Efficient Multi-WAN Transport for 5G with Otter

Mary Hogan, *Oberlin College;* Gerry Wan, *Google;* Yiming Qiu, *University of Michigan;* Sharad Agarwal and Ryan Beckett, *Microsoft;* Rachee Singh, *Cornell University;* Paramvir Bahl, *Microsoft*

https://www.usenix.org/conference/nsdi25/presentation/hogan

This paper is included in the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation.

April 28–30, 2025 • Philadelphia, PA, USA

978-1-939133-46-5

Open access to the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

Efficient Multi-WAN Transport for 5G with OTTER

Mary Hogan[‡], Gerry Wan[•], Yiming Qiu[†], Sharad Agarwal[*], Ryan Beckett[*], Rachee Singh[°], Paramvir Bahl[*]
[‡]Oberlin College, [•]Google, [†]University of Michigan, [*]Microsoft, [°]Cornell University

Abstract

In the ongoing *cloudification of 5G*, software network functions (NFs) are replacing fixed-function network hardware, allowing 5G network operators to leverage the benefits of cloud computing. The migration of NFs and their management to the cloud causes 5G traffic to traverse an operator's wide-area network (WAN) to the cloud WAN that hosts the datacenters (DCs) running 5G NFs and applications. However, achieving end-to-end performance for 5G traffic across two WANs is hard. Placing 5G flows across two WANs with different performance and reliability characteristics, edge and DC resource constraints, and interference from other flows is different and more challenging than single-WAN traffic engineering. We address this challenge and show that orchestrating paths across a multi-WAN overlay[1] allows us to achieve average 13% more throughput, 15% less RTT, 45% less jitter, or reduce average loss from 0.06% to under 0.001%. We implement our multi-WAN 5G flow placement in a scalable optimization prototype that allocates 26%–45% more bytes on the network than greedy baselines while also satisfying the service demands of more flows.

1 Introduction

The majority of 5G access networks today use legacy telecom equipment and architecture. The next generation of 5G is designed to deliver superior Quality-of-Service (QoS) through a combination of new frequency bands, new radio technology, and the *cloudification* of telecom network functions (NFs). These innovations are expected to revolutionize 5G and subsequent generations of access networks. In this work we focus on the implications of improved QoS and cloudification of access on wide-area networking (WAN).

As part of the cloudification of 5G, monolithic and proprietary hardware implementations of mobile wireless NFs are evolving into disaggregated software-based NFs that run on commodity off-the-shelf compute [79]. This disaggregation allows 5G operators to deploy their virtualized radio access networks (vRAN) on edge compute close to end users, and packet cores in datacenters (DCs) [56, 61].

Cloudification is driving performance-sensitive inter-domain traffic onto cloud WANs. For instance, traffic from user equipment (UEs) traverses a 5G operator's WAN to a cloud WAN to reach 5G Core NFs and applications hosted in the cloud [56, 61]. The burden of providing QoS to this traffic is now shared between two WANs: the 5G operator's WAN and the cloud WAN. However, existing traffic management mechanisms lack the ability to orchestrate paths shared between two WANs. In fact, 5G networks are already bottlenecked on WAN performance [91], and this will get worse as the upcoming 5G New Radio (5G NR) unlocks ultra-low latency and high-bandwidth radio modes.

Improving the performance of inter-domain 5G traffic is challenging for three main reasons. First, cloudified access traffic is destined to cloud DCs that host 5G NFs and applications. The location of these 5G services in the cloud is itself dynamic [33, 80] and conditional on the availability of compute resources, making it hard to achieve performance-optimal routing. Second, the primary mechanism to achieve network performance goals in WANs today is *intra-domain traffic engineering* (TE) [39, 41, 44] which does not capture inter-domain route performance goals or fine-grained service objectives of 5G traffic. Third, the dynamic nature of 5G traffic requires on-demand flow placement, which is not achievable by traditional TE mechanisms that operate on periodic (*e.g.,* 5 minute) allocation intervals.

