Optimizing Timely Bulk Data Transfers With Hybrid Elastic Cloud Resources

Long Luo, *Member, IEEE*, Yunxiang Zhou, *Student Member, IEEE*, Linjian Yu, Jin Shen, *Student Member, IEEE*, Hongfang Yu, *Senior Member, IEEE*, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*—The persistent disparity between the slow growth of wide-area network bandwidth and the escalating demand for high-speed bulk data transfer poses a significant challenge for inter-datacenter data transmission. Existing approaches typically rely on optimized algorithms for dedicated networks and often struggle to meet the stringent deadline requirements of modern applications cost-effectively. To overcome these limitations, we propose a novel cloud-accelerated transfer approach that is intrinsically deadline-aware. We present a Profit-driven RelAxation-based TransfEr Scheduling (PRATER) system that leverages cloud proxies and multipath transmission to accelerate bulk data transfer. Specifically, we tackle the problem of profit maximization by jointly optimizing cloud proxy deployment and traffic allocation under transmission deadlines, a problem formally modeled and identified as NP-hard. To efficiently derive near-optimal solutions, we decompose this complex problem into two interconnected subproblems: cost minimization through proxy deployment optimization and revenue maximization via traffic allocation optimization. Our efficient iterative algorithm resolves these subproblems by minimizing cloud proxy and bandwidth costs while maximizing timely data transfer completions, thereby enhancing overall system profitability. Extensive experimental results demonstrate that our approach achieves a significant profit improvement of approximately $2.4\times$–$6.1\times$ compared to state-of-the-art existing algorithms.**

***Index Terms*—Bulk data transfers, cloud proxy, elastic cloud, wide-area network.**

I. Introduction

In RECENT decades, the number of service providers and cloud platform providers operating applications on a global scale has grown significantly. These applications span a wide range of domains, including financial services, metaverse/immersive streaming services, online gaming, machine learning model training services, and AI-generated content (AIGC) services, etc [1], [2], [3]. These services rely heavily on geographically dispersed datacenters and the timely transmission of large volumes of data to function effectively [4], [5], [6], [7]. For example, financial companies need high-speed data transfer to complete transactions and manage risks promtly, which is essential for maintaining their competitive edge. Similarly, metaverse platforms require fast content loading to provide users with an immersive and seamless user experience. Machine learning model training is another domain that relies heavily on timely data transfer. As the size of datasets and the complexity of machine learning and large language models continue to grow and evolve, transferring training data swiftly and efficiently to the location where the model is being trained becomes increasingly crucial. Delays in data transfer can significantly slow down the training process and impede the development of advanced AI models. The timeliness of data transfer not only enhances operational efficiency for enterprises but also plays a vital role in improving customer satisfaction. As a result, timely data transfer services, particularly those with strict deadlines, have become increasingly critical in today's digital landscape.

The growing demand for timely data transfer has driven the development of various optimization techniques, especially in the context of inter-datacenter wide area networks. These techniques can be broadly classified into two approaches: utilizing private dedicated infrastructure or leveraging bandwidth leasing to facilitate data transfers. Industry giants like Google and Microsoft [8], [9], [10] have invested significantly in building high-speed dedicated connections to overcome the limitations and congestion of the public Internet. These dedicated connections are high bandwidth, enabling the rapid transfer of large volumes of data between datacenters. However, constructing and maintaining such dedicated infrastructure can be costly and complex, making it challenging for smaller service or content providers or those with limited resources to implement. As an alternative, many service providers rely on leasing bandwidth from Internet Service Providers (ISPs) to facilitate data transfers for their global applications. In this model, transfer costs are typically charged based on bandwidth usage [11], *e.g.*, the peak bandwidth usage over extended periods of a day, a week, or a month [12], [13]. Consequently, this approach can lead to high costs during periods of fluctuating demand. Moreover, the overall WAN cost is often determined by the sum of peak bandwidths across the network, which can result in inefficient and inflated expenses.

To address these challenges, there has been a recent growing interest in serving bulk data transfers with elastic cloud resources for efficiency and cost-effectiveness [14], [15], [16], [17], [18]. For example, researchers have proposed Sky computing [14],

Received 23 December 2024; revised 25 June 2025; accepted 3 November 2025. Date of publication 6 November 2025; date of current version 10 March 2026. This work was supported by National Key Research and Development Program of China under Grant 2023YFB2904600. Recommended for acceptance by L. Gao. *(Corresponding author: Long Luo.)*

Long Luo, Yunxiang Zhou, Linjian Yu, Jin Shen, and Hongfang Yu are with the University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: llong@uestc.edu.cn; yulinjian222@gmail.com; 18268462890@163.com; yuhf@uestc.edu.cn).

Schahram Dustdar is with Distributed Systems Group, TU Wien, 1040 Vienna, Austria, and also with ICREA, Pompeu Fabra University (UPF), 08002 Barcelona, Spain (e-mail: dustdar@dsg.tuwien.ac.at).

Digital Object Identifier 10.1109/TCC.2025.3630149

[19], an emerging platform that enables providers to build their planetary-scale applications on an infrastructure consisting of multiple heterogeneous competing commercial cloud providers. Furthermore, the capacity of the overlay network can be dynamically scaled by adding or removing instances (*i.e.*, VMs or containers) in the cloud, providing significant flexibility. This flexibility allows providers to dynamically launch instances to deploy transmission proxies on demand to accelerate data transfers and elastically adjust cloud resource usage based on real-time demands, avoiding committing to fixed bandwidth contracts that may not always align with actual needs. Consequently, bulk data transfers can be optimized more flexibly based on actual timeliness requirements, substantially reducing operational costs. Moreover, the shift toward cloud-based solutions has led to innovative methods for enhancing data transfer throughput. Prior researches have demonstrated the effectiveness of employing cloud-based TCP-split proxies to improve the performance of individual data flows [20], [21], [22], [23]. By partitioning a single TCP flow into multiple connections with reduced round-trip times (RTTs), these proxies can significantly enhance overall transmission throughput [24]. However, while existing studies have primarily focused on optimizing single-flow performance, a critical challenge remains: the strategic deployment of a limited number of TCP-split proxies to support the timely transfer of substantial data volumes (i.e., before specific deadlines), thereby enhancing the performance of geo-distributed applications.

Accelerating bulk data transfers with proxies in the cloud is complicated, and only a few works have made preliminary exploration in this field. For example, Skyplane [16] and Cloudcast [18] spawn multiple cloud instances to create parallel connections to improve throughput for unicast and multicast transfers, respectively. However, these works primary focus on forwarding mechanism using cloud proxies without TCP splitting capability. In practice, forwarding through proxies without TCP-splitting offers little to no improvement for a single connection, as several studies have demonstrated [20], [25]. These approaches fail to fully exploit the potential benefits of cloud-based acceleration technique. Cloudpliot [24] leverages TCP-split connections and addresses the placement of cloud proxies across multiple cloud regions to reduce flow completion time. However, it adopts a single routing path model and does not distribute traffic from the same participants across multiple cloud paths. Consequently, as traffic data scales up, the transmission rate of a single routing path may fail to deliver the data timely, especially before the specific deadlines. Thus, the proxy acceleration scheme alone cannot fully satisfy transfer deadline requirements when dealing with substantial volumes of data.

To address this challenge, we propose integrating the multipath transmission model within the cloud acceleration scheme. Multipath transmission and cloud proxies are two complementary and mutually beneficial approaches for improving bulk data transfer performance in geo-distributed applications. By strategically selecting cloud locations to deploy transmission proxies, creating multiple routing paths, and appropriately splitting traffic across these paths, we can effectively enhance data transfer efficiency and strike a cost-performance balance, as demonstrated in Section V. Therefore, there is an urgent need for proxy acceleration solutions that incorporate proxy deployment and route selection. However, achieving this goal is non-trivial. Firstly, we must determine the optimal placement of proxies, which is inherently challenging because the routing paths of data transfer requests are uncertain before proxy deployment. However, the optimal proxy deployment also relies on efficiently optimizing the traffic along the routing paths. This creates a complex interdependency between proxy placement and route optimization. Secondly, the path traverses cloud proxies will alter forwarding throughput, meaning that routing bandwidth becomes variable. Thirdly, data transfers will face multi-dimensional resource constraints, such as cloud cost, egress capacity and deadline limitations. To address these challenges, this paper proposes PRATER, a novel solution for optimizing bulk data transfers in geo-distributed environments. The main contributions of our work are as follows:

- Novel Data Transfer Model: We propose a novel data transfer model that integrates TCP-splitting based cloud acceleration and multiple path routing to address the challenges of high throughput and cost optimization for timely bulk data transfers. Our approach combines the strengths of both techniques to create a synergistic solution that efficiently utilizes cloud resources, accelerates data transfer, and enables optimal traffic allocation. To the best of our knowledge, we are the first to systematically combine these techniques, taking a holistic view of the problem and proposing a comprehensive solution that addresses both throughput and cost aspects.
- Problem Formulation: We formulate the joint optimization problem of proxy placement and traffic allocation with the objective of profit maximization (PMPTA) as a mixed nonlinear programming problem. This formulation considers transfer deadlines, bandwidth limitations, cloud costs, and the complex interplay between proxy deployment, traffic allocation, and resource constraints. It provides a comprehensive framework for optimizing data transfer performance and cost-efficiency.
- Algorithm Design: We propose a problem decomposition approach to tackle the complex PMPTA problem. By breaking it down into two subproblems, we develop efficient algorithms to solve each subproblem separately. We then integrate these solutions through an iterative framework that alternately solves the subproblems to obtain a comprehensive solution for the original PMPTA problem. This approach allows us to balance the trade-off between revenue and cloud costs, leading to a more efficient and profitable solution.
- Extensive Evaluation: We evaluate PRATER under various network conditions, workload patterns, and cloud pricing models. The results show that PRATER significantly improves data transfer performance and cost-efficiency compared to existing solutions. Our results highlight the effectiveness of our proposed algorithms and the benefits of joint optimization in real-world scenarios, showcasing its practicality and potential.

II. Related Work and Motivation

A. Related Work

Over the past decade, researchers have made significant strides in optimizing bulk data transfers for geo-distributed applications, fueled by the pressing need to optimize cost and performance. The distinctive characteristics of bulk data transfers pose unique challenges that have catalyzed the development of innovative traffic engineering algorithms.

