed on L1 nodes, alternatively referred to as edge nodes or points of presence (PoPs). If the target content is cached on the L1 node, the node will respond directly to the client. Otherwise, the L1 node will sequentially fetch the content backward from L2 nodes, L3 nodes, or even original servers.

Unlike traditional CDNs, edge CDNs deploy L1 nodes at much higher densities (e.g., more than 135 per $10^6 mi^2$ [55]) in locations much closer to clients (i.e., in every city) to further improve performance. The trade-off is that these closer L1 nodes typically possess reduced computing, storage, and bandwidth resources. Their bandwidth can be limited to tens of Gbps, which is 1-2 orders of magnitude less than traditional CDN nodes. These nodes also exhibit high heterogeneity in terms of hardware resources. In summary, L1 nodes in edge CDNs are more susceptible to traffic fluctuations and face greater challenges in fulfilling SLAs of various applications.

Request scheduling workflow. Client users find the edge CDN node to request through the DNS system. As shown in Fig. 2, when a client desires content from a specific domain name, it first issues a query to its local DNS (LDNS) resolver requesting IP addresses of the domain name. The LDNS forwards the query to DNS servers (①). CDN's DNS servers respond with IP addresses of all available nodes by searching the scheduling mapping (②). The client will request the content using one of the IP addresses replied by LDNS (③).

Scheduling mapping. The *scheduling mapping* is controlled by the scheduling center and represents the DNS servers' strategy. Each entry of the scheduling mapping specifies the assigned node for a *flow*, a group of content requests that forms the finest scheduling unit. During each time slot, the scheduling center adjusts the scheduling mapping based on real-time network performance and traffic demand data. In this way, the CDN system can control the nodes to which traffic demands are directed, as well as the bandwidth utilization of each node.

2.2 CDN Scheduling Problem

By performing the scheduling mapping, the scheduling system aims to achieve two goals: *(i) Primary goal*: guaranteeing that all flows' transmission performance (e.g., RTT) meets the corresponding SLAs; *(ii) Secondary goal*: minimizing the overall bandwidth costs of CDN nodes.

The challenge in achieving these two goals simultaneously is that each has a huge individual solution space. Specifically, the expected bandwidth cost is calculated at the beginning of a billing cycle (typically a month or 30 days), considering all edge CDN nodes (2,000+) in all time slots (typically 5 minutes; $30 \times 24 \times 60/5 = 8640$ slots for a 30-day billing cycle). On the other hand, SLA assurance considers all flows (10,000+) in each time slot. Therefore, it is infeasible to solve this multi-objective optimization problem at the minute level.

General workflow. Most previous studies separate the two goals into independent decision logic and on different time scales [7, 14, 45, 52, 58]. The common practice is: *(i)* Bandwidth planning: Minimize bandwidth costs by setting bandwidth budgets for edge CDN nodes based on estimated traffic demand. *(ii)* Flow scheduling: Meet SLA requirements for real-time traffic demands by providing adequate bandwidth from CDN nodes in each fine-grained time slot. Note that step *(ii)* makes a "best effort" to limit the bandwidth usage within the budgets from step *(i)*, but may still result in a cost increase from expected due to unpredictable traffic demand dynamics and systemic deviation [7, 14, 48].

CDN bandwidth costs. CDN vendors need to pay for the bandwidth utilization of their nodes, which are typically built by or connected to other ISPs. The bandwidth cost of a node in a billing cycle is given by $cost = f(\max\{b_{in}, b_{out}\})$, where b_{in} and b_{out} are the billable bandwidth of inbound traffic (from clients to nodes) and outbound traffic (from nodes to clients), and the billing function f subsequently computes the final cost. Billing functions include linear functions, multi-tier linear functions, regional or time-varying billing functions, and fixed costs (where charges remain constant regardless of utilization) [64]. In this work, we consider the widely adopted linear billing function. Since *outbound traffic* dominates the content delivery scenario [45, 52], the cost can be further simplified to $p \cdot b_{out}$, where p is the bandwidth unit price (e.g., in \$/Mbps) of a node.

Widely utilized methodologies for calculating the billable traffic b_{out} are: *(i) percentile-volume* billing and *(ii) average-volume billing*. Given node bandwidth utilization samples recorded for each time slot within a billing cycle, percentile-volume billing uses the specific percentile of all samples as the node's billable bandwidth. In contrast, average-volume billing employs the mean

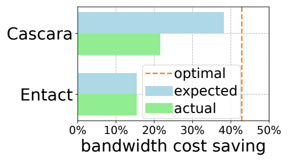


Fig. 3. The bandwidth cost saving achieved.

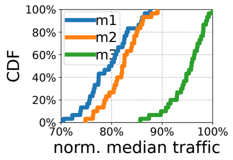


Fig. 4. CDF of daily traffic over three months.

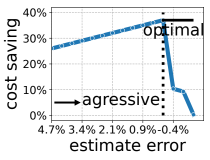


Fig. 5. Impact of estimate error.

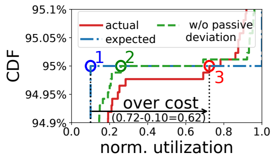


Fig. 6. Utilization discrepancies incur additional costs.

bandwidth utilization volume, which is similar to total-volume billing. Another linear billing method, maximum-volume billing, billed by the maximum utilization sample, is also considered in some cases. In this paper, we focus on 95th percentile billing, which has emerged as the predominant approach in production environments and has been proven to be NP-hard to optimize [7, 45, 52]. It is worth noting that practical edge CDN systems consist of nodes with mixed billing schemes. Please refer to discussion §7 for the integrated approach.

Optimization logic. The opportunity to reduce the cost of 95th percentile billable bandwidth lies in the *5% free time slots* [14, 45]. Each CDN node should minimize its 95th percentile bandwidth utilization sample, and increase the 95th to 100th percentile uncharged bandwidth utilization (referred to as *augmentation*). This cost optimization problem can be formulated as a Mixed Integer Linear Programming (MILP) [7, 45, 64]. In the actual solution process, previous works estimate the traffic demand to determine the billable bandwidth for each node in a billing cycle [7, 14, 45, 52]. After that, the scheduler heuristically performs real-time node augmentation to cover dynamic traffic demands in specific time slots.

2.3 Motivation: Pitfall of Existing Solutions

Although bandwidth cost optimization under 95th percentile billing is not a new topic, our analysis based on a large-scale dataset from a leading CDN system shows that: *Even the most advanced solutions struggle to reduce bandwidth costs effectively while meeting SLA requirements due to overlooking uncertainties in the practical scheduling system.*

Dataset description. The dataset contains traffic demand data from a large edge CDN system over three months. This system includes more than 2300 nodes serving Tbps-level traffic demands from 30 regions (i.e., provinces). We further built a testbed system (details in §6.1) to replay the collected data and conduct controlled experiments.

Baseline. We consider two scheduling methods: *(i) Entact* [62], which schedules by simplifying the 95-percentile (non-linear) billing model to a linear one, and *(ii) Cascara* [45], which heuristically augments nodes under performance constraints, representing the state-of-the-art solution. We also compute the *Optimal* cost savings as a theoretical upper bound by performing an offline optimization with full knowledge of the traffic demand in each time slot in the entire billing cycle.

Fig. 3 shows that **huge performance gaps** exist between the *optimal cost* derived from the theoretical upper bound, the *expected cost* believed to be achievable by the scheduling method, and the *actual cost* utilized to meet real-time traffic demands. Specifically, Entact only saves 15.2% of bandwidth costs because its linear billing model assumption does not apply to 95th percentile billing. While Cascara performs much better than Entact with 38.1% expected cost savings, it is still inferior to the optimal with a relative gap of 11%. Even worse, the performance of Cascara further deteriorates when actual traffic demand is considered, achieving only 21.6% actual cost savings. The root causes of these gaps are revealed as follows.

Gap 1: From optimal to expected cost. This gap arises from the *discrepancy in long-term global traffic demand estimation due to its high dynamics.* Here, long-term indicates the billing cycle level (temporal) and global indicates the region level (spatial). Recalling §2.2, the scheduling method minimizes the expected cost based on the estimated traffic demand of the entire billing cycle (month). Most existing solutions (including Cascara) assume that the traffic demand is stable across months. Therefore, they simply use the actual traffic demand from the last month as an estimate of the new month. However, our online measurements (Fig. 4) show that global traffic demand is highly dynamic and can vary widely from month to month.

Traffic demand dynamics (uncertainty *(i)*) directly results in inaccurate estimates, leading to sub-optimal expected costs. Take Cascara as an example, Fig. 5 illustrates the impact of the estimate error on the expected cost savings, where the error is calculated as the value of the estimated minus the actual value divided by the actual value. If the actual traffic demand is underestimated (where the estimate error is lower than 0), the expected cost will be "aggressive" (low). In this case, the scheduling method must augment more nodes to provide more bandwidth for additional traffic demand. If the augmentation slots are prematurely exhausted, the total expected cost will increase rapidly, thereby leaving a gap from the optimal cost.

Gap 2: From expected to actual cost. This gap is attributed to *real-time local discrepancies between the expected and actual bandwidth utilization.* Such discrepancies occur at the time slot and CDN node levels. Fig. 6 showcases how these discrepancies induce significant bandwidth cost increases from the perspective of a single node, where the normalized node utilization is the allocated outbound bandwidth divided by the node bandwidth capacity. The red line and the blue line illustrate the actual and expected bandwidth utilization distribution, respectively. It is noticeable that over-utilizing bandwidth on a node in just 0.1% of all time slots (from 94.9% to 95%, corresponding to about 45 minutes in a month) can result in a substantial rise in actual bandwidth cost (from 0.10 to 0.72). This sharp increase is highlighted at utilization points 1 and 3 in Fig. 6.