We develop OTTER (Overlay Traffic Transport and Efficient Resource allocation), a system that provides efficient, on-demand transport across operator and cloud WANs for 5G traffic with differing QoS needs. It uses a multi-WAN overlay to co-optimize the placement of 5G NFs and applications, and the network paths of 5G traffic. OTTER dynamically computes optimal routes for multiple performance metrics and implements them using scalable forwarding mechanisms involving VMs and cloud routing gateways. OTTER is cloud-agnostic, allowing operators to deploy it on cloud providers of their choice.

Our contributions. OTTER makes two key contributions. First, it develops an efficient algorithm for allocating network and compute resources to 5G demands across operator and cloud WANs. This algorithm forms the core of OTTER's multi-WAN SDN (Software Defined Networking) controller. Second, OTTER implements the optimal routing and compute placement strategy of the controller using a novel multi-WAN forwarding mechanism. We develop the multi-WAN forwarding mechanism as part of the OTTER Orchestrator, which programmatically creates multi-WAN overlay networks with fine-grained routing policies for operators to deploy their 5G NFs.

[1] OTTER topology release: https://OTTER-5GWAN.github.io/

We design OTTER with the observation that the operator and cloud have newly aligned economic incentives for 5G, as their business relationship goes beyond peering to also include SaaS and compute. The OTTER Orchestrator scales to country-wide sizes of large operators by leveraging cloud-native support. The OTTER Optimizer dynamically places both compute and network workloads on multi-WAN overlays, for multiple traffic classes simultaneously.

Results. OTTER's path orchestration across two commercial WANs outperforms both public paths as well as topologies designed for enterprise-grade privacy and security. Our deployment topology scaled out to the continental US shows 13% higher throughput on average, with a best case of 136% higher, which can be 6-10 Gbps more. For flows that care about round trip time (RTT), we achieve 15% average reduction, up to 42 ms less. Jitter-sensitive flows can expect an average reduction of 45% and loss is 0.06% less on average. OTTER's on-demand allocation outperforms greedy baselines by up to 26-45% in allocated bytes (the amount of demand that is satisfied), and comes close to what an infeasible, infinitely-fast optimizer can achieve.

2 Cloudification of Access

Traditionally, vendors have implemented cellular technology on monolithic and proprietary hardware. However, 5G cellular networks are being re-architected using SDN and NF virtualization (NFV) [9]. This new architecture enables operators to scale up and out NFs on statistically multiplexed compute at operator edge sites (at or near cellular base stations and mobile switching centers), cloud edge sites (at hosting centers or peering points), and cloud DCs [55]. The use of the same compute management plane across all three allows NFs and interactive applications (*e.g.,* AR/VR) to be deployed flexibly, as user demand dictates. This allows operators to unlock the elasticity and efficiency of cloud computing. This convergence of access networks and hyperscalers is known as the *cloudification* of 5G [27, 64, 65].

2.1 Implications for WANs

Performance-sensitive demands over multiple WANs. The deployment of 5G applications and NFs on the cloud puts the cloud WAN on the critical path of 5G traffic that originates in the operator WAN, as shown in Figure 1. Extremely performance-sensitive NFs, such as RAN (Radio Access Network) NFs are hosted in operator far edges [37, 64], close to cell sites. Other NFs, such as the 5G Core, are hosted in the cloud, in its edges and/or DCs. For example, Nokia's 5G Core runs on the AWS cloud [61], Microsoft's 5G Core runs on both on-prem servers and Azure DCs [56], and its voicemail NF is hosted on Azure DCs [52]. Multiple operators have already moved parts of their 5G infrastructure

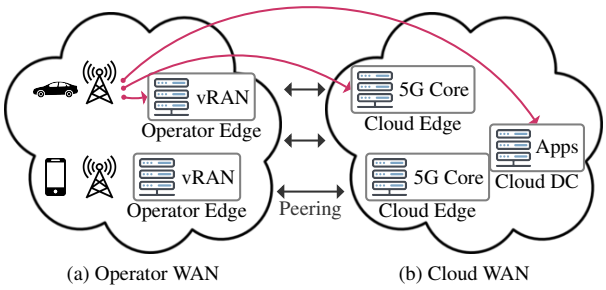


Figure 1: A simplified example of a 5G network

to the public cloud. AT&T is using Azure [8, 60]. Verizon is using AWS [12], leveraging multiple cloud technologies including Amazon EKS, AWS Wavelength, and HashiCorp Consul. The operator and cloud WANs peer [69] directly in multiple locations. An edge or DC can serve traffic flows from operator cell sites depending on where the destination NF or application is hosted [55]. Applications may include operator applications such as phone storage backup, private enterprise applications such as industrial AR/VR, and consumer applications such as game streaming.