A substantial body of current research concentrates on optimizing the bandwidth utilization of private wide area networks (WANs) to significantly enhance the performance of bulk data transfers. Notably, numerous cloud providers have proposed various traffic engineering solutions, such Google's B4 [9], [10], as well as Microsoft's SWAN [8], Cascara [26], TEAL [27] and BlastShield [28], to manage bulk data transfers within their respective WAN infrastructures. These WANs typically comprise a combination of private links from large cloud providers and third-party-owned infrastructure. Due to its design to accommodate peak traffic, this infrastructure is often costly, and the total link bandwidth capacity between two nodes in such a traditional network overlay is considered fixed. Given the inherently fluctuating nature of network traffic, one of the key challenges is the mismatch between dynamic traffic demands and available network bandwidth. Consequently, many prior works generally concentrate on saturating the available bandwidth to improve resource utilization or reducing peak bandwidth usage to achieve cost efficiency. For example, BDS+ [29] focuses on inter-datacenter (DC) multicast, which involves replicating massive amounts of data (*e.g.*, user logs, web search indexes) from one DC to multiple DCs in geo-distributed locations. It explores the utilization of the remaining available bandwidth reserved for online high-priority traffic on existing overlay paths. More recently, Google proposes Effingo [30], a massively-parallel data transfer service that optimizes throughput while ensuring smooth bandwidth usage and cost efficiency. To handle complex and large-scale requests, Effingo employs a decentralized control mechanism to manage over an exabyte of data daily with high resiliency and minimized network costs. This contrasts with centralized systems like Amoeba [31] and OWAN [32], which focus on meeting transfer deadlines but struggle with scalability.

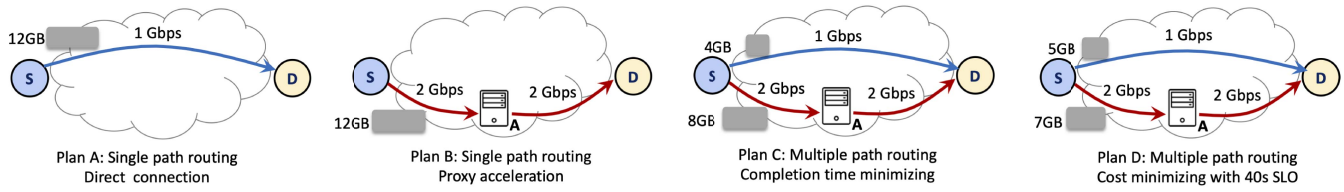
In contrast to these approaches, our work considers extending the overlay network to the cloud setting. Specifically, we leverage the elasticity of cloud resources to relay data transfers and scale network bandwidth by spawning cloud instances on demand. We also account for the cost of network bandwidth, addressing the unique challenges associated with dynamic resource availability and variable pricing models in cloud environments. This enables more flexible and cost-effective optimization of bulk data transfers across geographically distributed infrastructures.

Actually, Optimizing bulk data transfers with elastic cloud resources is not new. Numerous prior studies [16], [18], [24], [26], [28], [33], [34], [35], [36] have demonstrated the benefits of utilizing the cloud as an overlay to enhance the performance of global applications, particularly when a privately interconnected network is unavailable. For example, NetUber [33] explores the use of third-party cloud spot instances to construct large-scale overlay networks that address the variable bandwidth demands of inter-region data transfers. By adopting a bandwidth-based request model, NetUber dynamically adjusts resource allocation and provides users with a cost-effective connectivity solution. Similarly, J-QoS [34] proposes combining low-cost IP services with cloud overlays to ensure reliable and timely packet delivery for demanding applications. J-QoS achieves this objective by offering three services with different cost-performance trade-offs, allowing users to select the most suitable option based on their specific requirements. However, most of these studies [26], [28], [33], [34], [35], [36] focus primarily on bandwidth demand rather than on bulk transfers that prioritize the timely delivery of specific amounts of application data, as opposed to employing a one-size-fits-all bandwidth rate. Our work addresses this gap by considering the unique requirements of bulk data transfers and optimizing for the timely completion of specific data transfer tasks, rather than solely focusing on bandwidth provisioning.

Only a few recent studies have explored how to accelerate bulk transfers using elastic cloud resources, with a particular emphasis on balancing cloud costs and transfer throughput. Notably, Skyplane [16] leverages cloud elasticity and overlay networking to reduce transfer costs and enhance throughput for unicast bulk data replication. Furthermore, Cloudcast [18] extends Skyplane to support the acceleration of multicast transfers in large-scale data replication scenarios, particularly aiming to reduce cloud costs. The fundamental approach of these works involves employing parallel TCP connections to achieve higher throughput between two nodes, while the throughput of each individual connection remains limited.

In contrast to using direct TCP connections, prior studies [21], [22] propose the use of TCP splitting proxies to improve throughput, particularly in environments with long round-trip times (RTTs), such as mobile networks, web transfers over HTTP connections, and satellite connections with long distances. For instance, the OCD framework [20] divides traditional end-to-end TCP connections into three segments: source to cloud relay, inter-cloud relay, and cloud relay to the target. By leveraging elastic cloud resources, OCD dynamically adjusts the number and location of relay nodes, optimizing both the performance and reliability of data transmission. By splitting a single TCP flow into multiple connections with shorter RTTs, TCP-splitting proxies enhance overall throughput. However, most of these studies focus on isolated flows. CloudPilot [15] is noteworthy as it aligns with the focus of our work, improving system-wide performance for multiple flows in globally distributed applications. Unlike our work, which seeks to meet specific deadlines, CloudPilot primarily targets the reduction of flow completion times by strategically placing on-demand TCP proxies in the cloud while adhering to cost constraints.

Our work distinguishes itself from the aforementioned studies by focusing on meeting specific deadlines for bulk data transfers while considering the unique challenges posed by elastic cloud resources and variable pricing models. By optimizing the placement and utilization of cloud-based TCP-splitting proxies, we aim to achieve timely completion of bulk data transfers in a



Approach	w/ Proxy	Transmission strategies		Completion time (s)	Cost ($¢$)
		Multiple Path Routing	*Traffic Splitting*		
Plan A	no	no, only direct connection path	no	96	0
Plan B	yes	no, only cloud acceleration path	no	48	58
Plan C	yes	yes, direct connection path + cloud acceleration path	yes (Proportional to bandwidth)	32	42
Plan D	yes	yes, direct connection path + cloud acceleration path	yes (Deadline-aware)	40	38

Fig. 1. A toy example illustrating data transfer using different plans. The source and destination nodes are labeled 'S' and 'D', respectively, while cloud proxy node is denoted as 'A'. The number adjacent to each directed edge represents link bandwidth. Establishing a cloud proxy incurs a cost of 10¢, and there is an egress cost of 4¢per GB for data exiting the cloud proxy.

cost-effective manner, taking into account the system-wide performance of multiple flows in globally distributed applications.

B. Motivation

One of the primary advantages of cloud computing is resource elasticity, which allows for the flexible provisioning of cloud instances (*i.e.*, VMs or containers) across numerous regions. This elasticity directly translates into bandwidth scalability. By allocating cloud instances as relays, users can establish bypass paths that increase throughput beyond the network bandwidth limitations of direct overlay paths between the source and destination nodes. However, spawning cloud instances presents its own set of challenges. Adding elastic cloud instances incurs additional costs, necessitating a careful balance between cost and throughput.

An illustrated case study: Fig. 1 illustrates various scheduling methods for a toy data transfer demand, serving as a concrete example. This example demonstrates the inherent challenges involved in using cloud nodes to optimize both cost and throughput. Consider the scenario depicted in Fig. 1, where a 12 GB data is required to be replicated from source node S to destination node D. Suppose there is a direct path bandwidth limit of 1 Gbps between S and D. The simple direct replication strategy (Plan A) would take approximately 96 seconds. While full cloud acceleration strategy (Plan B) would spawn a Split-TCP proxy in the cloud and uses a cloud proxy path at 2 Gbps. This would result in a completion time of 48 seconds and incur a cost of 58 (10 setup + 48 egress). We can see that both Plan A and Plan B utilize a single-path transmission strategy. We could also explore a multipath strategy as Plan C, splitting the data and sending it simultaneously along both the direct path and the Split-TCP path. This approach might achieve a rate of 3 Gbps, reducing the completion time to 32 seconds at a cost of 42¢. In terms of minimizing completion time, Plan C is optimal. However, suppose the data replication has a service level objective (SLO) that requires completion before a deadline of 40 seconds, longer than the 32 seconds achieved by Plan C. In this case, we can achieve a better cost-performance trade-off. Plan D leverages uneven traffic splitting between the direct path and the overlay path, preferentially utilizing the direct path whenever possible to minimize reliance on cloud resources. By doing so, Plan D effectively reduces cloud costs while still satisfying the SLO.

Our intuition: This example illustrates that data transmission over multiple paths provides greater flexibility compared to single-path transmission. However, it also introduces an increased complexity due to the extensive search space of potential data transfer strategies. Furthermore, real-world cloud networks exacerbate this complexity with multiple regional cloud nodes, varying instance costs, and different egress bandwidth expenses [14], [16]. Designing a globally optimal strategy for multiple bulk transfer flows requires not only addressing diverse SLOs, such as meeting deadlines, but also accounting for the instance and bandwidth costs associated with cloud proxies. Therefore, balancing cost and performance is essential when developing data transfer plans in cloud-accelerated transmission scenarios. In this work, we propose optimizing the deployment of cloud proxies and traffic splitting for data transfer across multiple paths to maximize the total system revenue from serving data transfers completed before specific deadlines while minimizing cloud costs, thereby enhancing overall system profitability.

III. Optimizing Timely Data Transfers With Cloud Proxies: Problem Formulation

By strategically placing cloud proxies and dynamically adjusting the distribution of traffic across available paths, we can exploit the benefits of multipath transmission while minimizing the associated costs. This approach allows us to accommodate diverse SLOs and optimize system-wide performance, rather than focusing on individual flows in isolation. To achieve this goal, we propose a framework Prater that takes into account the characteristics of the underlying cloud network, including the location and pricing of cloud nodes, as well as the bandwidth constraints and costs associated with inter-node links. By considering these factors, our framework can make informed decisions about the placement of cloud proxies and the allocation of traffic across multiple paths to maximize revenue while minimizing costs.

In the remainder of this section, we first introduce the system models, which include the network model and transfer

request model. Build upon these models, we then formulate the corresponding optimization problem that captures the essential decision variables, constraints, and objective function.

A. System Models

Network model: In this work, we consider a set of the cloud regions that provide on-demand cloud instances to function as relay or proxy nodes to enhance data transmission throughput. Let N represent the set of cloud regions within the network. Each cloud region $i \in N$ is capable of providing on-demand instances to create transport proxies for data transfers. Furthermore, each server node in a cloud region $i \in N$ is associated with an egress capacity limit B_i, which constrains the maximum amount of data that can be transmitted out of the node at any given time.