Further, these real-time local discrepancies come from two sources of uncertainty. One is *SLA-constrained scheduling bias* (uncertainty *(ii)*). As illustrated in §2.2, since the scheduling system takes performance assurance as its primary goal, it may actively increase bandwidth utilization of high-quality nodes to meet SLA constraints, which is difficult to anticipate by the bandwidth planning. The other is *systemic scheduling deviation* (uncertainty *(iii)*). The actual bandwidth utilization may exceed expectations in the occurrence of unpredictable traffic surges on a node, or due to the scheduling delay from the scheduling decision to its execution [28, 48].

To verify the ratio of impact of systemic scheduling deviation on real-time local discrepancies, we allow the scheduling method to allocate traffic accurately as expected. The utilization line is plotted as the green line in Fig. 6. As highlighted at point 2, the actual bandwidth cost is significantly reduced to 0.26. However, the actual cost remains 2.6 times higher than expected, attributed to the SLA-constrained scheduling bias.

Summary. Our analyses show that existing solutions still suffer from both long-term global and real-time local discrepancies due to three types of uncertainties (*(i)* to *(iii)*) in practical scheduling systems, resulting in significantly increased bandwidth costs under the 95th percentile billing model. Understanding the root causes of these discrepancies provides opportunities either to proactively reduce future uncertainties or to reactively mitigate the impact of existing uncertainties.

2.4 Design Challenges

Designing an applicable system in practice to tackle uncertainties is still challenging.

Challenge 1: *How to reconcile long-term planning with real-time adjustment?* Effective cost optimization under percentile billing necessitates frequent strategy updates to adapt to global traffic dynamics across a month-long cycle. However, the underlying optimization is NP-hard [7, 14], making a full, frequent re-computation across thousands of nodes computationally infeasible. Simultaneously, the system must react to traffic surges and SLA violations at the minute level.

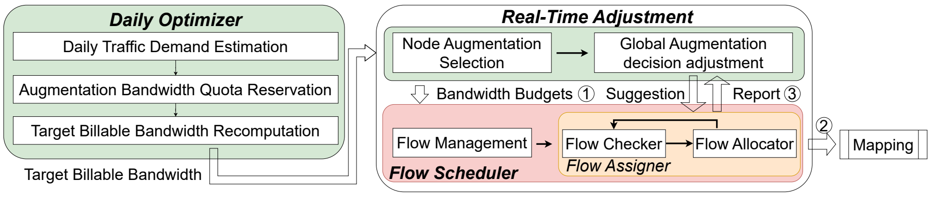

Daily Optimizer
Daily Traffic Demand Estimation
Augmentation Bandwidth Quota Reservation
Target Billable Bandwidth Recomputation
Target Billable Bandwidth
Real-Time Adjustment
Node Augmentation Selection
Global Augmentation decision adjustment
Bandwidth Budgets ①
Suggestion
Report ③
Flow Management
Flow Checker
Flow Allocator
Flow Assigner
Flow Scheduler
②
Mapping

Fig. 7. Oceanus' system design. In each 5-minute cycle, the planner in green provides (1) real-time budgets to the scheduler in red, which (2) updates the scheduling mapping and (3) reports back on budget adherence.

Solution: Oceanus decouples the bandwidth planning problem and executes it across multiple timescales. The system updates the long-term planning strategy daily by solving a computationally lighter linear program (LP), which significantly reduces complexity compared to full re-optimization. The node augmentation decision is offloaded to the real-time adjustment that handles traffic surges.

Challenge 2: *How to coordinate separated components under performance constraints?* The bandwidth planner (which sets bandwidth budgets) and the flow scheduler (which assigns traffic to meet performance SLAs) are logically separated, leading to the SLA-constrained scheduling bias. Jointly optimizing bandwidth cost and performance results in unaffordable computational complexity. Prior efforts to reserve a fixed proportion of bandwidth buffers [7, 14] are inefficient: these buffers are insufficient during urgent traffic spikes in 0.1% slots but are wasteful in most other time slots.

Solution: Oceanus implements a bidirectional feedback scheme that creates a tight, closed-loop control system. Specifically, in addition to the bandwidth budget, the bandwidth planner provides *suggestions* on where to move traffic from over-utilized regions. The flow scheduler reports back to the planner on any regional resource shortages it encountered.

Challenge 3: *How to mitigate the impact of node utilization discrepancies with limited resources?* Systemic scheduling deviations are inevitable. The naive solution is to use the limited augmentation slots to absorb these deviations. However, these limited resources can be quickly and inefficiently depleted, leaving the system vulnerable to future traffic peaks and forcing it to incur high costs.

Solution: Oceanus heuristically augments nodes with minimal marginal cost, ensuring that the most "expensive" free slots are saved for when they are truly needed. This strategy balances the consumption speed of free slots across nodes and reduces additional bandwidth costs when all free slots are eventually exhausted.

3 Oceanus Design Overview

Oceanus is designed to minimize bandwidth costs by traffic scheduling in the face of uncertainties in practical systems while promising that performance SLAs are met. As shown in Fig. 7, Oceanus contains two main components: *the Bandwidth Planner*, responsible for updating bandwidth budgets for nodes (§4), and *the Flow Scheduler*, responsible for updating scheduling mappings for traffic flows (§5) based on the latest bandwidth budgets.

The bandwidth planner operates two key metrics for each node: *the target billable bandwidth*, and the *real-time bandwidth budgets*. The former represents the target value of bandwidth at which a node is expected to be billed ultimately. The latter refers to the bandwidth a node is allowed to utilize at each time slot, which may exceed the former value during augmentation slots.

The bandwidth planner runs on **multiple timescales**. At the beginning of *a billing cycle*, Oceanus first solves an offline MILP problem to initialize the target billable bandwidth for each node (§4.1). At the beginning of *each day*, Oceanus employs a daily optimizer to refine the target billable bandwidth according to the estimated traffic demands on the current day (§4.2). For *each 5-minute time slot*, with the latest traffic demand data, Oceanus augments nodes with the **minimal marginal cost** to allocate additional bandwidth if the total traffic demands exceed the total billable bandwidth (§4.3).

The bandwidth budgets for nodes are determined by combining the target billable bandwidth and the augmentation decisions.

Upon receiving the latest bandwidth budgets and traffic demands, the flow scheduler updates the scheduling mapping every 5 minutes, which specifies the nodes assigned to flows. To achieve this, Oceanus first dynamically manages the granularity of *flows*, which serve as fundamental units of scheduling (i.e., entries in the scheduling mapping), to balance their rates. Simultaneously, the *flow checker* and the *flow allocator* iteratively mark and reassign flows to satisfy all SLA requirements while minimizing node bandwidth budget violations.

The bandwidth planner coordinates with the flow scheduler through the **bidirectional feedback scheme** to eliminate SLA-constrained scheduling bias quickly. Specifically, the flow scheduler reports node overuse and underuse situation to the real-time bandwidth planning module, which globally adjusts the node augmentation decisions accordingly. In turn, the real-time bandwidth planning module provides scheduling suggestions to the flow scheduler for assigning flows from overused nodes to underused nodes.

4 Cost-Driven Bandwidth Planning

The bandwidth planner in Oceanus estimates the billable bandwidth for each CDN node and computes the real-time bandwidth budgets for flow scheduling, which suggests the (ideal) upper bound for resource utilization. Oceanus's bandwidth planner operates on multiple timescales. Specifically, based on the target billable bandwidth estimated at the beginning of the billing cycle, i.e., *every month* (§4.1), Oceanus globally updates target billable bandwidth *every day* (§4.2), and computes the real-time bandwidth budgets *every 5 minutes* (§4.3).

4.1 Target Billable Bandwidth Initialization

At the beginning of a billing cycle, Oceanus initializes a minimum total bandwidth cost by optimizing the target billable bandwidth for each edge CDN node while meeting performance requirements. We formulate this bandwidth cost minimization problem as an *optimization problem*. We introduce the optimization goal and constraints here, and the detailed formulation is shown in Appendix A.

Decision variable. Let N be the set of all nodes in a 95th-percentile-billed edge CDN system. Oceanus determines the initial target billable bandwidth L_n for each node $n \in N$. L_n represents the total billable bandwidth from a node to clients in a billing cycle.

Optimization goal. Oceanus aims to minimize the total bandwidth costs ($\sum_{n \in N} p_n \cdot L_n$) by reducing the initial target billable bandwidth L_n for each node. Here, p_n is the bandwidth unit price for node $n \in N$.

Constraints and definitions. Oceanus considers the following aspects of constraints in the optimization problem:

- *(i) Link Capacity*: The bandwidth budget B_n^t (including augmented bandwidth) of each node in each time slot is limited by its bandwidth capacity C_n, where $t \in T$ is a 5-minute time slot in a billing cycle.
- *(ii) Augmentation Decision*: The augmentation time slots, during which the bandwidth budget B_n^t can exceed the target billable bandwidth L_n, should account for no more than 5% of all time slots (i.e., $|T|/20$).
- *(iii) Traffic Supply and Demand*: Edge CDN nodes provide bandwidth to meet the traffic demands of clients across geographic regions (e.g., provinces). The set of all regions is denoted as R. In time slot $t \in T$, let D_i^t denote the traffic demand of the client region $i \in R$, and X_{ij}^t denote the allocated bandwidth from CDN node in region j to clients in region i (i can be the same as j). It is clear that X_{ij}^t should balance bandwidth supply B_n^t and traffic demand D_j^t.

- *(iv) Performance Assurance*: Oceanus is designed to meet clients' performance requirements (i.e., SLAs) in bandwidth cost optimization by considering only nodes from *candidate region pools (CRPs)*. A CRP of client region $i \in R$ is defined as a set of node regions where RTT to region i does not violate the SLA, namely $CRP_i = \{j \in R | RTT_{ij} \leq SLA_i\}$. [1] Here, RTT_{ij} indicates the RTT between node region j and client region i; $CRP_i \subseteq R$.