Heterogenous network demands. 5G NFs have different performance requirements. For example, some NFs in the vRAN have stringent RTT, loss, and jitter requirements, while others such as UPF (User Plane Function) in the 5G Core need high throughput, and voicemail is performance-insensitive. Furthermore, upcoming releases such as 5G NR enable new applications, each with diverse performance requirements. For example, interactive AR/VR requires sub-20 ms network RTT for acceptable user experience [79, 91], beyond 4K video streaming needs 100+ Mbps throughput, and remote surgery demands high reliability. The 5G specification supports these diverse requirements through multiple radio-layer modes [65, 67, 79, 84], including: (1) *Ultra-reliable low latency communication (URLLC)* with ∼1 ms latency and 99.999% reliability, (2) *Enhanced mobile broadband (eMBB)* with multi-Gbps data rates, and (3) *Massive machine-type communication (mMTC)* for ultra-low energy communication at scales of up to 1 million nodes per KM^2.

Compute selection and routing decisions jointly matter. Placement of applications and NFs on compute that is *near* vs. *far* from the user [73], and the choice of network routes between users and this compute both impact performance. While operator edges are closer to users (few ms away) and hence are an ideal target in which to place highly responsive workloads (*e.g.,* vRAN), they have very limited compute due to physical space and power constraints at cellular base stations and mobile switching centers [74]. On the other extreme, the DC is furthest away (tens to hundred ms away) but has far more available compute.

Traditional selection techniques, such as those in CDNs (Content Delivery Networks), fall significantly short here. Directing all traffic from an entire subnet or LDNS (local

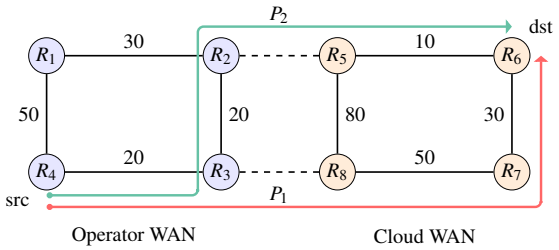


Figure 2: Sub-optimal routing for independent WANs.

DNS) coverage area to a single destination leads to undifferentiated performance for multiple flows with different requirements. Similar to DNS, the 5G NRF [33,80] (NF Repository Function) maintains profiles of available instances of NFs, and provides discovery and load balancing. However, it is not network-aware and only considers compute load. The problem is further exacerbated by the use of SFC (Service Function Chaining) [20] in 5G, where instances of different NFs are dynamically chained together to serve a request.

Picking a destination for a network demand based on compute capacity alone can lead to poor E2E performance. Picking a destination for a network demand based on network performance alone can lead to overloaded compute. Performance studies of existing 5G networks [91] have already found that "the untamed latency in the wireline paths, which is beyond mobile carriers' control, may neutralize 5G's latency advantage" and argue that "the wireline paths, upper-layer protocols, computing and radio hardware architecture need to co-evolve with 5G."

Need for multi-WAN coordination. Cloudification forces operators and clouds to share responsibility for tight performance guarantees of 5G workloads. Individual routing and compute placement decisions leads to sub-optimal performance. Figure 2 shows a latency sensitive flow traversing two WANs from source R_4 to destination R_6. Each link has an associated latency. The operator chooses the best egress point-of-presence (PoP) R_3 and path R_4R_3 to R_3 to minimize RTT on its network. The cloud routes traffic from R_8 to R_6 following the optimal RTT path, resulting in the E2E path P_1. However, a more efficient path P_2 exists. We show in §7 that decision making across both the cloud and operator unlocks such performance improvements.

Given the multitude of 5G networks across the world, multiple cloud providers, and dynamic nature of traffic and operational WANs, manual coordination to address performance problems as they appear does not scale.