Cost model: From the perspective of cloud customers, the cloud offers elasticity, enabling on-demand allocation of additional resources for transfer acceleration. However, this flexibility comes with costs, as cloud providers impose both instance charges and egress charges. Cloud providers typically charge customers for renting VMs or containers, with costs varying across different regions. These instance costs are typically region-specific, and customers must consider these expenses when allocating cloud resources for transfer acceleration. Egress charges are incurred for network traffic exiting a cloud region and are based on the volume of data transferred, rather than the transfer rate. As noted by [16], egress costs can also differ between regions. For intra-cloud transfers, which occur between two regions or zones within the same cloud, costs are typically higher for transfers between geographically distant endpoints compared to those between nearby endpoints. To accurately capture these costs, we introduce two key parameters $\mathtt{c}_i^{\mathtt{VM}}$ and $\mathtt{c}_i^{\mathtt{EGRESS}}$. $\mathtt{c}_i^{\mathtt{VM}}$ represents the cost associated with creating a proxy instance in cloud region i, and $\mathtt{c}_i^{\mathtt{EGRESS}}$ denotes the cost related to egress traffic leaving region i. In the context of using cloud resources for transfer acceleration, our cost model encompasses expenses related to creating cloud instances to deploy transport proxies, as well as costs associated with outgoing traffic from relay cloud regions.

Data transfer request model: To maintain the performance of cloud applications, bulk data transfer requests typically specify a volume of data to be moved within strict completion time constraints between datacenters. To model such a request f, we introduce a six-tuple $\langle s_f, d_f, \mathtt{ARR}_f, \mathtt{DL}_f, \mathtt{Vol}_f, \mathtt{Rev}_f \rangle$, s_f and d_f denote the source and destination nodes, respectively; $\mathtt{Vol}_f$ represents the volume of data to transfer; $\mathtt{ARR}_f$ indicates the arrival time of the request; $\mathtt{DL}_f$ specifies the deadline for completing the request; and $\mathtt{Rev}_f$ represents the revenue obtained from completing the transmission before $\mathtt{DL}_f$. We consider hard deadlines, namely the all-or-nothing model, as noted in previous work [31], [32], [37], [38]. This means that the system can only generate revenue if the data volume $\mathtt{Vol}_f$ is successfully transmitted to the specified destination on time; otherwise, the revenue becomes zero.

Cloud proxy forwarding model: To maximize the utilization of cloud resources, we consider a cloud proxy forwarding model that harnesses the power of TCP splitting in this work. This is motivated by the fact that forwarding using cloud proxies without TCP splitting capability has been shown to yield little to no throughput improvement in previous work [21], [22], [24], [39]. In contrast, TCP splitting has demonstrated significant benefits in cloud overlay networks, where a connection that uses a single TCP splitting proxy can achieve substantial improvement over a direct Internet connection [25]. Furthermore, performance can be additionally enhanced by employing Multi-Path TCP (MP-TCP). However, MP-TCP is not universally supported by communicating parties, which limits its practical applicability. To strike a balance between performance and practicality, we consider single-path TCP splitting as the forwarding model in this work. Table I summarizes the main symbols used.

TABLE I
KEY NOTATIONS USED IN PROBLEM FORMULATION

Symbol	Description
$\mathtt{N}$	Set of nodes, including cloud regions
$\mathtt{c}_i^{\mathtt{VM}}$	Instance cost of region i (per VM)
$\mathtt{c}_i^{\mathtt{EGRESS}}$	Egress cost of region i (per GB)
$\mathtt{BW}_i$	Bandwidth limit of an instance in region i (Gbps)
δ	Length of each time slot
$\mathtt{T}$	Number of time slots
$\mathtt{F}$	Set of data transfer requests
$\mathtt{P}_f$	Transmission path candidates for request f
$\mathtt{I}_{i,p}^{+}$	Binary indicator that equals 1 if path p built upon an instance in region i as middle proxy, and 0 otherwise
$\mathtt{Vol}_f$	Data volume of request f
$\mathtt{Rev}_f$	Revenue from completing request f
$\mathtt{ARR}_f$	Arrival time slot of request f
$\mathtt{DL}_f$	Deadline time slot of request f
$\mathtt{R}_{f,p}$	Achievable maximum transmission rate of path candidate $p \in \mathtt{P}_f$
x_f	Whether request f meets its deadline
y_i	Whether creating a proxy in region i for data transmission
z_i	Total data volume flowing out region i
$q_{f,p,t}$	Percentage of request f's data volume routed through path p during time slot t

B. Problem Formulation

Our goal is to maximize system profitability by minimizing cloud costs and maximizing the revenues from selectively serving requests before their deadlines. We achieve this by optimizing proxy deployment to improve data transfer performance while reducing elastic cloud resource costs.

However, selecting the optimal proxy deployment is a complex task due to the vast number of possible proxy locations, each with distinct transmission rates and costs, across various cloud regions. Specifically, the diverse range of cloud regions, each with its own unique characteristics, makes it challenging to determine the most effective proxy deployment strategy. Moreover, as the number of proxies along a single path increases, the possible combinations to create proxy paths grow exponentially, leading to a significant increase in complexity. To address this complexity, we simplify the use of cloud proxies from two perspectives: limiting the number of proxies per path and identifying path candidates prior to optimization. Firstly, each transfer path to include no more than two intermediate proxies, resulting in a maximum of three segments: source to first proxy, first proxy to second proxy (optional), and second proxy

to destination. This constraint significantly reduces the search space while still allowing flexibility in cloud path usage. Next, we generate all possible paths through permutations and combinations, adhering to the restriction on the number of proxies per path. The optimization process then selects a subset of these identified path candidates that yields the optimal performance. This approach strikes a balance between performance optimization and computational feasibility, enabling us to develop an effective optimization framework that efficiently navigates the complex landscape and identifies the optimal proxy deployment strategy. To fully leverage cloud resources for enhanced data transmission, we adopt a discrete time system where the overall system time is divided into discrete time slots. This framework allows us to optimize resource allocation with finer granularity. Within this framework, each transfer request can split traffic across multiple paths as well as across different time slots. This flexibility empowers us to optimize traffic allocation in both space (across paths) and time (across time slots) dimensions. The core idea is to make the following decisions.

- *Timely request completion.* For time-sensitive data transfer requests, we aim to complete as many requests as possible by their specified deadlines. We introduce a binary variable x_f to represent whether a request f can be finished on time, where $x_f = 1$ if the request f is completed on time, and $x_f = 0$ otherwise.
- *Proxy placement optimization.* We optimize the deployment locations of cloud proxies. We use a binary variable y_i to indicate whether a proxy is created in cloud region i for data transmission.
- *Egress traffic optimization.* The cross-region cost depends on the amount of traffic transmitted by cloud regions. We control the total amount of traffic z_i exiting region i for cost optimization.
- *Traffic allocation optimization.* We use multiple paths to parallelize acceleration, introducing continuous variables $q_{f,p,t}$ to denote the percentage of the data volume of request f allocated to flow over path p during time slot t. This variable captures the dynamic allocation of data across multiple paths and time, enabling a flexible and efficient distribution of bulk traffic.

With the decision variables defined above, the optimization process must satisfy several critical constraints, such as transfer deadlines and capacity limitations. The necessary constraints are are outlined below.

Transfer deadline constraint: Since data transfers that miss their deadlines typically become useless for users, we restrict from allocating resources to deadline-miss transfers to avoid useless data delivery and conserve resources for those can be finished before deadlines. Consistent with prior work [31], [32], [40], [41], each deadline-meeting request must be fully completed by the specified deadline, as expressed by equation (1). This ensures that resources are allocated efficiently and effectively to meet the required deadlines.

$$\forall f \in \mathtt{F}: \qquad \sum_{t=\mathtt{ARR}_f}^{\mathtt{DL}_f} \sum_{p \in \mathtt{P}_f} q_{f,p,t} = x_f \tag{1}$$

Proxy allocation constraint: We know that a cloud path can only be established for data transfers when an instance is spawned in that cloud region. This feasibility relationship is captured by the following constraint, denoted as Ineq. (2). This constraint ensures that a proxy is allocated only if there exists at least a request using a cloud path that traverses it. In other words, it prevents a request from using a cloud path without a corresponding proxy allocation. Let $\mathtt{P}_f$ denote the set of path candidates of request f, the proxy allocation constraint can be formulated as:

$$\forall i \in \mathtt{N}: \qquad \sum_{f \in \mathtt{F}} \sum_{t \in \mathtt{T}} \sum_{p \in \mathtt{P}_f} I^{+}_{i,p} q_{f,p,t} \leq y_i \tag{2}$$

Path capacity limitation constraint: The optimization process must adhere to several capacity constraints to ensure efficient and feasible data transmission. One of the key constraints is the path capacity constraint, which dictates that the percentage of data volume flowing over each path candidate during every time slot cannot exceed its capacity. This constraint is mathematically expressed as Ineq. (3), where $R_{f,p}$ represents the achievable transmission rate of path candidate p for request f, and δ denotes the length of each time slot. By enforcing this constraint, we can ensure that the data transmission rate over each path candidate does not exceed its capacity, thereby preventing network congestion and ensuring efficient data transmission.

$$\forall f \in \mathtt{F}, p \in \mathtt{P}_f, t \in \mathtt{T}: \mathtt{Vol}_f q_{f,p,t} \leq \mathtt{R}_{f,p} \delta \tag{3}$$

The achievable transmission rate of a candidate cloud path is determined by the bottleneck segment with the lowest throughput along its connection. To compute this rate, we need to analyze the split-TCP path, which divides the end-to-end connection into segments, typically between the client and the first proxy, between proxies, and from the last proxy to the destination server. The effective transmission rate, or throughput, of each segment is closely related to the TCP window size and round-trip time (RTT). By measuring the RTT for each segment, we can calculate the throughput for each segment using the following formula: $\mathtt{TP}_{\mathtt{segment}} = \frac{\mathtt{Window\ Size}}{\mathtt{RTT}}$. This formula is based on the mathematical model of TCP throughput, which takes into account the TCP window size and RTT [42]. To obtain the overall effective transmission rate of the split-TCP path, we need to identify the bottleneck segment, which is the segment with the lowest throughput. By computing the throughput of each segment, we can determine the achievable transmission rate of the cloud path using split-TCP. This approach allows us to accurately estimate the achievable transmission rate of a candidate cloud path, taking into account the characteristics of the underlying network and the TCP protocol. By using this information, we can make informed decisions about which cloud path to select for data transmission, ensuring optimal performance and efficiency.