Solution. This problem is a *Mixed Integer Linear Program (MILP)*, which is computationally hard (NP-hard) to solve[7, 14, 45, 58, 64]. Fortunately, this problem can be computed offline in advance (e.g., on the last day of a previous billing cycle) and is not time-sensitive. Common optimization solvers, such as GUROBI[15] and CPLEX[10], can first compute the relaxed *Linear Programming (LP)* version of the problem to find the lower bound (the minimum cost), and then manage to reduce the gap between the feasible solutions and the lower bound. We used GUROBI 11 on a 16-core, 64GB RAM machine, setting the optimality gap to 5% and the time limit to 5 hours. Prior works [45, 52] also provided other ideas to speed up the solving process.

Practice. At the beginning of a new billing cycle, the traffic demand D_i^t is unknown. Therefore, Oceanus uses the actual traffic demand in the last billing cycle as an estimate[2], as done in previous work [7, 14, 45, 58]. However, as illustrated in §3, the optimizing performance will be severely degraded if this estimate is inaccurate. To solve this issue, Oceanus introduces the following fine-grained operations, i.e., daily billable bandwidth update (§4.2) and real-time bandwidth budget generation (§4.3).

4.2 Daily Billable Bandwidth Update

The daily update aims to adjust the target billable bandwidth frequently to accommodate the long-term dynamics of traffic demands. Through proactively increasing the billable bandwidth, the rate of augmentation bandwidth consumption can be reduced, preventing early depletion. To achieve this, Oceanus first estimates the current day's traffic demands based on the actual traffic demands of previous days. Oceanus also calculates the augmentation bandwidth quota that can be utilized today. Then, the portion of traffic demands that can be served by augmented bandwidth is separated from the overall traffic demands. Finally, target billable bandwidths are recomputed using linear programming to cover the remaining traffic demands from a global perspective. The multi-step algorithm is outlined in Alg. 1 in Appendix A.

Daily traffic demand estimation. To provide sufficient resources, Oceanus needs to estimate today's traffic demands. As traffic demands exhibit strong daily and weekly seasonality, the predictor employs two distinct daily traffic demand models for each region to track different patterns on weekdays and weekends. At the start of each day d, the corresponding traffic demands estimation series $\hat{D}_{i,d}$ for region $i \in R$ is updated using the previous day's actual regional traffic demand series $D_{i,d-1}$ through seasonal exponentially weighted moving averages (EWMA) with a one-day lag:

$$\hat{D}_{i,d}^{new} \leftarrow \alpha \cdot D_{i,d-1} + (1-\alpha) \cdot \hat{D}_{i,d}^{old} \tag{1}$$

where we set $\alpha = 0.5$ to balance the ability to track traffic trends and filter out irregular surges. This method is simple yet robust and effective, as shown in §6.2.

Augmentation bandwidth quota reservation. Given estimated traffic demands, Oceanus needs to determine how much demand should be served by node augmentation today. Therefore, Oceanus begins with *(i)* calculating the remaining bandwidth quota for augmentation ($DailyQuota$)

[1]Here we take latency as the SLA requirement. Other performance requirements, such as throughput and loss rate, can also be incorporated into this definition. For example, to ensure the throughput to region i is at least $Thresh_i$, we can define $CRP_i = \{j \in R | Thrpt_{ij} \geq Thresh_i\}$.

[2]Oceanus estimates D_i^t at the region level rather than at the flow level, because the traffic demand of a specific flow varies drastically due to human factors and is therefore difficult to predict.

for today, based on historical bandwidth consumption in this billing cycle. Specifically, Oceanus identifies historical time slots where the bandwidth budget B_n^t exceeded the latest target billable bandwidth L_n' as utilized augmentation slots (line 4). Then Oceanus divides the remaining augmentation slots evenly over the remaining days (lines 5-7).

After that, Oceanus *(ii)* allocates the augmentation bandwidth quota $DailyQuota$ to the estimated traffic demand $\hat{D}_i$, to calculate the traffic covered by the billed bandwidth ($NormalTraf_i$). In detail, as shown in Fig. 8, Oceanus performs a binary search to identify the minimum percentile q and corresponding $NormalTraf_i$ at each region i, ensuring that the sum of traffic demands exceeding $NormalTraf_i$ (green portion) does not exceed the augmentation bandwidth quota $DailyQuota$ (lines 8-9).

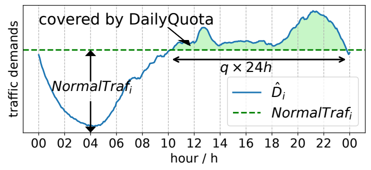


Fig. 8. An example of augmentation bandwidth quota reservation.

Target billable bandwidth recomputation. Based on the billed bandwidth $NormalTraf_i$, Oceanus can update the target billable bandwidth L_n. As illustrated in Eqs. 12-16, the optimization process is similar to the one in §4.1, where traffic demands D_i^t are substituted with $NormalTraf_i$. To reduce the computational complexity, Oceanus avoids selecting specific CDN nodes for augmentation at this stage and leaves this decision to the subsequent real-time process (§4.3). As a result, the daily billable bandwidth update is formulated as an LP problem and can be solved in seconds. In the case of a significant traffic surge or the integration of newly constructed nodes in the middle of a billing cycle, Oceanus also prioritizes the urgent global re-run of this LP problem.

4.3 Real-time Node Augmentation

When the real-time traffic demands exceed the total billable bandwidth, Oceanus must allocate additional bandwidth via node augmentation. The node augmentation decision should satisfy three requirements: *(i)* the consumption of augmentation slots across nodes should be balanced; *(ii)* the additional bandwidth cost can be minimized even when all nodes' remaining augmentation slots are exhausted; *(iii)* the augmentation selection process should account for performance assurance, i.e., prioritizing bandwidth allocation in regions facing resource shortages.

Oceanus maintains a priority queue to order nodes according to their utilization history. Oceanus first augments nodes with the highest priority to provide sufficient bandwidth to cover traffic demands. Next, the augmentation decision is adjusted globally based on the bandwidth utilization *report* provided by the flow scheduler. Oceanus also generates scheduling *suggestions* to guide the scheduler in rescheduling flows from nodes at resource-scarce regions to nodes at resource-rich regions. Finally, the bandwidth budgets, which denote the amount of bandwidth available for nodes, are output to the flow scheduler. The node augmentation algorithm is summarized at Alg. 2 in Appendix A.

Node augmentation. At each time slot, Oceanus first checks whether node augmentation is necessary. When the total billable bandwidth (i.e., *GivenResources*) cannot cover all traffic demands (line 1-5), it starts to augment several nodes to allocate additional bandwidth. Oceanus maintains the node priority queue using two metrics: *the marginal cost* (MC_n) and the augmented slots (line 6-7). The augmented slots refer to the number of historical time slots where the actual utilization U_n^t exceeded the target billable bandwidth L_n. The marginal cost represents the additional bandwidth cost incurred for augmenting a node by one more time slot:

$$MC_n = p_n \cdot \left(P_{95}(\{\cdots, U_n^{t-1}, \mathbf{C}_\mathbf{n}^\mathbf{t}, L_n^{t+1}, \cdots\}) - P_{95}(\{\cdots, U_n^{t-1}, \mathbf{L}_\mathbf{n}^\mathbf{t}, L_n^{t+1}, \cdots\})\right) \quad (2)$$

where p_n denotes the unit price of node n, $P_{95}(\cdots)$ computes the 95-percentile value for a sequence. In the utilization sequence, U_n represents the historical utilization samples from time slot 1 to $t-1$, and the future utilization samples from time slot $t+1$ to the end of the billing cycle are estimated by the billable bandwidth L_n. At the current time slot t, the former sequence assumes the node is augmented with utilization to its capacity C_n, while the latter sequence assumes the node is not augmented with utilization equal to the target billable bandwidth L_n.

Oceanus prioritizes augmenting nodes with the smallest marginal cost and the least augmented slots (line 11). Early in the billing cycle, when nodes still have available augmentation slots, the marginal costs are equal to 0. Oceanus can sequentially augment the node with the least augmented slots. After all free augmentation slots are exhausted, Oceanus can supply sufficient bandwidth with minimal additional bandwidth cost.

Global augmentation decision adjustment. Despite sufficient bandwidth being supplied globally, local resource shortages can still occur. This is because the flow scheduler adjusts scheduling mappings locally and fails to find available bandwidth resources for traffic demands from region i from nodes located at regions in CRP_i. As a result, the flow scheduler is forced to allocate traffic demands exceeding the bandwidth budget on certain nodes to ensure performance SLAs. To reduce the violation of bandwidth budgets, Oceanus adopts a feedback mechanism to address the local resource shortage promptly. The report F_i reports the bandwidth misalignment between the actual serving bandwidth and the bandwidth budgets for nodes in region i. For example, a positive F_i indicates a bandwidth shortage in region i (i.e., certain nodes are over-utilized), whereas a negative F_i indicates a bandwidth surplus.

Oceanus adjusts the augmentation decisions in two steps: *(i)* augmenting more nodes at regions within CRP_i for each bandwidth shortage region i, until all bandwidth shortage is fulfilled (line 14-18); *(ii)* recycling augmentation bandwidth from regions with a bandwidth surplus (line 20-23). Specifically, Oceanus can stop augmenting a node n in region i, if there exists a region j in CRP_i that has sufficient bandwidth surplus to accommodate the additional traffic demands from region i after stopping augmenting node n. This excessive deployment and conservative recycling strategy can effectively avoid node augmentation oscillation.

Oceanus also generates the suggestion H^t for the next round of flow mapping. For example, H^t_{ij} represents the suggested amount of traffic demands to be rescheduled from region i to region j, as either region i experiences a bandwidth shortage or its augmentation bandwidth is recycled.