New incentives for multi-WAN co-operation. Historically, it has been challenging to achieve collaborative routing beyond traditional peering agreements on the Internet. However, the cloudification of access has aligned the economic incentives of cloud and operator networks. The operator is not only selling access to end users to the cloud, it is now also a customer of the cloud for hosting 5G applications and NFs. There is a shared responsibility to manage compute and network between the two WANs for 5G demands.

3 OTTER

Prior empirical work [91] has shown that as 5G performance over-the-air has improved, WANs have become the bottleneck. We tackle this problem by placing heterogenous performance demands on the most appropriate compute and network paths across operator and cloud WANs. We refer to this as the multi-WAN flow placement problem, and it requires two core capabilities — the ability to direct a flow to a specific compute endpoint over a specific network path, and the ability to calculate that optimal path and endpoint.

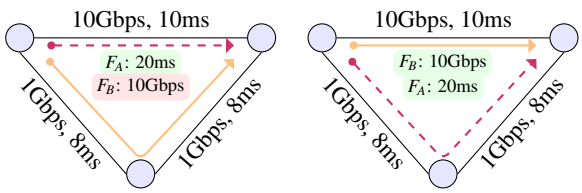
3.1 Design Goals

We design OTTER with the following goals:

Co-exist with non-5G WAN traffic. 5G traffic will share operational WANs that serve existing customers. For example, operator WANs already provide wireline connectivity to consumers or *eyeballs*. Cloud WANs carry enterprise and consumer workloads. Moreover, both types of WANs operate existing TE systems to engineer the capacity of their network efficiently [11,25,26,39,41,89]. However, both operator and cloud TE systems aim to achieve network welfare goals such as high utilization or bandwidth fairness [39,41], rather than optimizing for individual service objectives. Instead of overhauling existing TE systems in cloud and operator WANs, we design OTTER to be an overlay that operates in a layer of abstraction above existing WAN TE.

Dynamic flow placement. This overlay design choice further allows OTTER to place 5G flows on network paths and compute endpoints on-demand. In contrast, existing TE and other optimizations operate on batched traffic demands by periodically (*e.g.,* every 5 minutes) computing flow allocations from a predicted traffic matrix [39,41], or running a single optimizer instance for a single input demand [40]. A broker [39] enforces those periodic allocations by squelching flows that exceed them. While existing TE will continue to run on each WAN separately, OTTER can dynamically place flows without being subject to periodic allocations nor requiring bandwidth brokers in 5G deployments.

Support multiple, fine-grained service objectives. Today's WAN TE systems provide service differentiation using only a small number of priority classes (*e.g.,* high priority vs. low priority) [10, 39]. Such few classes are insufficient to express the more complex and fine-grained objectives of 5G traffic. These coarse-grained priority classes can fail to capture the demands of lower priority flows, even when there exists an allocation that maps all flows to acceptable paths. Consider a simple example in Figure 3, where each link can carry at most one flow. Flow F_A has high priority with a latency requirement of 20 ms, and flow F_B has low priority



(a) Preferential (b) Non-preferential

Figure 3: Two flow allocation strategies

with a throughput requirement of 10 Gbps. On the left, we show the allocation where flows are assigned in strict priority order, preferring shorter paths—this is how modern TE systems like OneWAN [45] and SWAN [39] operate. F_A will be assigned first to the top path while F_B will be assigned to the remaining bottom path and fall short of its throughput requirement. On the right, we show an alternate allocation that can map both flows to paths that meet their service demands. However, it is not trivial to find such an allocation at the scale of hundreds of endpoints and thousands of flows, each with their own performance demands across a variety of metrics.

Take inter-domain routing protocols as a given. Inter-WAN routing leads to sub-optimal E2E routes due to limited information disclosure and different goals between WANs [47, 48]. While prior work has shown that ASes can exchange information to produce E2E optimal routes, such negotiation entails heavyweight communication overheads that may not be practical [47, 48]. In the context of 5G, this overhead is further exacerbated by numerous traffic flows, each with widely varying objectives. So, we design OTTER without requiring changes to inter-domain routing protocols, policies, and implementations.