Cloud capacity limitation constraint: To ensure efficient resource utilization and data delivery, the optimization process must also consider the capacity constraints of region instances and instance bandwidth. First, the allocated traffic load for each region instance must not exceed its bandwidth limit, as ensured by Ineq. (4), where $\mathtt{BW}_i$ represents the bandwidth limit

of an instance in region i. This constraint guarantees that the total traffic assigned to each region instance remains within its available bandwidth capacity, preventing over-allocation and potential bottlenecks. Secondly, the total egress traffic of each region must adhere to the paid amount, as ensured by Ineq. (5).

$$\forall i \in \mathtt{N}, t \in \mathtt{T}: \sum_{f\in\mathtt{F}}\sum_{p\in\mathtt{P}_f} I^+_{i,p}\frac{\mathtt{Vol}_f z_{f,p,t}}{\delta} \le \mathtt{BW}_i \tag{4}$$

$$\forall i \in \mathtt{N}: \sum_{t\in\mathtt{T}}\sum_{f\in\mathtt{F}}\sum_{p\in\mathtt{P}_f} I^+_{i,p}\mathtt{Vol}_f q_{f,p,t} \le z_i \tag{5}$$

By incorporating the above capacity constraints into the optimization process, the system can effectively manage data transmission while respecting the bandwidth limitations of paths, instances, and regions, thereby ensuring efficient resource utilization and data delivery.

Objective: maximizing system profitability: To ensure the practicality of our approach, we focus on maximizing the overall system profitability. This involves calculating the total revenue generated by delivering transfer requests before their specified deadlines, as well as accounting for the additional cloud costs incurred during data transfer. The total revenue produced by the transfer requests delivered before their specified deadlines can be calculated as: $\sum_{f\in\mathtt{F}}\mathtt{Rev}_f x_f$, where $\mathtt{Rev}_f$ represents the revenue generated by delivering transfer request f, and x_f is a binary variable indicating whether the request is delivered before its deadline. On the other hand, the additional cloud cost incurred during data transfer can be accounted for through $\sum_{i\in\mathtt{N}}(\mathtt{c}_i^{\mathtt{VM}} y_i + \mathtt{c}_i^{\mathtt{EGRESS}} z_i)$, where $\mathtt{c}_i^{\mathtt{VM}}$ represents the cost of instance i, y_i is a binary variable indicating whether instance i is used, $\mathtt{c}_i^{\mathtt{EGRESS}}$ represents the egress cost of instance i, and z_i is a variable representing the amount of egress traffic from instance i. By considering both the revenue generated by delivering data transfers and the incurred cloud costs, we can formulate the joint optimization problem of proxy placement and traffic allocation with the objective of profit maximization as follows.

$$\textbf{Maximize}: \sum_{f\in\mathtt{F}}\mathtt{Rev}_f x_f - \sum_{i\in\mathtt{N}}\left(\mathtt{c}_i^{\mathtt{VM}} y_i + \mathtt{c}_i^{\mathtt{EGRESS}} z_i\right) \tag{6}$$

$$\textbf{Subjectto}: (1), (2), (3), (4), (5)$$

$$\forall f \in \mathtt{F}: \quad x_f \in \{0,1\} \tag{7}$$

$$\forall i \in \mathtt{N}: \quad y_i \in \{0,1\} \tag{8}$$

$$\forall i \in \mathtt{N}: \quad z_i \ge 0 \tag{9}$$

$$\forall f \in \mathtt{F}, p \in \mathtt{P}_f, t \in [\mathtt{ARR}_f, \mathtt{DL}_f]: q_{f,p,t} \in [0,1] \tag{10}$$

$$\forall f \in \mathtt{F}, p \in \mathtt{P}_f, t \notin [\mathtt{ARR}_f, \mathtt{DL}_f]: \; q_{f,p,t} = 0 \tag{11}$$

This formulation enables us to maximize the overall system profitability while taking into account the costs associated with using cloud proxies. By solving this optimization problem, we can determine the optimal strategy for using cloud proxies to maximize system profitability.

The problem is known to be hard. Even if the cloud budget have been given and only the egress capacities is bottleneck, it is NP-hard to compute proxy deployment and traffic allocation to achieve the minimum average transfer completion time [24]. It is also NP-hard to compute traffic allocation for maximizing the number of transfers that can be finished before the deadlines, when the cloud proxies have been already deployed and only the path rate is bottleneck [37]. Overall, the NP-hardness of the joint optimization problem of proxy placement and traffic allocation highlights the need for efficient and effective algorithms for solving this problem in practice.

IV. Problem Decomposition and Algorithm Design

In this section, we present our solution to the complex profit maximization cloud proxy placement problem. Our approach involves breaking down the original complex problem into subproblems, developing efficient solution for each subproblem, integrating them strategically to obtain a comprehensive and effective solution.

A. Problem Decomposition and Algorithm Workflow

The profit-maximizing proxy placement and traffic allocation (PMPTA) problem is a complex optimization problem involving multiple variables and constraints. To make it more tractable, we decompose it into two subproblems, which can be addressed more easily.

- Subproblem 1: Request-Limited PMPTA (RL-PMPTA). Given a set of requests that the system commits to completing before their deadlines, the revenue generated from data transfers is fixed. In this case, solving the profit maximization cloud proxy placement problem is equivalent to minimizing cloud costs. The RL-PMPTA subproblem focuses on finding the optimal cloud resource allocation to minimize costs while meeting the deadlines of the accepted requests.
- Subproblem 2: Cost-Limited PMPTA (CL-PMPTA). Suppose the total cost for using cloud resources is fixed, solving the profit maximization cloud proxy placement problem can be transformed to maximizing the revenue of transferring deadline-meeting transfers. The CL-PMPTA subproblem focuses on finding the optimal set of requests to finish them before deadlines with strategically cloud resource allocation to maximize revenue while keeping costs within the fixed budget.

We propose a framework to obtain an effective schedule for the PMPTA problem by alternately solving the RL-PMPTA and CL-PMPTA subproblems. This approach allows us to balance the trade-off between revenue and cloud costs, leading to a more efficient and profitable solution.

Algorithm workflow: Our algorithm workflow involves alternately solving the RL-PMPTA and CL-PMPTA subproblems to obtain an effective schedule for the profit maximization cloud proxy placement problem. The workflow can be summarized as follows. The process begins with solving RL-PMPTA subproblem, which initially considers all requests as inputs. The RL-PMPTA subproblem determines the cloud proxy deployment schemes $\boldsymbol{y}$ and $\boldsymbol{z}$ that yield the minimum cloud cost while meeting the deadlines of all requests. Following the outputs from the RL-PMPTA, we adjust the value of $\boldsymbol{y}$ and $\boldsymbol{z}$ to $\boldsymbol{y}'$ and $\boldsymbol{z}'$ according to a predetermined rule. This involves either

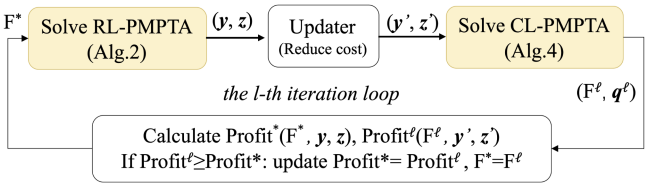


Fig. 2. Overall algorithm logic.

reducing the egress limit in under-utilized regions or removing the instances with the lowest utilization. We take $\boldsymbol{y}'$ and $\boldsymbol{z}'$ as the input of the CL-PMPTA subproblem and solve it to maximize transfer revenue under this capacity constraint. With the updated cloud resources $\boldsymbol{y}'$ and $\boldsymbol{z}'$, we may not able to accommodate all transfer requests within their deadlines. In this case, we will decline some requests in each iteration and update the list of requests that can be completed on time accordingly. This above process will run multiple times to find an effective final schedule, as shown in Fig. 2.

B. Algorithm Design to RL-PMPTA Problem

The objective of the RL-PMPTA problem is to minimize the cloud cost incurred to complete a set of requests before their respective deadlines. Specifically, given a set of requests $\mathtt{F}' \subseteq \mathtt{F}$, we aim to optimize the variables $\boldsymbol{z}$, $\boldsymbol{y}$, and $\boldsymbol{q}$ to minimize the cloud cost while ensuring that all requests are completed before deadlines. Note that in this problem, the values of variables $\boldsymbol{x}$ are fixed. This problem can be formulated as follows.

$$\textbf{Minimize}: \quad C = \sum_{i \in \mathtt{N}} \mathtt{C}_i^{\mathtt{VM}} y_i + \sum_{i \in \mathtt{N}} \mathtt{C}_i^{\mathtt{EGRESS}} z_i \tag{12}$$

$$\textbf{Subjectto}: \quad (2), (3), (4), (5)$$

$$\forall f \in \mathtt{F}': \quad \sum_{t=\mathtt{ARR}_f}^{\mathtt{DL}_f} \sum_{p \in \mathtt{P}_f} q_{f,p,t} = 1 \tag{13}$$

$$\forall n_i \in \mathtt{N}: \quad y_i \in \{0,1\} \tag{14}$$

$$\forall n_i \in \mathtt{N}: \quad z_i \geq 0 \tag{15}$$

$$\forall f \in \mathtt{F}', \forall p \in \mathtt{P}, \forall t \in [\mathtt{ARR}_f, \mathtt{DL}_f]: q_{f,p,t} \in [0,1] \tag{16}$$

$$\forall f \in \mathtt{F}', \forall p \in \mathtt{P}, \forall t \notin [\mathtt{ARR}_f, \mathtt{DL}_f]: q_{f,p,t} = 0 \tag{17}$$

It remains a challenging mixed-integer linear programming (MILP) problem, making it difficult to achieve an optimal solution within polynomial time complexity. To solve it efficiently, we develop an algorithm (Algorithm 1) based on relaxation-rounding. The algorithm relaxes the original RL-PMPTA model to obtain a fractional solution $\hat{y}$ for the cloud proxy deployment. It then uses a Pairwise Rounding algorithm to round the fractional $\hat{y}$ values to integers $\overline{y}$, which guarantees the activation of at least one new cloud proxy instance in each iteration. The integer solution $\overline{y}$ is assigned to $\boldsymbol{y}^*$. If the RL-PMPTA model is still infeasible with the updated $\boldsymbol{y}^*$, the process repeats for another iteration. The algorithm terminates when a feasible solution for the RL-PMPTA model is found, with the final $\boldsymbol{y}^*$ providing enough cloud instances to meet the transfer deadlines. To update

Algorithm 1: Algorithm for the RL-PMPTA Problem.