Final output. The real-time node bandwidth budgets are finally computed. Specifically, if the node is selected for augmentation, the bandwidth budget is set to its capacity. Otherwise, the bandwidth budget is configured to its target billable bandwidth (line 25-28). Such bandwidth budgets guide the flow scheduler (§5) to assign flows to nodes.

5 Performance-Oriented Flow Scheduling

Given the bandwidth budgets generated by the bandwidth planner, the flow scheduler determines scheduling strategies for all flows, the basic units of traffic demand. As shown in Fig. 7, within each 5-minute time slot, the *flow manager* first dynamically aggregates traffic demands into uniformly-sized *flows*, to reduce the scale of the scheduling problem. The *flow assigner* then scans and adjusts the scheduling mapping, ensuring that all performance SLAs are considered as hard constraints, while making best efforts to implement node bandwidth budgets as the optimizing goal.

5.1 Flow Management

Oceanus aggregates traffic demands at the *flow* level rather than the finest granularity to reduce storage and compute overhead [6, 19, 38, 61]. Specifically, an original flow is defined by: *(i)* the requested domain name, which indicates the application types and performance requirements, and *(ii)* the LDNS IP address, which identifies the source location. Requests inside an original flow share similar resource requirements (e.g., CPU, memory, bandwidth, and network performance).

Flows must be appropriately divided or aggregated to balance their traffic demand and quantity for efficient scheduling. On one hand, scheduling flows at a finer granularity provides precise control, which can ensure more traffic meets SLA and reduce node over-utilization events. However, this creates an unmanageable online scheduling problem with millions of flows that is challenging to execute within the required five-minute epoch. On the other hand, it is necessary to divide the giant flow (e.g., those exceeding 100Gbps) to assign them across lightweight nodes of edge CDNs. A giant, undivided flow can frequently exceed a single node's capacity or bandwidth budget, leading to repeated rescheduling. This frequent reassignment destabilizes node utilization and is cache-unfriendly, ultimately degrading performance.

To achieve this goal, the flow manager elastically controls the granularity of flows. The number and rate of flows can be expanded or contracted vertically or horizontally. As the traffic demand of a flow increases or decreases, the flow rate will first expand or contract vertically. When the traffic demand of a flow reaches the limit of horizontal expansion or contraction, the flow is split or aggregated with other flows. The limit for expansion and contraction is elastic to prevent frequent flow reconstruction. For instance, when bandwidth demand exceeds 10 Gbps, the flow is subdivided into two smaller flows. These flows are only aggregated when the total bandwidth demand decreases below 8Gbps.

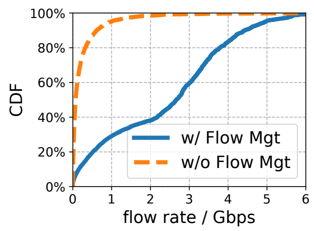


Fig. 9. Distribution of flow rate.

Fig. 9 illustrates the distribution of flow rate. Compared to scheduling at the finest granularity, the flow manager reduces the scale of flows by 80% and balances their resource demands. The coefficient of variation (CV) of flow rates drops from 2.10 to 0.67.

5.2 Flow Assignment

The flow assignment problem is typically a bin-packing problem, which is challenging to solve due to the indivisibility of flows [7, 14, 58]. In edge CDN systems, the flow assigner must use a lightweight algorithm to update the scheduling mapping in under 5 minutes, given the enormous search space of over 10 million possible <flow, node> pairs.

To narrow the search space and accelerate updates, Oceanus adopts a local heuristic reassignment approach, i.e., only reassigns necessary flows to address performance or bandwidth constraints. Specifically, the flow assigner consists of two modules: the *flow checker* and the *flow allocator*. The former repeatedly scans all nodes and flows to identify flows requiring reassignment. Guided by the *suggestion* provided by the bandwidth planner, the latter iteratively reassigns marked flows to new nodes and provides *report* back to the bandwidth planner, reporting real-time regional bandwidth shortages or surpluses. The flow reassignment algorithm is summarized in Alg. 3 in Appendix A.

Flow checker. The flow checker performs two types of checks on the current assignments. *(i)* At the start of each time slot, it marks all flows whose real-time collected performance violates the corresponding SLA (taking latency as an example in line 3). *(ii)* It continuously scans all nodes to verify if bandwidth utilization exceeds the allocated bandwidth budget. If any node is over-utilized, the checker marks the minimum number of flows on that node for reassignment to reduce utilization below the budget (lines 7–10). To account for passive scheduling deviations, Oceanus limits the bandwidth utilization of a node to ηB_n^t, where η is an annealing factor. This factor gradually increases from 0.95 to 1 during each time slot.

Flow allocator. The flow allocator reassigns each marked flow s through the following steps.

- *Candidate Node Pool Construction*: The allocator identifies all available nodes in regions where the corresponding network performance to the flow's source region i meets SLA, forming the candidate node pool (i.e., $NodePool_s = \{n \in N_j | j \in CRP_i\}$) (line 14). Here, CRP_i denotes the set of node regions where performance to region i meets SLA (§4.1).

- *Normal Reassignment*: The flow is assigned to the node in the pool with the largest remaining bandwidth (i.e., $\eta B_n^t - U_n^t$, lines 15–17).
- *Bandwidth Takeover*: If no node has sufficient remaining bandwidth, the allocator allows the flow to take over bandwidth from lower-priority flows, and reassigns these affected flows afterward (lines 18–25). A low-priority flow is characterized by its relaxed SLA, which allows allocating resources more easily from a wider candidate node pool. The order of regions to be taken over is determined by the bandwidth planner's suggestion (§4.3), prioritizing regions with the greatest surplus bandwidth.
- *Violation Handling*: If the assignment cannot be completed without violating bandwidth budgets, the allocator assigns the flow to the last resort node in the candidate node pool with the largest remaining bandwidth (i.e., $B_n^t - U_n^t$) to minimize the violation (line 26). Nonetheless, the SLA requirement for the flow is still satisfied.

Report computation. In practice, forced bandwidth budget violations do occur, though most last less than 30 minutes and are self-rectifying. To balance bandwidth provision across regions and promptly eliminate such violations, the allocator computes the bandwidth provision and utilization for each region (line 29). This information is then used by the bandwidth planner to customize augmented node selection (§4.3), optimizing the use of limited augmentation bandwidth.

Final output. The updated scheduling mapping reflects the latest strategy for assigning flows to specific nodes. This mapping is incrementally distributed to all DNS servers for deployment.

6 Evaluation

We evaluate the performance of Oceanus in a large-scale edge CDN. First, we demonstrate that Oceanus achieves significant bandwidth cost savings (§6.1). Next, we analyze how Oceanus handles three types of uncertainties (§6.2). Finally, we evaluate the performance of the flow scheduler (§6.3).

6.1 Trace-Driven Evaluation on Bandwidth Cost Savings

Testbed setup. We set up a customized traffic scheduling simulation testbed to evaluate the bandwidth cost of Oceanus. The testbed simulates operations of the real-world system by replaying historical traffic demand and corresponding scheduling decisions. This allows us to precisely examine the bandwidth costs for edge nodes and the SLA compliance for individual traffic flows, based on the provided traffic demand and node state traces.

Specifically, the scheduler computes a bandwidth budget for each online node and determines the scheduling mapping for each flow in each 5-minute epoch, utilizing the historical traffic demand and performance data collected up to that time point. According to the generated mapping, the testbed calculates the bandwidth utilization of each node (by summing the demands of all assigned flows) and the performance of each flow. This resultant utilization and performance data is then fed as input to the next scheduling epoch, effectively closing the simulation loop. The billing cycle is set to 30 days, with node bandwidth utilization recorded every 5 minutes, resulting in a total of 8,640 time slots for each billing cycle. Please refer to Appendix C for more details.

Traces. The trace data was collected from a leading edge CDN vendor in China over a three-month period (June–August 2024), incorporating a high-demand-volatility event: the Paris 2024 Olympics. This dataset captures over Tbps of traffic demand across 2,300 nodes located in 30 regions (i.e., provinces). The real traffic transmission data was directly collected from edge nodes in the production environment, and subsequently aggregated at the ⟨domain name, user region⟩ level on a 5-minute timescale. After the flow management, over 10,000 flows are considered in each time slot, representing over 95% of the total system's traffic demand. These flows exhibit Gbps-level dynamic traffic demands and have specific performance assurance priorities, limiting the candidate-assigned nodes for scheduling. Appendix C provides the detailed trace collection method.

A real-time measurement system [5, 19] monitored network performance (using mean RTT as the key metric) from any region and Autonomous System (AS) to our nodes. The three-month measurement dataset is crucial for evaluating potential SLA violations resulting from any scheduling decision. We also recorded the timestamps of all node failures and newly constructed node online events. All nodes are billed by the 95th percentile bandwidth in this testbed.

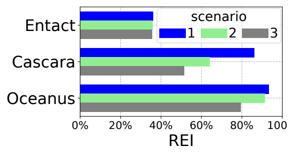
Scenarios. We designed three scenarios to simulate increasing levels of uncertainty, each posing progressively greater challenges for online cost reduction:

- *Scenario 1:* Only the bandwidth planning is performed, and the actual real-time traffic demand is given to the scheduler. Node utilization is directly set to the node's allocated bandwidth budget, assuming that all bandwidth budgets can be accurately implemented. The primary uncertainty considered is the long-term global dynamics of future traffic.
- *Scenario 2:* Schedulers are required to generate scheduling mappings based on the actual real-time traffic demand. Node utilization is computed by summing the actual traffic demands of all assigned flows. Schedulers may actively violate the bandwidth budgets of certain nodes to guarantee SLAs. This scenario additionally introduces the SLA-constrained scheduling bias.
- *Scenario 3:* Schedulers make decisions based on the last round of observed traffic demand, while the actual node utilization is recorded in real time according to gradually fluctuating traffic demands. This scenario closely resembles real-world conditions, introducing the uncertainty of systemic scheduling deviation.