Not require private WAN data. While sharing intra-WAN topologies could lead to better multi-WAN paths, ASes have economic incentives to limit such visibility [47, 48, 75, 81]. 5G cloudification presents a unique opportunity for efficient cross-domain routing without sharing such private data. The operator, as a customer of the cloud, owns the workloads deployed on VMs in cloud edges, DCs, and operator edges. The cloud provides the control plane for managing bare metal at these locations, such as provisioning VMs and establishing connectivity. Thus both the operator and the cloud have visibility into network performance at an overlay layer between all compute locations. OTTER uses this visibility to make informed flow and compute placement. We show that the operator can deploy and run OTTER across both WANs, without access to private data or capabilities, and still provide significant performance improvements.

Multi-cloud support with open interfaces. 5G operators want to avoid cloud-lock in and deploy their NFs to one or more cloud providers of their choice. To achieve this, we design OTTER as a multi-cloud network overlay across operator and cloud WANs. OTTER uses publicly available VMs as nodes in the overlay, and a variety of public cloud NFs for

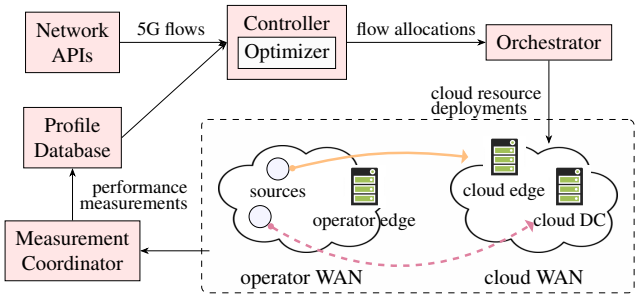


Figure 4: Overview of OTTER

routing between the nodes. This allows operators to use OTTER in multi-cloud deployments, without requiring private or special privileges from an underlying cloud.

3.2 OTTER Overview

OTTER places flows in a multi-WAN environment to satisfy their service demands. It does so by co-optimizing which 5G application / NF compute will serve the flow, and the route that flow will take to it. Figure 4 shows OTTER's architecture consisting of the Controller and Orchestrator. The Controller accepts flow demands to 5G NFs and applications using APIs [28, 34, 53, 57]. These "Quality on Demand" APIs [3] allow an application to create a session by issuing an HTTP POST to the */sessions* API, providing flow identifiers (IP addresses, port ranges, device phone number, etc.) and a QoS profile that best matches the application needs (see the "quality-on-demand v0.11.1" API on Swaggger [3]). Such standardized APIs have been implemented by 5G operators (see the implementations by Deutsche Telekom, Orange, and Spry Fox Networks posted on GitHub [3]) and by clouds that offer to host 5G services [15]. In addition to allowing application developers to specify their 5G flow needs, these APIs allow for termination of a session, extending a session, and getting notified when the requested QoS cannot be fulfilled. A UE that disconnects and reconnects, or incurs an inter-RAT (Radio Access Technology) handover, can create a new session with this API.

The Controller allocates compute to meet these performance requirements within the limits of compute availability on edges and DCs. The Controller calculates the most appropriate network paths to the allocated compute. Unlike TE systems, OTTER relies on available compute capacity at destination sites, as well as periodic measurements of network paths that it makes. This not only allows OTTER to route flows along alternate paths, but also to alternate destinations. The Orchestrator manages cloud resources and implements the forwarding mechanism to steer traffic along routes computed by the Controller. It provisions compute as per the Controller's optimal placement.

While similar challenges can be faced by other applications across WANs, OTTER is designed to serve 5G deployments. The business alignment between "eyeball" networks and cloud networks, where 5G deployments are hosted

across both, drives the necessary incentive to orchestrate both WANs. The diversity of 5G applications requires a design that supports multiple classes of traffic, not just bulk traffic [40]. End-user flows have to be placed in real time, not at traditional, periodic TE intervals. OTTER leverages APIs [28,34,53,57] already deployed by 5G providers to accept these application demands, where such APIs have not been deployed at scale for other applications nor networks. Finally, we design OTTER as an overlay without forcing one WAN to have administrative control nor require use of private capabilities – this design hence does not preclude a cloud from supporting multiple 5G operators, and a 5G operator from relying on multiple clouds.