Input : Network $\mathtt{G} =< \mathtt{N}, \mathtt{E} >$, a set of requests $\mathtt{F}$ with committed deadlines, path candidates $\mathtt{P}$
Output: Cloud proxy deployment solution $\boldsymbol{y}$, $\boldsymbol{z}$
Set $\boldsymbol{y}^* = 0$; // Start with no instances deployed
while *RL-PMPTA is infeasible given* $\boldsymbol{y}^*$ **do**
  // Iteratively find minimal $\boldsymbol{y}^*$ for problem feasibility
  Substitute $\boldsymbol{y}^*$ in the Relaxed RL-PMPTA model and solve to obtain $\hat{\boldsymbol{y}}$;
  Apply Alg. 2 to round fractional $\hat{\boldsymbol{y}}$ to integers $\overline{\boldsymbol{y}}$;
  $\boldsymbol{y}^* = \overline{\boldsymbol{y}}$;
end
Substitute $\boldsymbol{y}^*$ in the RL-PMPTA and solve to obtain $\boldsymbol{z}$;
return $\boldsymbol{y}^*$, $\boldsymbol{z}$

the cloud proxy deployment solution, each iteration consists of the following steps. Step 1: Relaxing integer variables $\boldsymbol{y}$. By replacing the $y_i \in \{0,1\}$ with $y_i \in [0,1], \forall i \in \mathtt{N}$, we obtain in a linear programming problem, referred to as the Relaxed RL-PMPTA, which can be efficiently solved using standard commercial LP solvers. Step 2: Transforming the continuous $\hat{\boldsymbol{y}}$ to integers $\overline{\boldsymbol{y}}$ using pairwise rounding. Upon solving the Relaxed RL-PMPTA problem, we obtain a fractional solution denoted as $\hat{\boldsymbol{y}}$. To derive an integer solution $\overline{\boldsymbol{y}}$, we employ a pairwise rounding technique on $\hat{\boldsymbol{y}}$, as illustrated in Algorithm 2. The pairwise rounding technique involves pairing up the fractional values in $\hat{\boldsymbol{y}}$ and rounding them to the nearest integer. This approach ensures that the resulting integer solution $\overline{\boldsymbol{y}}$ is feasible and close to the optimal solution.

C. Algorithm Design to CL-PMPTA Problem

In the CL-PMPTA problem, the input is the the values of $\boldsymbol{y}$ and $\boldsymbol{z}$ obtained from solving RL-PMCCP problem, which are related to resource limits under certain cost budget. Our objective is to maximize the transfer revenue by optimizing decision variable $\boldsymbol{x}$ and $\boldsymbol{q}$, as expressed in (18)-(21). This problem is also a MILP problem, which can be challenging to solve optimally.

$$\textbf{Maximize}: \quad R = \sum_{f \in \mathtt{F}} \mathtt{REV}_f x_f \tag{18}$$

$$\textbf{Subjectto}: \quad (1), (2), (3), (4), (5)$$

$$\forall f: x_f \in \{0,1\} \tag{19}$$

$$\forall f \in \mathtt{F}, \forall p \in \mathtt{P}, \forall t \in [\mathtt{ARR}_f, \mathtt{DL}_f]: q_{f,p,t} \in [0,1] \tag{20}$$

$$\forall f \in \mathtt{F}, \forall p \in \mathtt{P}, \forall t \notin [\mathtt{ARR}_f, \mathtt{DL}_f]: q_{f,p,t} = 0 \tag{21}$$

To solve it efficiently, we employ a relaxation-rounding algorithm, which consists of the following steps:

Step 1: Relaxing integer variables $\boldsymbol{x}$. By substituting constraint (19) with (22), we obtain the relaxed LP model (referred to as the Relaxed CL-PMPTA), which is easier to solve.

$$x_f \in [0,1], \quad \forall f \in \mathtt{F} \tag{22}$$

Algorithm 2: Pairwise Rounding Algorithm.

Input : Fractional solution $\hat{\boldsymbol{y}}$
Output: Integer solution $\overline{\boldsymbol{y}}$

Set $\sigma_i = \hat{y}_i - \lfloor \hat{y}_i \rfloor, \forall i \in \mathtt{N}$;
Set $\mathtt{N}' = \mathtt{N} \setminus \{i \mid \sigma_i \in \{0,1\}\}$;
while $|\mathcal{N}'| > 1$ **do**
  Select $i_1, i_2 \in \mathtt{N}'$, and $i_1 \neq i_2$;
  Set $\eta_1 = \min\left\{1 - \sigma_{i_1}, \frac{c_{i_2}}{c_{i_1}}\sigma_{i_2}\right\}$
  Set $\eta_2 = \min\left\{\sigma_{i_1}, \frac{c_{i_1}}{c_{i_2}}(1 - \sigma_{i_2})\right\}$
  With probability $\frac{\eta_1}{\eta_1+\eta_2}$, set $\sigma'_{i_1} = \sigma_{i_1} + \eta_1, \sigma'_{i_2} = \sigma_{i_2} - \frac{c_{i_1}}{c_{i_2}}\eta_1$
  With probability $\frac{\eta_2}{\eta_1+\eta_2}$, set $\sigma'_{i_1} = \sigma_{i_1} - \eta_2, \sigma'_{i_2} = \sigma_{i_2} + \frac{c_{i_2}}{c_{i_1}}\eta_2$
  if $\sigma'_{i_1} \in \{0,1\}$ **then**
    Set $\overline{y}_{i_1} = \lfloor \hat{y}_{i_1} \rfloor + \sigma'_{i_1}$, $\mathtt{N}' = \mathtt{N}' \setminus \{i_1\}$
  end
  if $\sigma'_{i_2} \in \{0,1\}$ **then**
    Set $\overline{y}_{i_2} = \lfloor \hat{y}_{i_2} \rfloor + \sigma'_{i_2}$, $\mathtt{N}' = \mathtt{N}' \setminus \{i_2\}$
  end
end
if $|\mathcal{N}'| = 1$ **then**
  Set $\overline{y}_i = \lceil \hat{y}_i \rceil, \quad i \in \mathtt{N}'$
end
return $\overline{\boldsymbol{y}}$

Algorithm 3: Algorithm for the CL-PMPTA Problem.

Input : Set of considered requests $\mathtt{F}'$, path candidates $\mathtt{P}$, given deployment $\boldsymbol{y}$, $\boldsymbol{z}$ from RL-PMPTA
Output: Set of deadline-meeting requests $\mathtt{F}^*$ and traffic allocation solution $\boldsymbol{q}$

Solve Relaxed CL-PMPTA problem to obtain fractional solution $\{\hat{x}_f | \forall f \in \mathtt{F}'\}$
Sort requests in $\mathtt{F}'$ by their $\hat{x}_f$ values in descending order
Initialize $\mathtt{F}^* = \emptyset$, $\boldsymbol{q} = \boldsymbol{0}$
for $f \in \mathtt{F}'$ **do**
  Set $\boldsymbol{z}^* = \boldsymbol{z}$, $q_f = \emptyset$
  Sort $\mathtt{P}_f$ by $\mathtt{R}_{f,p}$ in descending order
  for $p \in \mathtt{P}_f$ **do**
    // Compute transferable volume on p
    $\mathtt{Vol}_p = \min(\mathtt{Vol}_f, \mathtt{R}_{f,p}\delta(\mathtt{DL}_f - \mathtt{ARR}_f), \min_{i \in p} z_i^*)$
    $\mathtt{Vol}_f = \mathtt{Vol}_f - \mathtt{Vol}_p$, $q^*_{f,p} = \mathtt{Vol}_f$
    for $i \in p$ **do**
      $z_i^* = z_i^* - \mathtt{Vol}_f$
    end
    if $\mathtt{Vol}_f = 0$ **then**
      $\boldsymbol{q} = \boldsymbol{q}^*$, $\boldsymbol{z} = \boldsymbol{z}^*$, $\mathtt{F}^* = \mathtt{F}^* \cup \{f\}$
      break; // Fully satisfied
    end
  end
end
return $\mathtt{F}^*$, $\boldsymbol{q}$

Step 2: Approximate rounding. After solving the relaxed CL-PMPTA problem, we obtain a fractional solution denoted as $\hat{\boldsymbol{x}}$. To derive an integer solution $\overline{\boldsymbol{x}}$, we first sort the requests in descending order based on the values of $\hat{\boldsymbol{x}}$, which indicate their contribution to the objective value. We then employ a greedy algorithm (see Algorithm 3) to achieve a near-optimal solution. Specifically, the algorithm sequentially examines each request and checks whether the current remaining capacity can accommodate it. Finally, identifies all requests $\mathtt{F}^*$ that can be completed before their deadlines, along with the associated allocation solution $\boldsymbol{q} = \{q_{f,p,t} | \forall f \in \mathtt{F}^*, \forall p \in \mathtt{P}, \forall t \in [\mathtt{ARR}_f, \mathtt{DL}_f]\}$. The relaxation-rounding algorithm provides an efficient and near-optimal solution to the CL-PMPTA problem.

Complexity analysis of solving CL-PMPTA problem. The time complexity of the algorithm for the CL-PMPTA problem is $\mathcal{O}(LP(|\mathtt{F}||\mathtt{P}||\mathtt{T}|, |\mathtt{N}||\mathtt{F}| + |\mathtt{N}||\mathtt{T}| + |\mathtt{F}||\mathtt{P}||\mathtt{T}|) + K|\mathtt{F}||\mathtt{P}|)$, where $LP(|\mathtt{F}||\mathtt{P}||\mathtt{T}|, |\mathtt{N}||\mathtt{F}| + |\mathtt{N}||\mathtt{T}| + |\mathtt{F}||\mathtt{P}||\mathtt{T}|)$ represents the complexity of solving the Relaxed CL-PMPTA problem (Line 1, Algorithm 3), K is the maximum number of cloud instances (proxies) per path, and $|\mathtt{F}|$ and $|\mathtt{P}|$ denote the total number of requests and path candidates, respectively. This complexity is evidently influenced by the values of $|\mathtt{F}|$, $|\mathtt{P}|$, and K. While $|\mathtt{F}|$ is fixed, limiting K significantly optimizes runtime by restricting the maximum proxies per path, thereby reducing path candidates $|\mathtt{P}|$. As described in Section III-B, we set $K = 2$. This decision is based on the observation that the marginal gain in throughput (or general performance) significantly diminishes as the number of proxies in a deployment increases beyond a small threshold, such as one or two, as demonstrated in studies like [43]. Although an initial proxy can offer substantial benefits, the incremental improvement of adding subsequent proxies is often minimal. Furthermore, employing more proxies directly translates to higher operational and deployment costs. Therefore, by allowing each cloud path to include at most two proxies ($K = 2$), we strike an effective balance between algorithmic efficiency, achievable performance gains (particularly throughput), and overall cost-effectiveness.

V. Performance Evaluation

A. Simulation Setup

Network parameters: In our experiments, we use realistic parameters based on the Google Cloud Platform (GCP) infrastructure. We set the rate limit for a cloud node at 2 Gbps, which represents the standard outbound bandwidth for a container on GCP [44]. The connectivity characteristics (RTT, leading to achievable throughput) between any pair of these locations are based on GCP's inter-region latency data [45]. Candidate paths are dynamically constructed for each request by considering direct connections and paths through one or two proxy nodes chosen from this pool of potential GCP locations, with path feasibility determined by the derived throughput and other constraints. The cost of setting up each instance is a, while the

cost of transferring data is b per GB [24], [46], [47]. This configuration provides a realistic framework for evaluating algorithm performance in cloud environments.