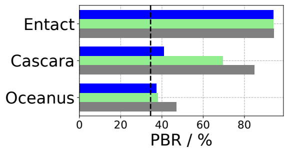
Baselines. To evaluate the cost savings achieved by Oceanus, we implemented two baseline methods in our testbed environment: Entact [62] and Cascara [45]. We also consider naive load balancing algorithm, which sets a node's bandwidth budget to the ratio between its capacity and the total capacity multiplied by the total traffic demand. This method does not save bandwidth costs. Since these methods cannot directly generate scheduling mappings in our testbed, we combined them with a global flow scheduler to produce scheduling mappings at each time slot based on the bandwidth budgets they output. The global flow scheduler, similar to [42, 56], reassigns all flows during each time slot heuristically at order of their priority. Additionally, we solved the offline optimization problem, assuming all traffic demands and uncertainties are known in advance, to determine the optimal cost savings that can theoretically be achieved.

Metrics. We adopt two metrics to quantify bandwidth cost savings, as adopted in work [7]:

- *Relative exploitation index* (*REI*): This normalized metric measures how effectively a method utilizes the cost-saving potential:
$$REI = \frac{Cost_{LB} - Cost_{achieved}}{Cost_{LB} - Cost_{optimal}} \tag{3}$$
where $Cost_{LB}$ is the cost achieved by naive load balancing, $Cost_{achieved}$ is the cost achieved by the evaluated method, and $Cost_{optimal}$ is the theoretical optimal cost achievable with full knowledge. A higher *REI* value close to 1 indicates minimal additional bandwidth cost loss.
- *Percentile billing ratio* (*PBR*): This metric represents the percentage value of the billed bandwidth relative to the total traffic demands:
$$PBR = \frac{|\{Traf_t | Traf_t \leq BW_{billed}\}|}{I} \tag{4}$$
where $Traf_t$ is the total traffic demands at time slot t, BW_{billed} is the total billed bandwidth, and I is the number of time slots in a billing cycle (i.e., 8,640). For naive load balancing, *PBR* is 95%, as the utilization curve of each node mirrors the overall traffic demand. A lower *PBR* indicates reduced overall billed bandwidth.

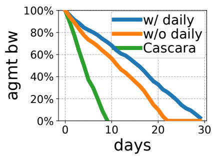


(a) REI of three methods.

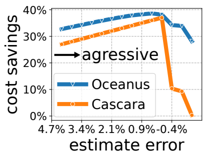


(b) PBR of three methods.

Fig. 10. Bandwidth cost savings achieved by Oceanus. In (b), the dashed line indicates the optimal *PBR* that can be achieved. Higher *REI* and lower *PBR* indicate more successful cost saving.



(a) Augmentation bw consumption rate.



(b) Impact of estimate error.

Fig. 11. Oceanus adapts to the traffic dynamics by regulating the consumption rate of augmentation bandwidth.

Overall Performance. As shown in Fig. 10, Oceanus achieves 79.4% of the total cost-saving potential (REI), directly reducing bandwidth costs by 21%. Under 95th-percentile billing, Oceanus's billed bandwidth (PBR) reaches the 47th percentile of traffic demands. In contrast, Cascara achieves only 51.5% of cost-saving potential with an 84.8th-percentile PBR, saving merely 8.1% in costs. These results confirm that Oceanus effectively mitigates the impact of uncertainties.

Performance under different scenarios. As shown in Fig. 10a and 10b, Oceanus adapts to increasing uncertainties across scenarios while limiting additional costs, achieving 93.3%, 91.2%, and 79.4% REI in Scenarios 1–3. Cascara performs well in pure bandwidth planning (86.1% REI, 40.9% PBR in Scenario 1) but deteriorates under high uncertainties (51.5% REI, 84.8% PBR in Scenario 3). Entact performs poorly throughout due to its simplified linear billing model.

6.2 Bandwidth Cost Reduction Deep Dive

In this part, controlled experiments are conducted to analyze how individual designs of Oceanus can mitigate the impact of uncertainties on overall bandwidth costs.

Daily traffic demand estimate accuracy. The daily predictive model is primarily designed to track dynamic traffic trends while effectively mitigating the influence of significant daily and weekly seasonality and high-frequency noise. Various time series prediction models, such as EWMA and Holt-Winters [16], can be employed to achieve this goal.

As illustrated in Fig. 12, we applied a seasonal EWMA with varying α settings to a three-month series of regional traffic demand, evaluating the Mean Absolute Percentage Error (MAPE) for each month and overall. Performance was also compared against the Holt-Winters model (represented by the horizontal dotted line in the corresponding color). The best-performing α setting is marked on each line. Setting α to 0.5 achieves the lowest estimate error (6.9%) over the entire period. However, we found no single setting consistently outperformed others over every month, likely due to the inherent unpredictability of long-term traffic dynamics and sudden, real-time traffic spikes. Furthermore, the Holt-Winters model achieved worse overall performance because it is required to predict 24×60/5=288 steps in a row, under which prediction errors accumulate easily. More sophisticated time series models, such as Prophet [47, 49], might overcome this limitation.

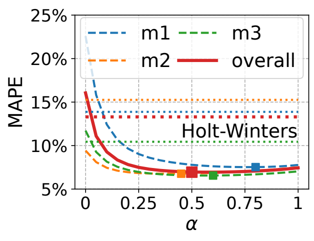
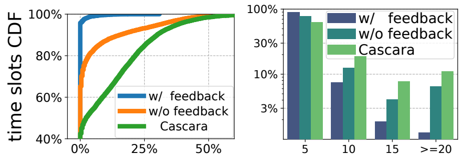


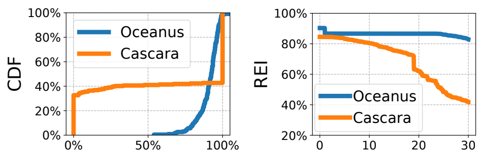
Fig. 12. Estimation MAPE for different predictive models and parameters.

Daily target billable bandwidth update. Oceanus daily updates target billable bandwidth to regulate augmentation consumption under long-term traffic demand dynamics (Fig. 11a). Careful recomputation ensures even consumption of augmentation resources, depleting slots precisely at month-end. Skipping daily updates risks overly aggressive target billable bandwidth, causing premature depletion of augmentation slots. In contrast, Cascara struggles to adapt to the dynamics due to its limited ability to preemptively adjust target billable bandwidth.



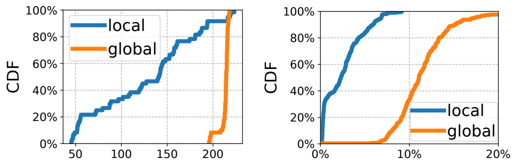
(a) Over-utilized nodes ratio. (b) Over-utilization duration.

Fig. 13. Oceanus limits occurrences and durations of node over-utilization.



(a) CDF of nodes' active augmentation times. (b) Cost-saving degradation within a month.

Fig. 14. Oceanus reduces the bandwidth cost increase caused by node over-utilization.



(a) Time expenditure (s). (b) Mapping change ratio

Fig. 15. CDF of the time expenditure for update and the change ratio of scheduling mappings.

	local	global
over-utilized nodes	4.0	5.7
RTT / ms	15.3	16.1
mapping change ratio	2.5%	11.5%
computation time / s	127.3	213.1

Table 1. Scheduling Performance.

Oceanus's in-cycle adjustability makes it insensitive to initial target billable bandwidth inaccuracies. As shown in Fig. 11b, Oceanus experiences less cost-saving degradation than Cascara under initial traffic estimate errors (defined at §2.3). Another evidence is shown in Fig.14b, where after detecting the initial estimate error, Oceanus adjusts the target billable bandwidth on the second day of the month, preventing premature depletion of augmentation slots.

Coordinated traffic scheduling through feedback. Oceanus employs bidirectional feedback to address real-time regional bandwidth provision imbalances. Fig. 13a shows the CDF of over-utilized node ratios over the billing cycle. With feedback, Oceanus prevents over-utilization in >95% of time slots (vs. 66% without feedback). Cascara's SLA-constrained infeasible bandwidth budgets cause widespread over-utilization during >50% of the billing cycle.

This improvement is also achieved by rapidly resolving over-utilization events. Fig. 13b illustrates the ratio of over-utilization event persistent time in all over-utilization events within a billing cycle, where the y-axis grows exponentially. Note that the fourth group of bars represents all cases exceeding 20 minutes, which accounts for its potentially increased magnitude. Feedback resolves 89.4% of over-utilization events within 5 minutes, significantly reducing free slot over-consumption.

Minimal marginal cost based augmented node selection. Oceanus mitigates inevitable node utilization discrepancies by consuming augmentation slots while reducing active augmentations based on the severity of discrepancies. Fig. 14a illustrates the CDF of nodes' active augmentation times after the entire billing cycle. In response to node over-utilization, it dynamically adjusts active augmentation frequency to balance consumption. Cascara continuously augments the same nodes, leaving insufficient buffer slots to handle node utilization discrepancies effectively.

When free slots are exhausted, Oceanus prioritizes augmenting nodes with minimal marginal cost (§4.3) to limit additional bandwidth cost (Fig. 14b). Conversely, Cascara's imbalanced node augmentation strategy wastes half its cost-reduction efforts.

6.3 Large-Scale Evaluation on Flow Scheduling

Real-world deployment. Oceanus's flow scheduler has been deployed in production and continues to evolve. Prior to the deployment of Oceanus's flow scheduler, a global flow scheduler introduced in §6.1 was applied. We finally compare the performance of scheduling mappings generated by Oceanus's *local* flow scheduler and the *global* flow scheduler in our production system.