4 OTTER Controller

The OTTER Controller maps traffic demands to network paths and compute resources. It does so while maximizing the allocated traffic and minimizing the degree to which 5G application-specific service requirements are violated. The resulting mapping of flows to destination compute sites and the network paths to reach them is an input to the OTTER Orchestrator. It informs the Orchestrator at which of the available edge sites or DCs to terminate 5G flows and which network paths to use. In this section, we formalize this problem, referred to as the *5G multi-WAN flow placement* problem, explain the constraints on the Optimizer, and describe techniques used to ensure its scalability.

The Optimizer takes as input the set of paths P that are available in the multi-WAN network. Each path $p \in P$ is represented by the tuple $(s,d,L,\sigma)_p$, where s_p is the source, d_p is the destination, L_p is the set of links used by the path, and σ_p is a vector of performance metrics for path p. σ_p^m refers to the value of metric m for path p. For example, the RTT, jitter, and packet loss of path p are represented by $\sigma_p^{rtt}, \sigma_p^{jit}, \sigma_p^{loss}$. L is the set of links in the network, and each link $l \in L$ has capacity c_l. Let P_l be the set of paths that share a link l, and P_d be the set of paths that share a destination d.

The Optimizer takes as input a set of 5G flows F that need to be allocated. Each flow $f \in F$ is represented by the tuple $(s,D,\mathcal{D},bw,r)_f$. s_f is the source of the demand. D_f is the set of destinations (specific edges & DCs) that can serve the flow. $\mathcal{D}_f$ represents the service demands of the flow with $\mathcal{D}_f^m$ giving the demand for metric m. bw_f is the requested bandwidth. r_f is a vector of destination resource requirements of the flow – compute, memory, and storage. We write r_f^π to refer to the requirements of flow f with respect to resource type π. Unlike TE, an OTTER flow is not limited to a single destination. Rather, we leverage the agility of the cloud to place compute resources for serving a 5G application or NF across multiple destinations. This allows OTTER to further optimize path performance by adjusting the destinations of flows. Compared to resource-constrained edge sites, a destination d that is a DC has $\approx \infty$ capacity c_d^π. For scalability, individual flows with the same source, destination set, service demands, and resource requirements are treated as part of a single "flow aggregate" for the Optimizer to allocate. The Optimizer solves the flow placement problem such that all flows within the same flow aggregate are routed in the same way and to the same destination.

Modeling service demands. We encode the service demand of a flow that is obtained from operator APIs (§ 3.2) as a vector of *demand functions* $\mathcal{D}_f$. A demand function $\mathcal{D}_f^m$ maps a flow's performance expectation, *i.e.,* what the flow expects to achieve for a given allocation, to a *tolerance coefficient* $\mathcal{D}_f^m(\sigma_p^m) \in (0,1]$. The tolerance coefficient is a measure of how good the path is for a flow. For example, an application may consider a path RTT under 100 ms acceptable, over 200 ms unacceptable, and linearly decrease its preference for a path between 100 and 200 ms. In this case, the Optimizer computes the tolerance coefficient to be 1 for all paths with an RTT under 100 ms, ε for all paths greater than 200 ms, and a value inversely proportional to the RTT for paths between 100 and 200 ms:

$$\mathcal{D}_f^{rtt}(\sigma_p^{rtt}) = \left\{ \begin{array}{ll} 1, & \text{if } \sigma_p^{rtt} \leq 100 \\ -\frac{1}{100}\sigma_p^{rtt} + 2, & \text{if } 100 < \sigma_p^{rtt} \leq 200 \\ \varepsilon, & \text{if } \sigma_p^{rtt} > 200 \end{array} \right\}$$

The Optimizer's goal is to allocate demands such that the degree of satisfaction (tolerance coefficients) is maximized. All tolerance coefficients are constant, allowing the objective function to be linear and the optimization problem to be a linear program (LP). The use of demand functions allows the Optimizer to scale over numerous path metrics and performance demands, as well as future-proof it for new applications that have arbitrary service demands.