Workloads: Due to the lack of publicly available inter-datacenter traffic demand data, we generate data transfer requests based on the method used in CloudPilot [15], which is the most relevant work in this area. Each data transfer request consists of a fixed size of 2 GB that must be transmitted before a specified deadline δ. The source and destination nodes are randomly selected to effectively simulate typical traffic scenarios. Each request is associated with a positive revenue that correlates with the data size. This workload generation approach allows us to comprehensively evaluate the performance of our proposed algorithms under various traffic patterns and constraints.

Comparison schemes: To evaluate the performance of the solution PRATER proposed in this work, we compare it with the baseline without cloud acceleration and the state-of-the-art data transfer approaches, CloudPilot [15], based on split cloud proxies as this work, described as follows.

- `Direct`: This is the baseline method that transfers data without cloud acceleration, commonly used in most current inter-datacenter data transfer solutions [15].
- `F-FCT`: The flow-greedy algorithm proposed by CloudPilot [15] first computes the potential gains in reducing flow completion time (FCT) by using various paths based on different proxy locations, compared to the Direct transmission method for each flow request. It then sorts the requests by gain and processes them greedily, selecting those that maximize the gain sequentially until the budget is exhausted.
- `2-P RB`: The two-proxy greedy with rollback algorithm proposed by CloudPilot [15] creates a set of path candidates that includes all possible combinations of adding two proxy instances and evaluate them in a greedy manner. At each step, it evaluates the total FCT obtained for each candidate. If a candidate achieves a better total FCT, its allocation is retained. The algorithm terminates when no candidate improves the total FCT or when the budget is exhausted.
- `Optimal`: The optimal solution obtained by directly solving the MILP model discussed in Section III-B using the standard solver MOSEK [48]. Although this approach is computationally intensive, it provides an upper bound on performance and serves as a benchmark for evaluating the effectiveness of heuristic designs.

Performance metrics: We compare total profit, revenue, cost, and the proportion of deadline-meeting requests across a wide range of experimental parameters to evaluate the effectiveness of PRATER. Additionally, to evaluate efficiency, we also record the algorithm runtimes.

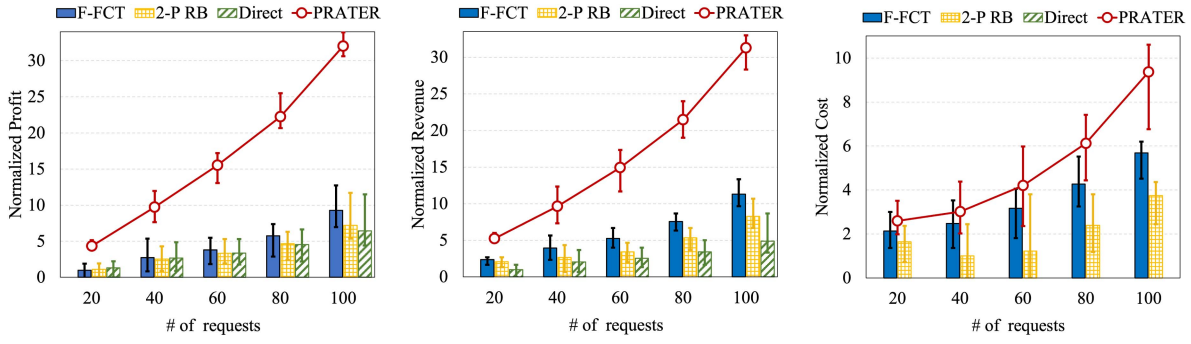
To ensure a fair comparison, CloudPilot algorithms (`F-FCT` and `2-P RB`) are budgeted to match the transfer cost incurred by PRATER. This standardized the cloud cost constraint, facilitating an equitable evaluation of their efficiency in optimizing data transfers within the given budget and deadline limits. Performance metrics for F-FCT and 2-P RB are derived directly from their original CloudPilot source code [24], eliminating re-implementation discrepancies and ensuring accurate capability reflection. To ensure robust results, each parameter setting is independently executed five times, and we report the best-case, average, and worst-case outcomes.

B. Experimental Results

1) General Performance: To comprehensively evaluate the performance of the baseline algorithms, we conducted three sets of experiments, varying workloads, transfer deadline tightness, and cloud cost configurations.

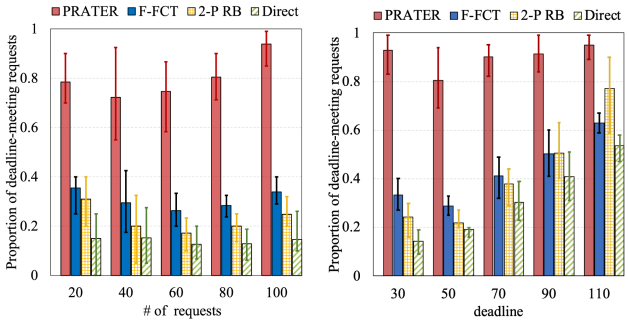
Impact of varying workloads: We first investigate the impact of different workloads on the performance of the algorithms by varying the number of requests. This experiment aims to evaluate how well the algorithms scale and adapt to handle increasing volumes of data transfer tasks, providing insights into their robustness and efficiency under varying workload conditions. Specifically, we evaluate the performance of the algorithms by gradually increasing the number of requests from 20 to 100. Fig. 3 shows the results of total profit, revenue and cost, which are normalized with respect to the result of F-FCT when the number of requests is 20. As the number of requests increases, PRATER consistently outperforms all compared algorithms across all experiments and performance metrics. Specifically, from Fig. 3(a), it can be observed that PRATER achieves the highest profit across all workload settings, and its advantage becomes more pronounced as the number of requests increases. The 2-P RB algorithm ranks second, while the F-FCT and Direct algorithms yield lower profits and exhibit similar performance. PRATER achieves an average improvement of approximately $4.04\times$, $3.82\times$, and $4.10\times$ over Direct, F-FCT, and 2-P RB, respectively. These findings highlight the superiority of PRATER in handling varying workload conditions and its ability to maintain high performance even as the number of requests grows. Moreover, the results show that the total profit for all algorithms grows almost linearly as the number of requests increases. This is due to the increased optimization space under a higher number of requests, allowing the algorithms to select more suitable flows, thereby enhancing revenue. The costs of using cloud servers can be distributed across more requests, resulting in lower total costs for accelerating each transmission request, which improves overall profit margins. The normalized revenue results in Fig. 3(b) follow a similar trend to the profits, with PRATER generating the highest revenue, followed by 2-P RB, and then F-FCT and Direct. This indicates that cloud-acceleration solutions, PRATER and CloudPilot (2-P RB and F-FCT), are more effective in selecting high-yield requests for execution. In terms of normalized cost, as shown in Fig. 3(c), Direct incurs no cloud cost, PRATER has slightly higher costs compared to 2-P RB and F-FCT. This is because PRATER, by accepting more requests, naturally incurs a higher total cost. However, these additional costs are significantly outweighed by the significant improvement in profit, demonstrating a highly efficient return on investment.

To investigate into the performance, we also collect the deadline satisfaction of requests for achieved by compared algorithms. Fig. 4(a) shows that PRATER consistently achieves the highest satisfaction ratio, maintaining above 0.8. F-FCT and 2-P



(a) Total Profit (b) Total Revenue (c) Total Cost

Fig. 3. Total profit, revenue, and cost under varying workloads (normalized to the minimum values observed across all compared algorithms and experiments).



(a) Varying workloads (b) Varying deadline tightness

Fig. 4. Deadline satisfaction of requests.

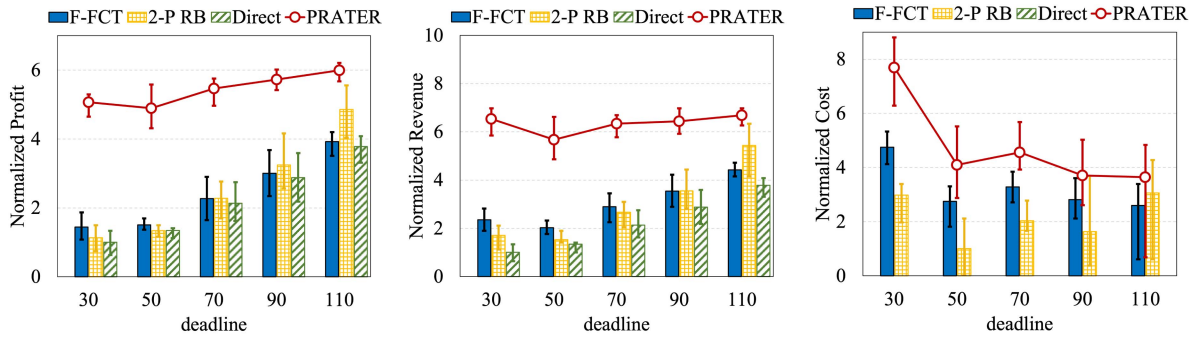
RB perform similarly, with their satisfaction ratios decreasing more rapidly as the number of requests increases. The Direct algorithm has the lowest satisfaction ratio, dropping below 0.2 when the workload reaches 100 requests. This results can explain why PRATER always outperforms benchmarks in Fig. 3. We also observe that the performance improvement of F-FCT and 2-P RB over the Direct algorithm is relatively small. This could be because although F-FCT and 2-P RB leverage the cloud for acceleration, they are not aware of transfer deadlines. As a result, they may unnecessarily accelerate some requests while missing the opportunity to complete more urgent transfers within their deadlines. Consequently, they incur cloud costs without generating revenue, leading to lower total profits.

Impact of deadline tightness: In this set of experiments, we evaluate the effect of transfer deadline tightness on algorithm performance by varying the value of the deadline parameter δ. By adjusting δ, we simulate scenarios with different levels of time constraints, ranging from relaxed to stringent deadlines. Fig. 5 illustrates the normalized total profit, revenue, and cost under varying deadlines (from 30 to 110) for the compared algorithms (F-FCT, 2-P RB, Direct, and PRATER). In Fig. 5(a), PRATER consistently achieves the highest profit across all deadline settings. PRATER achieves an average improvement in total profit of approximately $2.76\times$, $2.44\times$, and $2.56\times$ compared to Direct, F-FCT, and 2-P RB, respectively. The profits of F-FCT, 2-P RB, and Direct are lower and exhibit similar trends, increasing as the deadline becomes more relaxed. Fig. 5(b) shows the normalized revenue results, which follow a similar pattern to the profits. PRATER generates the highest revenue, particularly at tighter deadlines, while the other three algorithms have lower revenues. Regarding normalized cost, as depicted in Fig. 5(c), PRATER incurs the highest cost across all deadlines, with the gap between PRATER and the other algorithms being largest at the tightest deadline. The costs of PRATER decrease as the deadline becomes less strict, while F-FCT and 2-P RB becomes less stable.