Performance. The key metrics, averaged over the entire billing cycle, are summarized in Tab. 1. Although Oceanus's local scheduler adjusts the scheduling mapping in a localized manner, its scheduling performance matches global search. It reduces actively over-utilized nodes from 5.7 to 4.0 and assigns requests to nodes with 0.8 ms lower average RTT, compared to the global scheduler.

Oceanus reduces changed mapping entries by 78.3% and updates 40.3% faster due to its local adjustments. Conversely, the global scheduler unnecessarily reassigns all flows each time slot, causing: *(i)* High computation time: at least 180 seconds (Fig. 15a). High computation time hinders the system's ability to react promptly to abrupt network failures. *(ii)* Scheduling mapping instability: adjusting more than 6% mappings per update (Fig. 15b). Frequent scheduling mapping changes undermine cache hit rates of edge nodes, leading to additional bandwidth cost for L2 nodes and higher request latency for users.

7 Discussion

Traffic demand prediction. For daily forecasting, Oceanus tracks long-term regional traffic demand trends throughout the billing cycle to guide adjustments to the target billable bandwidth. Consequently, the system does not require highly precise prediction of every irregular traffic fluctuation, and the seasonal EWMA method achieves acceptable accuracy for this purpose (§6.2). Predictable event-induced traffic bursts (e.g., sports live-streaming, new TV series releases, software updates) can be captured in advance and accounted for by the monthly and daily traffic predictive models. Conversely, unanticipated traffic surges (e.g., breaking news or sudden Content-Provider (CP) initiated multihoming shifts [2, 29, 36]) may deplete free augmentation resources prematurely, but the resulting deficits are factored into the following day's daily plan. If necessary, an occasional LP re-organization is also acceptable.

Nevertheless, more sophisticated time-series prediction methods, such as those utilizing machine learning (ML) [23, 39, 49], can exploit additional cost-saving potentials in the online scenario. For daily planning, traffic demand changing trends can be captured more rapidly, enabling precise billable bandwidth adjustments. For real-time augmentation, accurate short-term traffic predictions could effectively mitigate the impact of systemic deviation and optimize node augmentation selection and stability. We leave more effective predictors for future work.

Generalization Although Oceanus is introduced under the 95th-percentile billing scheme for CDN scheduling, it is more complicated in practice and can be generalized to broader contexts.

- *Nodes with mixed billing schemes.* The real-world system consists of nodes with diverse billing schemes, as introduced in §2.2. Oceanus's objective functions in monthly MILP (§4.1) and daily LP (§4.2) can be generalized to other billing schemes and multiple billing cycle durations [64], which does not introduce additional complexity, as alternative billing schemes are linear. The computation of marginal cost during real-time node augmentation (§4.3) also supports various billing schemes. The integration of nodes with diverse billing schemes provides two advantages to the system: *(i)* the overall bandwidth costs can be further reduced; *(ii)* a buffer is provided to percentile-billed nodes when free slots are exhausted. In Appendix B, we provide an example analysis on how to further reduce cost by average-volume billed nodes.
- *Complex flow scheduling constraints.* Oceanus employs a heuristic flow scheduling approach (§5), enabling the incorporation of customized constraints. For example, Oceanus can restrict flow reassignment to a limited candidate node pool based on SLA assurance, service compatibility, business logic, multi-tenant considerations, or other criteria.
- *Broader traffic engineering (TE) context.* Oceanus's idea can also be applied to other contexts, such as inter-datacenter scheduling [45, 52, 64], multi-CDN scheduling [29], to achieve cost reduction as expected under widely existing uncertainties.

Feedback-based TE in the real world. Oceanus's design provides a better trade-off for interpretability, safety, and adaptability compared to ML-based methods. First, Oceanus's decisions are transparent and debuggable, a critical requirement for online operations, unlike the 'black box' nature of many ML models. Second, the bidirectional feedback loop forms a robust negative feedback structure, which inherently promotes stability and quick convergence, supported by Fig.13b. Conversely, ML systems typically provide no formal guarantees against SLA violations or node over-utilization. Third, Oceanus can immediately adapt to changes in the network (e.g., newly constructed nodes) by updating its model inputs, which remains a challenge for ML systems.

8 Related Work

Cost reduction. A lot of works employ traffic engineering to perform cost reduction in various scenarios, such as multihoming [14], CDN multihoming [29], inter-datacenter transferring [25, 64], cloud-edge scheduling [45, 52, 65], and CDN traffic assignment [7, 58]. While the linear billing scheme has been well-explored in [3, 25, 34, 37, 62], most works [17, 26, 43] focus on the non-linear percentile billing scheme, which has been proven to be NP-hard [14]. Among them, some works [22, 27, 57] manage to shift loads to different time slots, which conflicts with the real-time requirements of CDN services. To solve the online problem, Jetway [12] adopts a maximum flow algorithm, TARDIS [9] uses the Shapley [46] value, Cascara [45] and EdgeCross [52] formalize an MILP model and heuristically solve it, onTPC [7] and Iris [58] employ machine learning methods. However, most of the works fall short of handling uncertainties in practical systems, and thus cannot achieve the desired effect in the real world.

Traffic scheduling. To optimize transmission performance, cloud providers schedule traffic either to specific nodes [8, 13, 38, 44, 56] or through specific paths [5, 6, 21, 41], based on the real-time measurement data. The methods they adopt also vary. For example, Donar [53] and [20, 54, 63] propose distributed algorithms, Akamai's work[31, 38] adopts the stable marriage algorithm, and Espresso[56] simply uses a global greedy allocation algorithm. More works investigate multi-objective coordinated scheduling, such as node and path coordinated selection [18, 50], jointly service placement and request scheduling in edge computing [11, 30, 32], as well as load and performance coordinated optimizing in PCDN resource management [51, 60]. Entact [62] jointly optimizes performance and cost, which is similar with our work, but simplifies the cost model to reduce the computational complexity.

9 Conclusion

Large-scale edge CDNs suffer from high bandwidth costs. However, even state-of-the-art solutions still face large performance gaps between optimal, expected, and actual cost savings due to overlooking uncertainties in edge CDN systems. This paper presents Oceanus, a coordinated traffic scheduling system designed for large-scale edge CDN systems. Oceanus is designed to address the above uncertainties in both proactive and reactive manners. Extensive trace-driven and real-world experiments demonstrate that Oceanus can significantly reduce bandwidth costs over baselines with no performance degradation.

Acknowledgments

We sincerely thank our shepherd and all anonymous reviewers for their constructive feedback. This work was supported in part by the National Key R&D Program of China (2022YFB2901800).

References

[1] [n. d.]. THE GLOBAL INTERNET PHENOMENA REPORT MARCH 2024. https://www.applogicnetworks.com/phenomena.

[2] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. 2012. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *2012 Proceedings IEEE Infocom*. IEEE, 1620–1628.

[3] 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.

[4] Protick Bhowmick, Md. Ishtiaq Ashiq, Casey Deccio, and Taejoong Chung. 2023. TTL Violation of DNS Resolvers in the Wild. In *Passive and Active Measurement: 24th International Conference, PAM 2023, Virtual Event, March 21–23, 2023, Proceedings*. 550–563. doi:10.1007/978-3-031-28486-1_23

[5] 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.

[6] Fangfei Chen, Ramesh K Sitaraman, and Marcelo Torres. 2015. End-user mapping: Next generation request routing for content delivery. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 167–181.

[7] 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.

[8] David Chou, Tianyin Xu, Kaushik Veeraraghavan, Andrew Newell, Sonia Margulis, Lin Xiao, Pol Mauri Ruiz, Justin Meza, Kiryong Ha, Shruti Padmanabha, et al. 2019. Taiji: managing global user traffic for large-scale internet services at the edge. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 430–446.

[9] Richard G Clegg, Raul Landa, João Taveira Araújo, Eleni Mykoniati, David Griffin, and Miguel Rio. 2014. Tardis: Stably shifting traffic in space and time. *ACM SIGMETRICS Performance Evaluation Review* 42, 1 (2014), 593–594.

[10] IBM ILOG Cplex. 2009. V12. 1: User's Manual for CPLEX. *International Business Machines Corporation* 46, 53 (2009), 157.

[11] Vajiheh Farhadi, Fidan Mehmeti, Ting He, Thomas F La Porta, Hana Khamfroush, Shiqiang Wang, Kevin S Chan, and Konstantinos Poularakis. 2021. Service placement and request scheduling for data-intensive applications in edge clouds. *IEEE/ACM Transactions on Networking* 29, 2 (2021), 779–792.

[12] Yuan Feng, Baochun Li, and Bo Li. 2012. Jetway: Minimizing costs on inter-datacenter video traffic. In *Proceedings of the 20th ACM international conference on Multimedia*. 259–268.

[13] 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.

[14] David K Goldenberg, Lili Qiuy, Haiyong Xie, Yang Richard Yang, and Yin Zhang. 2004. Optimizing cost and performance for multihoming. *ACM SIGCOMM Computer Communication Review* 34, 4 (2004), 79–92.

[15] Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual. https://www.gurobi.com.

[16] {Robin John} Hyndman and George Athanasopoulos. 2018. *Forecasting: Principles and Practice* (2nd ed.). OTexts, Australia.

[17] 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.

[18] Wenjie Jiang, Rui Zhang-Shen, Jennifer Rexford, and Mung Chiang. 2009. Cooperative content distribution and traffic engineering in an ISP network. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. 239–250.

[19] Yuchen Jin, Sundararajan Renganathan, Ganesh Ananthanarayanan, Junchen Jiang, Venkata N Padmanabhan, Manuel Schroder, Matt Calder, and Arvind Krishnamurthy. 2019. Zooming in on wide-area latencies to a global cloud provider. In *Proceedings of the ACM Special Interest Group on Data Communication*. 104–116.

[20] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* 49, 3 (1998), 237–252.

[21] Raul Landa, Lorenzo Saino, Lennert Buytenhek, and João Taveira Araújo. 2021. Staying alive: Connection path reselection at the edge. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 233–251.

[22] Nikolaos Laoutaris, Georgios Smaragdakis, Pablo Rodriguez, and Ravi Sundaram. 2009. Delay tolerant bulk data transfers on the internet. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. 229–238.

[23] Fuyou Li, Zitian Zhang, Yunpeng Zhu, and Jie Zhang. 2020. Prediction of twitter traffic based on machine learning and data analytics. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*.

IEEE, 443–448.

[24] Jinyang Li, Zhenyu Li, Ri Lu, Kai Xiao, Songlin Li, Jufeng Chen, Jingyu Yang, Chunli Zong, Aiyun Chen, Qinghua Wu, et al. 2022. Livenet: a low-latency video transport network for large-scale live streaming. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 812–825.

[25] 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.

[26] Wenxin Li, Xiaobo Zhou, Keqiu Li, Heng Qi, and Deke Guo. 2018. TrafficShaper: Shaping inter-datacenter traffic to reduce the transmission cost. *IEEE/ACM Transactions on Networking* 26, 3 (2018), 1193–1206.

[27] Yuhua Lin, Haiying Shen, and Liuhua Chen. 2015. Ecoflow: An economical and deadline-driven inter-datacenter video flow scheduling system. In *Proceedings of the 23rd ACM international conference on Multimedia*. 1059–1062.

[28] Hongqiang Harry Liu, Raajay Viswanathan, Matt Calder, Aditya Akella, Ratul Mahajan, Jitendra Padhye, and Ming Zhang. 2016. Efficiently delivering online services over integrated infrastructure. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. 77–90.

[29] Hongqiang Harry Liu, Ye Wang, Yang Richard Yang, Hao Wang, and Chen Tian. 2012. Optimizing cost and performance for content multihoming. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. 371–382.

[30] Xiao Ma, Ao Zhou, Shan Zhang, and Shangguang Wang. 2020. Cooperative service caching and workload scheduling in mobile edge computing. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2076–2085.

[31] Bruce M Maggs and Ramesh K Sitaraman. 2015. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review* 45, 3 (2015), 52–66.

[32] 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.

[33] Zili Meng, Yaning Guo, Chen Sun, Bo Wang, Justine Sherry, Hongqiang Harry Liu, and Mingwei Xu. 2022. Achieving consistent low latency for wireless real-time communications with the shortest control loop. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 193–206.

[34] 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.

[35] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker. 2019. Cache Me If You Can: Effects of DNS Time-to-Live *(IMC '19)*. 101–115. doi:10.1145/3355369.3355568

[36] Matthew K Mukerjee, Ilker Nadi Bozkurt, Devdeep Ray, Bruce M Maggs, Srinivasan Seshan, and Hui Zhang. 2017. Redesigning cdn-broker interactions for improved content delivery. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. 68–80.

[37] 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. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 311–324.

[38] 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.

[39] Nipun Ramakrishnan and Tarun Soni. 2018. Network traffic prediction using recurrent neural networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 187–193.

[40] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 953–971.

[41] Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. 2019. Internet performance from facebook's edge. In *Proceedings of the Internet Measurement Conference*. 179–194.

[42] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. 2017. Engineering egress with edge fabric: Steering oceans of content to the world. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 418–431.

[43] Haiying Shen and Chenxi Qiu. 2018. Scheduling inter-datacenter video flows for cost efficiency. *IEEE Transactions on Services Computing* 14, 3 (2018), 834–849.

[44] Patrick Shuff. 2016. Building a billion user load balancer. (2016).

[45] 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.

[46] 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.

[47] Sean J Taylor and Benjamin Letham. 2018. Forecasting at scale. *The American Statistician* 72, 1 (2018), 37–45.

[48] 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.

[49] Oskar Triebe, Hansika Hewamalage, Polina Pilyugina, Nikolay Laptev, Christoph Bergmeir, and Ram Rajagopal. 2021. NeuralProphet: Explainable Forecasting at Scale. arXiv:2111.15397 [cs.LG] https://arxiv.org/abs/2111.15397

[50] Vytautas Valancius, Bharath Ravi, Nick Feamster, and Alex C Snoeren. 2013. Quantifying the benefits of joint content and network routing. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*. 243–254.

[51] 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.

[52] 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.

[53] Patrick Wendell, Joe Wenjie Jiang, Michael J Freedman, and Jennifer Rexford. 2010. Donar: decentralized server selection for cloud services. In *Proceedings of the ACM SIGCOMM 2010 conference*. 231–242.

[54] Hong Xu and Baochun Li. 2013. Joint request mapping and response routing for geo-distributed cloud services. In *2013 Proceedings IEEE INFOCOM*. IEEE, 854–862.

[55] Mengwei Xu, Zhe Fu, Xiao Ma, Li Zhang, Yanan Li, Feng Qian, Shangguang Wang, Ke Li, Jingyu Yang, and Xuanzhe Liu. 2021. From cloud to edge: a first look at public edge platforms. In *Proceedings of the 21st ACM Internet Measurement Conference*. 37–53.

[56] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. 2017. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 432–445.

[57] Ziwen Ye, Qing Li, Chunyu Qiao, Xiaoteng Ma, Yong Jiang, Qian Ma, Shengbin Meng, Zhenhui Yuan, and Zili Meng. 2024. KEPC-Push: A Knowledge-Enhanced Proactive Content Push Strategy for Edge-Assisted Video Feed Streaming. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 321–338. https://www.usenix.org/conference/atc24/presentation/ye-ziwen

[58] Huiyou Zhan, Haisheng Tan, Huang Xu, Chi Zhang, Hongqiu Ni, Pengfei Zhang, Weihua Shan, and Xiang-Yang Li. 2023. Online Midgress-Sensitive Traffic Allocation for Percentile Charging in Pracitcal CDNs. In *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[59] Huanhuan Zhang, Congkai An, Anfu Zhou, Yifan Zhu, Weilin Sun, Yixuan Lu, Jiahao Chen, Liang Liu, Huadong Ma, and Aiguo Fei. 2024. Venus: Enhancing QoE of Crowdsourced Live Video Streaming by Exploiting Multiflow Viewer Assistance. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 170–184.

[60] 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.

[61] 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.

[62] 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.

[63] Rui Zhang-Shen and Nick McKeown. 2004. Designing a predictable Internet backbone network. HotNets.

[64] Gongming Zhao, Jingzhou Wang, Hongli Xu, Zhuolong Yu, and Chunming Qiao. 2023. COIN: Cost-Efficient Traffic Engineering with Various Pricing Schemes in Clouds. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[65] Pengxiang Zhao, Jintao You, and Xiaoming Yuan. 2024. Circling Reduction Algorithm for Cloud Edge Traffic Allocation Under the 95th Percentile Billing. *IEEE/ACM Transactions on Networking* (2024).

Appendix

A Detailed Algorithms for Bandwidth Planner and Flow Scheduler

We show the detailed algorithms adopted by the bandwidth planner and the flow scheduler.

At the start of each billing cycle, Oceanus first initializes the target billable bandwidth for each edge node by solving an MILP optimization problem (§4.1). The optimization problem is formalized as follows:

$$\begin{aligned}
\min_{L} \quad & \sum_{n \in N} p_n \cdot L_n && (5)\\
\text{s.t.} \quad & B_n^t \leq C_n, \forall t \in T, n \in N; && (6)\\
& B_n^t - \lambda_n^t * M <= L_n, \forall t \in T, n \in N; && (7)\\
& \sum_t \lambda_n^t \leq \lfloor \frac{|T|}{20} \rfloor, \forall n \in N; && (8)\\
& \sum_i X_{ij}^t \leq \sum_{n \in N_j} B_n^t, \forall j \in R, t \in T; && (9)\\
& \sum_j X_{ij}^t \geq D_i^t, \forall i \in R, t \in T; && (10)\\
& X_{ij}^t \leq 0, \forall t \in T, j \notin CRP_i. && (11)
\end{aligned}$$

where $\lambda_n^t \in \{0, 1\}$ represents the augmentation decision for node n in time slot t and $\lfloor \cdots \rfloor$ indicates rounding down. M is a large constant integer (e.g., infinitely positive) [7, 45, 58, 64], ensuring that B_n^t can be larger than L_n if node n is augmented (i.e., $\lambda_n^t = 1$). Eq. 6, Eqs. 7-8, Eqs. 9-10, and Eq. 11 correspond to the four types of constraints, respectively.

At the start of each day, the bandwidth planner updates the target billable bandwidth for each node (§4.2). Specifically, Oceanus adopts a multi-step process to solve the problem, which is outlined in Alg. 1.

Based on the latest target billable bandwidth, the bandwidth planner selects nodes to augment to provide sufficient bandwidth to cover traffic demands every 5 minutes (§4.3). Oceanus first selects nodes with minimal marginal costs, and then globally adjusts the decision to balance bandwidth provision across regions. The algorithm is outlined in Alg. 2.

The flow assigner adjusts the scheduling mapping also every 5 minutes, based on the latest bandwidth budget at each node (§5). The assignment algorithm is outlined in Alg. 3.

B Mixed Billing Schemes' Potential

In this section, we first discuss the benefits provided by including average-billed nodes in the system. Next, the method for computing bandwidth budgets for average-billed nodes is explained. It is noteworthy that the fundamental principle remains applicable to other linear billable bandwidth computation schemes and billing functions.

Overall bandwidth costs reduction. The first advantage lies in the further bandwidth cost reduction. By offloading traffic demands from percentile-billed nodes to average-billed nodes, the utilization curve of the percentile-billed nodes can be flattened. This allows the 95th-percentile billable bandwidth of a percentile-billed node to be reduced to the $(1 - 1/p)$th-percentile billable bandwidth, where p represents the relative price of the average-billed node.