Fig. 4(b) shows the deadline satisfaction ratio under varying deadline tightness. As the deadline becomes tighter, all compared algorithms exhibit a decline in their ability to meet deadline. However, PRATER maintains its superiority, with a satisfaction ratio above 0.8 even at the tightest deadline of 30. F-FCT and 2-P RB have comparable performance, while the satisfaction ratio of Direct solution falls sharply with tighter deadlines, below 0.2 at the tightest deadline.

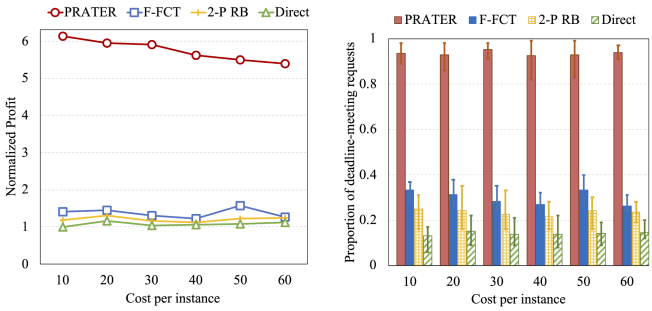
Impact of cloud cost: We also vary the cost settings of instances and bandwidth to evaluate the effectiveness of the algorithms in handling diverse cloud cost configurations. By altering the pricing models and cost parameters, we can assess how the algorithms adapt and optimize data transfers in different cost scenarios. This experiment provides insights into the cost-efficiency and flexibility of the algorithms in real-world cloud environments where pricing may vary across providers and over time.

Fig. 6 presents the performance of the compared algorithms (PRATER, F-FCT, 2-P RB, and Direct) under different instance costs, ranging from 10 to 60 cost units per instance. Fig. 6(a) shows the normalized total profit for each algorithm. As the instance cost increases, the total profit keep relatively stable for all algorithms. However, PRATER consistently achieves the highest profit across all instance costs, maintaining a significant advantage over the other algorithms. Its profit is approximately $6.13\times$, $4.33\times$, and $5.22\times$ that of Direct, F-FCT, and 2-P RB, respectively. F-FCT, 2-P RB and Direct have similar performance. Fig. 6(b) illustrates the deadline satisfaction ratio, where PRATER maintains the highest satisfaction ratio, consistently above 0.9, even as the instance cost increases. F-FCT and 2-P RB exhibit comparable performance, with their satisfaction ratios being lower than PRATER but higher than Direct. The Direct algorithm has the lowest deadline satisfaction ratio.



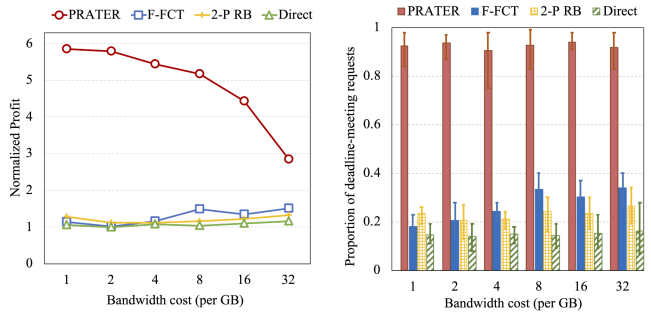
(a) Total Profit (b) Total Revenue (c) Total Cost

Fig. 5. Total profit, revenue, and cost under varying deadlines (normalized to the minimum values observed across all compared algorithms and experiments).



(a) Total Profit (Normalized) (b) Deadline Satisfaction

Fig. 6. Performance under different instance costs (a).

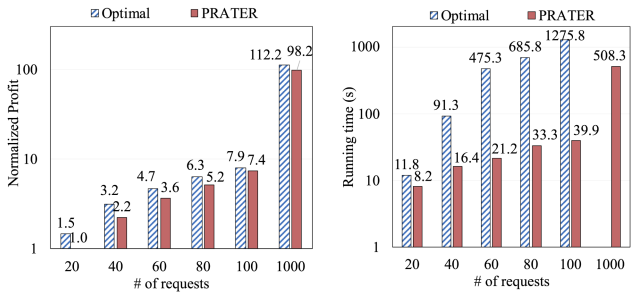


(a) Total Profit (Normalized) (b) Deadline Satisfaction

Fig. 7. Performance under different egress costs (b).

In scenarios with varying bandwidth costs, as Fig. 7 shown, PRATER achieves a total profit approximately $5.53\times$, $5.12\times$, and $4.54\times$ greater than the other three algorithms, respectively. PRATER consistently maintains high performance across different cost conditions, largely due to its cost-aware proxy deployment scheme and deadline-aware scheduling strategy, which together allow for higher improvement through more effective cost and revenue control when scheduling data transfers.

We can also see that PRATER is more sensitive to changes in bandwidth costs than instance costs, compared with other algorithms. As shown in Figs. 6(a) and 7(a), the total profit



(a) Total Profit (b) Running time

Fig. 8. PRATER vs. Solver-based optimal solution. The solver-based optimal solution for 1000 requests is obtained by solving the relaxed Linear Programming (LP).

achieved by PRATER decreases as instance and bandwidth costs increase, but the decline is much smaller for instance costs. This is because that the number of proxies is relatively same for all algorithms, while PRATER incurs more egress cost for serving more transfers. Moreover, as illustrated in Figs. 6(b) and 7(b), the proportion of deadline-meeting requests for each algorithm remains largely unaffected by changes in cost. This stability is primarily due to the number of requests in this scenario is relatively large, which allows instance costs to be distributed across more requests. This reduces the cost per request, enabling a stable proportion of deadline-meeting requests even as overall costs increase.

2) Comparison to the Optimal Solution: Lastly, we compare the performance of our algorithm against solver-based optimal solution in terms of normalized profit and running time under varying numbers of requests $|F|$ (from 20 to 1000). In our approach, the path set size $|P|$ is not an independent variable; it is determined by $|F|$ as our algorithm generates eligible paths for each request. Consequently, an increase in $|F|$ inherently leads to an increased total number of potential paths generated and evaluated, thus implicitly scaling the effective $|P|$ considered. Experiments are conducted on a personal laptop (AMD R7-4800 U processor, 16 GB memory). The simulations used Python 3.8 and PyCharm, with no other programs running to ensure stable performance. Fig. 8(a) shows that as the number of requests increases, the profit grows for both and the optimal

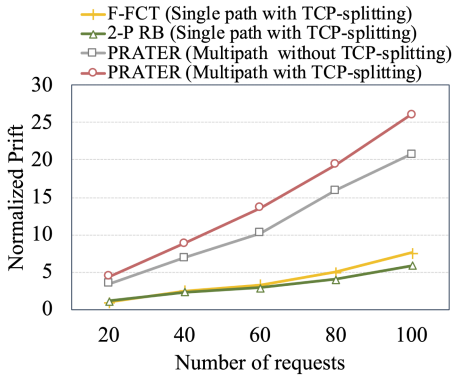


Fig. 9. Component-wise comparison.

consistently achieves higher profits compared to our algorithm. Although the gap between the two algorithms widens as the number of requests increases, it remains relatively small. Fig. 8(b) illustrates the running time (in seconds) of both algorithms on a logarithmic scale. The solver-based optimal solution exhibits an exponential increase in running time as the number of requests grows. In contrast, the running time of our algorithm PRATER increases at a much slower rate. At 100 requests, the optimal solution takes 1275.8 seconds to solve, while PRATER only requires 39.9 seconds, which is approximately 32 times faster. In summary, the experimental results demonstrate that while the solver-based optimal solution achieves slightly higher profits compared to PRATER, it comes at the cost of significantly longer running times. PRATER, on the other hand, provides a good balance between profit and running time, making it more suitable for larger-scale scenarios with a higher number of requests.

3) Component-Wise Comparison: Fig. 9 presents an ablation study that quantifies the individual and combined contributions of multipath routing and TCP-splitting to profit, as the number of requests increases. We benchmark PRATER's performance against established single-path TCP proxy algorithms, specifically Cloudpilot's F-FCT and 2-P RB. The results demonstrate that the multipath routing capability of PRATER contributes largely in achieving high profit. This substantial performance improvement by multipath alone stems from its ability to effectively distribute traffic across an average of 3 to 6 available paths, thereby significantly outperforming single-path alternatives, even when those alternatives incorporate TCP-splitting. Moreover, the integration of TCP-splitting within the multipath PRATER framework further enhances performance, resulting in even higher profit levels.

4) Limitations and Future Work: While PRATER demonstrates promising performance, our evaluation, particularly for thousands of requests (Fig. 7(b)), indicates a certain degree of scalability challenge that warrants future investigation. Beyond this, a full quantitative analysis of computational overhead, assumptions on RTT stability, TCP window sizes, and scalability to 10+ cloud regions is beyond the current scope. Future work will focus on optimizing for larger scales, rigorously analyzing its resource consumption, and developing adaptive mechanisms for more dynamic network conditions to enhance its robustness and applicability.

VI. Conclusion

In this paper, we address the challenge of timely bulk data transfers across datacenters under deadline constraints. Given the slow growth of inter-datacenter bandwidth, we propose a new approach that leverages elastic cloud proxies and multipath routing to accelerate data transfers. Recognizing the cost associated with cloud proxies, we joint optimize cloud proxy deployment and traffic allocation, aiming to maximize the system profit. To effectively tackle this NP-hard problem, we present an algorithm that incorporates techniques such as problem decomposition, relaxation-rounding, and iterative searching. It can efficiently find near-optimal solutions for cloud proxy deployment and traffic allocation within a practical scheduling time, enabling the system to accept and fulfill as many transfer requests as possible while minimizing the costs associated with cloud proxies and bandwidth consumption. Extensive simulations demonstrate that the approach we proposed significantly outperforms existing direct and cloud-based transfer solutions, substantially enhancing overall profitability. While optimizing performance and cost, we recognize that security and privacy implications of using third-party relays in proxy-based transfers are crucial and warrant further exploration.