For ease of explanation, consider a scenario with a single 95th-percentile billed node, n_{95}, and a single average billed node, n_{avg}. Each of these nodes has an unlimited capacity. The unit price for the average billed node n_{avg} is p times higher than for the percentile billed node n_{95}. During a billing cycle $T = \{t_1, \cdots, t_I\}$ with I time slots, the traffic demand series is given by $D = \{D_t\}$.

Algorithm 1: Daily Target Billable Bandwidth Update

```
Input: Previous day's regional traffic demands: $D_i$; history bandwidth budgets $B_n^t$; latest target billable bandwidths $L'_n$; remaining days $d$.
Output: New target billable bandwidths $L_n$.
1  $\hat{D}_i \leftarrow$ UpdateTrafficModel($D_i$,TodayIsWeekend);
2  $AgmtResources \leftarrow 0$;
3  for $n \in N$ do
4      $Slots \leftarrow \sum_t \mathbb{1}_{\{B_n^t > L'_n\}}$;
5      $AgmtResourecs+ = (MaxSlots - Slots) \cdot (C_n - L'_n)$;
6  end
7  $DailyQuota \leftarrow AgmtResources/d$;
8  $q \leftarrow$ BinarySearch($\{\hat{D}_i\}$, DailyQuota);
9  foreach region $i \in R$ do $NormalTraf_i \leftarrow$ Percentile($\hat{D}_i, q$);
10 Solve LPs to find new $L_n$:
```

$$\min_{L} \quad \sum_{n \in N} p_n \cdot L_n \tag{12}$$

$$\text{s.t.} \quad L_n \leq C_n, \forall n \in N; \tag{13}$$

$$\sum_i X_{ij} \leq \sum_{i \in N_j} L_n, \forall j \in R; \tag{14}$$

$$\sum_j X_{ij} \geq NormalTraf_i, \forall i \in R; \tag{15}$$

$$X_{ij} \leq 0, \forall j \notin CRP_i. \tag{16}$$

At the start of the allocation, we leave all traffic demands on node n_{95}. The utilization u_t of node n_{95} equals D_t, i.e., $u_t = D_t$. The initial billable bandwidth can be computed as $P_{95}(D)$, where $P_q(\cdot)$ denotes the q-percentile largest value for a sequence. Now we migrate a specific amount of traffic demand δ_t at time slot t to node n_{avg}. Let the remaining cost of node n_{95} be $x = P_{95}(\{u'_t\})$, where u'_t denotes the remaining load at time t. The migrated traffic demand δ_t is set to:

$$\delta_t = \max(u_t - x, 0) \tag{17}$$

At this time, the total bandwidth cost can be represented as a function of x:

$$cost = P_{95}(\{u'_t\}) + p \cdot \frac{1}{T}\sum_t^T \delta_t = x + p \cdot \frac{1}{T}\sum_t^T \max(u_t - x, 0) \tag{18}$$

To find the minimum overall bandwidth cost, let

$$\frac{\mathrm{d}cost}{\mathrm{d}x} = 1 - p \cdot \frac{1}{T}\sum_t^T \mathbb{1}_{\{u_t \geq x\}} = 0 \tag{19}$$

where $\mathbb{1}_{\{u_t \geq x\}}$ is the indicator function. Finally, we get:

$$\sum_t^T \mathbb{1}_{\{u_t \geq x\}} = \frac{1}{p} \cdot T \tag{20}$$

The Eq. 20 shows that we can continuously migrate the excess traffic away from the 95-percentile billed node until the remaining 95th-percentile largest utilization reaches the value x for $1/p$ of all times. This migration essentially flattens the utilization curve on node n_{95} to meet $P_{95}(\{u'_t\}) = P_{1-1/p}(\{u'_t\})$. As a trade-off, the migrated traffic demands will be billed by the average volume at a higher unit price. A steeper traffic demand curve will reduce the amount of this part of the traffic demand, thus saving more bandwidth costs.

Buffer for percentile billed nodes. The average-billed nodes also act as a buffer when the free slots of percentile-billed nodes are exhausted. Although the marginal cost of average-billed

Algorithm 2: Real-Time Node Augmentation

```
Input: Real time traffic demands $D_i^t$; target billable bandwidth $L_n$; history utilization rate of nodes $U_n^t$; report $F_i$.
Output: Actual bandwidth budgets $B_n^t$; suggestion $H_{ij}^t$.
$GivenResources \leftarrow \sum_{n \in N} L_n$;
if $\sum_i D_i^t \leq GivenResources$ then
    foreach $n \in N$ do $B_n^t \leftarrow L_n$;
    return;
end
foreach $n \in N$ do $MarginalCost_n \leftarrow$ ComputeMarginalCost($n$);
foreach $n \in N$ do $AgmtSlots_n \leftarrow$ ComputeAgmtSlots($n$);
$Pool \leftarrow$ Sort($MarginalCost, AgmtSlots$);
$A \leftarrow \emptyset$;
while $\sum_i D_i^t \leq GivenResources$ do                                  // augment
    $n \leftarrow Pool.pop()$;
    $A \leftarrow A + \{n\}$; $GivenResources += C_n - L_n$
end
foreach $i$ in $F_i \geq 0$ do                                               // feedback
    while $F_i \geq 0$ do
        $n$ at region $j \leftarrow Pool.pop(CRP_i)$;
        $A \leftarrow A + \{n\}$; $F_i -= C_n - L_n$; $H_{ij}^t += C_n - L_n$;
    end
end
foreach $n \in A$ at region $i$ do                                           // recycle
    for region $j \in R$ where $F_j + (C_n - L_n) \leq 0$ and $i \in CRP_j$ do
        $A \leftarrow A - n$; $F_j += C_n - L_n$; $H_{ij}^t += C_n - L_n$;
    end
end
foreach $n \in N$ do
    if $n \in A$ then $B_n^t \leftarrow C_n$;
    else $B_n^t \leftarrow L_n$;
end
```

nodes is slightly higher than zero, it is still extremely low. Oceanus prioritizes the utilization of average-billed nodes when unanticipated high traffic demand occurs late in the billing cycle, while free slots of the percentile billing nodes are exhausted. This approach prevents a significant increase in target billable bandwidth for percentile-billed nodes during the final days.

Bandwidth budgets computation. The bandwidth planner computes the entire day's bandwidth budgets of average billed nodes in daily optimization, where the objective function in (12) can be extended as follows:

$$\min_{L_n^t} \sum_{n \in N_{95}} p_n \cdot L_i + \sum_{n \in N_{avg}} p_n \cdot \frac{1}{T} \cdot \sum_{t}^{T} L_n^t + \rho \cdot \sum_{i \in N_{avg}}^{T} |L_n^t - L_n^{t-1}| \tag{21}$$

where the first item minimizes the bandwidth cost of percentile billed nodes (§4.2), the second item minimizes the bandwidth cost of average billed nodes, and the last item promises the stability of bandwidth budgets for average billed nodes. The penalty's weight ρ is set to 10^{-4}.

C Testbed Setup and Trace Collection Detail

We provide a detailed introduction to the trace collection method and the traffic replay testbed.

Traffic demand collection. We recorded the requested domain name, timestamp, user region source, node ID, and transmitted data size for every request at our edge nodes. Information associated

Algorithm 3: Flow Assignment

```
Input: Last scheduling mapping $M^{t-1}$; bandwidth budgets $B_n^t$; node utilization $U_n^t$; flow RTT $lat_s$; suggestion $H^t$.
Output: New scheduling mapping $M^t$; report $F^t$
$Q \leftarrow \emptyset$;
foreach $s \in S$ do                                                   // flow performance check
    if $lat_s \geq SLA_s$ then $Q \leftarrow Q + \{s\}$;
end
while not TimeOut do
    foreach $n \in N$ do                                               // node utilization check
        if $U_n^t \geq \eta B_n^t$ then
            $flows \leftarrow$ MarkFlows$(n, U_n^t - \eta B_n^t)$;
            $Q \leftarrow Q + flows$;
        end
    end
    if $Q$ is empty then break;
    foreach $s \in Q$ at region $i$ do                                 // reassign
        $NodePool_s \leftarrow \{n \in N_j | j \in CRP_i\}$;
        sort $NodePool_s$ by remaining bandwidth $\eta B_n^t - U_n^t$;
        $n \leftarrow NodePool_s[0]$;
        if $\eta B_n^t - U_n^t \geq size_s$ then assign $s$ to $n$; $Q \leftarrow Q - \{s\}$;
        if not assigned then
            sort $H_i^t$;
            foreach region $j \in H_i^t$ do
                $n, newflows \leftarrow$ TakeOverFromRegion (j);
                $Q \leftarrow Q + newflows$;
                assign $s$ to $n$; $Q \leftarrow Q - \{s\}$; break;
            end
        end
        if not assigned then assign $s$ to $NodePool_s[0]$;
    end
end
foreach $i \in R$ do $F_i^t \leftarrow \sum_{n \in i} U_n^t - B_n^t$;     // feedback
```

with individual users was not collected. This raw data was subsequently aggregated to compute the traffic demand for each ⟨domain name, user region⟩ pair on a 5-minute timescale.

Modeling Systemic Deviations. The actual traffic demand fluctuates between the moment when measurement data is collected and when the scheduling decision takes effect. We model this as a traffic demand error representing all forms of systemic uncertainty, as the new, precise demand is only observable after the next round of measurement data is collected.

Unmodeled Factors. We did not explicitly track public DNS settings. The uncertainty stemming from Internet settings (e.g., BGP announcement changes, non-standard DNS settings [4, 28, 35]) is treated as a component of the aforementioned systemic uncertainty. We also omit performance deterioration due to cache misses, as cache setup is generally quick and only affects a small group of users. Similarly, backward traffic from edge nodes to L2 nodes is omitted, as its scale is orders of magnitude smaller than the outbound traffic to users.

Received June 2025; accepted September 2025