References

[1] S. Deng et al., "Cloud-native computing: A survey from the perspective of services," *Proc. IEEE*, vol. 112, no. 1, pp. 12–46, Jan. 2024.
[2] S. S. Gill et al., "Modern computing: Vision and challenges," *Telematics Informat. Rep.*, 2024, Art. no. 100116.
[3] P. Plebani, S. Schulte, D. A. Tamburri, and S. Dustdar, "Service-oriented computing: A trajectory for research to 2030," *IEEE Internet Comput.*, vol. 28, no. 3, pp. 59–63, May/Jun. 2024.
[4] L. Luo, H. Yu, K.-T. Foerster, M. Noormohammadpour, and S. Schmid, "Inter-datacenter bulk transfers: Trends and challenges," *IEEE Netw.*, vol. 34, no. 5, pp. 240–246, Sep./Oct. 2020.
[5] B. Fatemipour, Z. Zhang, and M. St-Hilaire, "A survey on replica transfer optimization schemes in geographically distributed data centers," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 6, pp. 6301–6317, Dec. 2024.
[6] S. Luo, R. Wang, and H. Xing, "Efficient inter-datacenter allreduce with multiple trees," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 5, pp. 4793–4806, Sep./Oct. 2024.
[7] S. Luo, R. Wang, K. Li, and H. Xing, "Efficient cross-cloud partial reduce with crew," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 11, pp. 2224–2238, Nov. 2024.
[8] C.-Y. Hong et al., "Achieving high utilization with software-driven wan," in *Proc. ACM SIGCOMM 2013 Conf. SIGCOMM*, 2013, pp. 15–26.
[9] S. Jain et al., "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.
[10] C.-Y. Hong et al., "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proc. ACM Special Int. Group Data Commun. Conf.*, 2018, pp. 74–87.
[11] K. Oh, M. Zhang, A. Chandra, and J. Weissman, "Network cost-aware geo-distributed data analytics system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1407–1420, Jun. 2022.
[12] L. Luo, Q. Jin, J. Xie, G. Sun, and H. Yu, "Cost-efficient scheduling of multicast transfers with deadline guarantees across edge datacenters," *IEEE Trans. Serv. Comput.*, vol. 16, no. 1, pp. 191–205, Jan./Feb. 2023.
[13] G. Zhao, J. Wang, H. Xu, Z. Yu, and C. Qiao, "COIN: Cost-efficient traffic engineering with various pricing schemes in clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
[14] Z. Yang et al., "Skypilot: An intercloud broker for sky computing," in *Proc. 20th USENIX Symp. Networked Syst. Des. Implementation*, 2023, vol. 23, pp. 437–455.
[15] K. Toledo, D. Breitgand, D. Lorenz, and I. Keslassy, "Cloudpilot: Flow acceleration in the cloud," *Comput. Netw.*, vol. 224, 2023, Art. no. 109610.

[16] P. Jain, S. Kumar, S. Wooders, S. G. Patil, J. E. Gonzalez, and I. Stoica, "Skyplane: Optimizing transfer cost and throughput using {Cloud-Aware} overlays," in *Proc. 20th USENIX Symp. Networked Syst. Des. Implementation*, 2023, pp. 1375–1389.
[17] C. X. Cai, F. Le, X. Sun, G. G. Xie, H. Jamjoom, and R. H. Campbell, "CRONets: Cloud-routed overlay networks," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 67–77.
[18] S. Wooders et al., "Cloudcast: High-throughput, cost-aware overlay multicast in the cloud," in *Proc. 21st USENIX Symp. Networked Syst. Des. Implementation*, 2024, vol. 24, pp. 281–296.
[19] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *Proc. Workshop Hot Top. Operating Syst.*, 2021, pp. 26–32.
[20] A. Bergman et al., "Pied piper: Rethinking internet data delivery," 2018, *arXiv:1812.05582*.
[21] V. Farkas, B. Héder, and S. Nováczki, "A split connection TCP proxy in LTE networks," in *Proc. Inf. Commun. Technol.: 18th EUNICE/IFIP WG 6.2, 6.6 Int. Conf., Budapest, Hungary, Aug. 29-31, 2012 Proc. 18*, Springer, 2012, pp. 263–274.
[22] B. H. Kim, D. Calin, and I. Lee, "Enhanced split TCP with end-to-end protocol semantics over wireless networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2017, pp. 1–6.
[23] S. Kopparty, S. V. Krishnamurthy, M. Faloutsos, and S. K. Tripathi, "Split TCP for mobile ad hoc networks," in *Proc. Glob. Telecommun. Conf.*, 2002, vol. 1, pp. 138–142.
[24] K. Toledo, D. Breitgand, D. Lorenz, and I. Keslassy, "Cloudpilot project," Accessed: Dec. 23, 2024. [Online].Available: https://github.com/kfirtoledo/CloudPilot_Project
[25] F. Lai, M. Chowdhury, and H. Madhyastha, "To relay or not to relay for {Inter-Cloud} transfers?," in *Proc. 10th USENIX Workshop Hot Top. Cloud Comput.*, 2018, pp. 1–6.
[26] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *Proc. 18th USENIX Symp. Networked Syst. Des. Implementation*, 2021, pp. 201–216.
[27] Z. Xu et al., "Teal: Learning-accelerated optimization of wan traffic engineering," in *Proc. ACM SIGCOMM 2023 Conf.*, 2023, pp. 378–393.
[28] U. Krishnaswamy, R. Singh, N. Bjørner, and H. Raj, "Decentralized cloud wide-area network traffic engineering with {BLASTSHIELD}," in *Proc. 19th USENIX Symp. Networked Syst. Des. Implementation*, 2022, pp. 325–338.
[29] Y. Zhang et al., "BDS+ : An inter-datacenter data replication system with dynamic bandwidth separation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 918–934, Apr. 2021.
[30] L. Pápay et al., "An exabyte a day: Throughput-oriented, large scale, managed data transfers with effingo," in *Proc. ACM SIGCOMM 2024 Conf.*, 2024, pp. 970–982.
[31] H. Zhang et al., "Guaranteeing deadlines for inter-datacenter transfers," in *Proc. 10th Eur. Conf. Comput. Syst.*, 2015, pp. 1–14.
[32] X. Jin et al., "Optimizing bulk transfers with software-defined optical WAN," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 87–100.
[33] P. Kathiravelu, M. Chiesa, P. Marcos, M. Canini, and L. Veiga, "Moving bits with a fleet of shared virtual routers," in *Proc. IFIP Netw. Conf. (IFIP Network.) Workshops*, 2018, pp. 1–9.
[34] O. Haq, C. Doucette, J. W. Byers, and F. R. Dogar, "Judicious QoS using cloud overlays," in *Proc. 16th Int. Conf. Emerg. Netw. EXperiments Technol.*, 2020, pp. 371–385.
[35] K. Chen and N. Yang, "BwShare: Efficient bandwidth guarantee in cloud with transparent share adaptation," *Comput. Netw.*, vol. 170, 2020, Art. no. 107095.
[36] P. Zhao, J. You, and X. Yuan, "Circling reduction algorithm for cloud edge traffic allocation under the 95th percentile billing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 5, pp. 4254–4269, Oct. 2024.
[37] L. Luo, H. Yu, Z. Ye, and X. Du, "Online deadline-aware bulk transfer over inter-datacenter wans," in *Proc. IEEE INFOCOM 2018-IEEE Conf. Comput. Commun.*, 2018, pp. 630–638.
[38] L. Luo, L. Yu, T. Ma, and H. Yu, "Flexible and efficient multicast transfers in inter-datacenter networks," in *Proc. IEEE/ACM 30th Int. Symp. Qual. Serv.*, 2022, pp. 1–10.
[39] M. Luglio, M. Y. Sanadidi, M. Gerla, and J. Stepanek, "On-board satellite" split TCP" proxy," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 2, pp. 362–370, Feb. 2004.
[40] S. Gandhi and Y. Viniotis, "Guarantees for mix-flows in inter-datacenter wans in single and federated clouds," in *Proc. IEEE 45th Conf. Local Comput. Netw.*, 2020, pp. 244–255.
[41] X. Dong and B. Cai, "Balanar: Balancing deadline guarantee and jain's fairness for inter-datacenter transfers," *Comput. Netw.*, vol. 235, 2023, Art. no. 109988.
[42] F. Kelly, "Mathematical modelling of the internet," in *Proc. Math. Unlimited—2001 Beyond*, 2001, pp. 685–702.
[43] G. Siracusano, R. Bifulco, S. Kuenzer, S. Salsano, N. B. Melazzi, and F. Huici, "On the fly TCP acceleration with miniproxy," in *Proc. Workshop Hot Topics Middleboxes Netw. Function Virtualization*, 2016, pp. 44–49.
[44] G. Inc, "GCP machine resource," Google Cloud, Tech. Rep., Accessed: Dec. 23, 2024. [Online]. Available: https://cloud.google.com/compute/docs/general-purpose-machines
[45] G. Inc, "Google cloud inter-region latency and throughput," Google Cloud, Tech. Rep., Accessed: Dec. 23, 2024. [Online]. Available: https://lookerstudio.google.com/reporting/fc733b10-9744-4a72-a502-92290f608571/page/70YCB
[46] G. Inc, "Virtual private cloud pricing," Google Cloud, Tech. Rep., Accessed: Dec. 23, 2024. [Online]. Available: https://cloud.google.com/vpc/pricing
[47] G. Inc, "Price list," Google Cloud, Tech. Rep., Accessed: Dec. 23, 2024. [Online]. Available: https://cloud.google.com/pricing/list
[48] M. ApS, "Mosek," Tech. Rep. Accessed: Dec. 23, 2024. [Online]. Available: https://www.mosek.com/

Long Luo (Member, IEEE) received the MS and PhD degrees in communication engineering from the University of Electronic Science and Technology of China UESTC, in 2015 and 2020, respectively. She is currently an associate professor with the (UESTC). Her research interests include networking and distributed systems.

Yunxiang Zhou (Student Member, IEEE) is currently working toward the master's degree with the School of Information and Communication Engineering, the University of Electronic Science and Technology of China. His research interests include cloud computing and traffic engineering.

Linjian Yu received the master's degree with the School of Information and Communication Engineering, the University of Electronic Science and Technology of China. His research interests include cloud computing and traffic engineering.

Jin Shen (Student Member, IEEE) is currently working toward the master's degree with the School of Information and Communication Engineering, the University of Electronic Science and Technology of China. His research interests focuses on serverless computing.

Hongfang Yu (Senior Member, IEEE) received the BS degree in electrical engineering in 1996 from Xidian University, and the MS and PhD degrees in communication and information engineering in 1999 and 2006, respectively, from the University of Electronic Science and Technology of China UESTC. She is currently a professor with the (UESTC). Her research interests include AI network systems and network security.

Schahram Dustdar (Fellow, IEEE) received the PhD degree in business informatics with the University of Linz, Austria, in 1992. He is currently a full professor of computer science (informatics) with a focus on internet technologies heading the Distributed Systems Group, TU Wien, Wein, Austria. He has been the Chairman of the Informatics Section of the Academia Europaea, a member of the IEEE Conference Activities Committee, the Section Committee of Informatics of the Academia Europaea, and the Academia Europaea: The Academy of Europe, Informatics Section. He was the recipient of the ACM Distinguished Scientist Award and IBM Faculty Award.