Modeling resource requirements. Each flow specifies a resource requirement vector r_f. Each element represents compute in vCPUs, RAM, and disk storage needed to serve the 5G workload. While we focus on those three resources, the vector can be expanded and compared component-wise to the resource capacities of destination sites.

Decision variables. The Optimizer assigns network flows to paths in P and compute resources to destination sites in D. We have two decision variables: $x_{f,p}$, the amount of bandwidth of flow f assigned to path p, and $y_{f,d}^\pi$, the amount of resources of type π for flow f allocated at destination d. OTTER co-optimizes the flow allocation and the destination placement. See Table 1 for a summary of variables.

Objective function. The Optimizer aims to assign network flows to multi-WAN paths such that as much traffic is allocated as possible while still satisfying performance demands. The cost of a single flow assignment is the product of its allocated bytes and the overall tolerance coefficient summed over all performance metrics. To prevent bias towards large flows, we normalize this cost by the requested bandwidth of the flow bw_f. The objective function is as follows:

Inputs	
P	paths in network
s_p, d_p	source and destination of path p
L_p	links used by path p
σ_p^m	performance metric m for path p
L	links in network
c_l	capacity of link l
F	flows to allocate
s_f, D_f	source and destination set of flow f
$\mathcal{D}_f^m$	path performance demands for metric m of flow f
bw_f	requested bandwidth of flow f
r_f^π	resource requirements of flow f for resource π
c_d^π	resource capacities of destination d for resource π
Outputs	
$x_{f,p}$	amount of flow f assigned to path p
$y_{f,d}^\pi$	amount of resource π for flow f assigned to destination d

Table 1: Symbol table for the OTTER optimizer.

maximize	$\sum_{f\in F}\sum_{p\in P}\left(\frac{1}{bw_f}x_{f,p}\cdot\sum_m \mathcal{D}_f^m(\sigma_p^m)\right)$		
subject to			
	$x_{f,p} \geq 0$	$\forall f,p$	(c1)
$\sum_{p\in P}$	$x_{f,p} \leq bw_f$	$\forall f$	(c2)
$\sum_{f\in F}\sum_{p\in P_l}$	$x_{f,p} \leq c_l$	$\forall l$	(c3)
$\sum_{f\in F}$	$y_{f,d}^\pi \leq c_d^\pi$	$\forall d,\pi$	(c4)
$\sum_{d\in D_f}$	$y_{f,d}^\pi = r_f^\pi * \left(\sum_{p\in P_d} x_{f,p}\right)/bw_f$	$\forall f,\pi$	(c5)

Table 2: OTTER optimizer formulation.

$$\text{argmax}\sum_{f,p}\left(x_{f,p}\cdot\sum_{m\in\{rtt,jit,loss,bw\}}\mathcal{D}_f^m(\sigma_p^m)\right)/bw_f$$

We note that our objective function does not inherently prioritize one metric over another. Rather, applications can indicate priority through the demand functions.

Constraints. Table 2 summarizes the constraints in OTTER's optimization. Constraint $c1$ ensures that the total allocated flow is non-negative. $c2$ and $c3$ ensure that the allocated flows are bounded by their requested bandwidth as well as the bottleneck capacity of the paths. Destination sites have resource constraints that limit how many flows can map to them, and $c4$ prevents overload. Constraint $c5$ requires that the proportion of resources assigned to a flow at a destination is equal to the proportion of the amount of bandwidth allocated to the flow at the same destination.

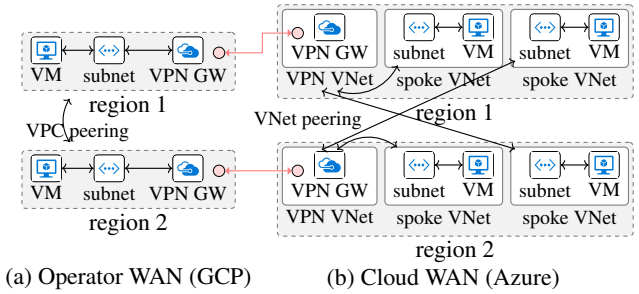
On-demand 5G flow allocation. Unlike TE systems which allocate flows periodically (*e.g.,* every 5 minutes) for predicted traffic demands, the OTTER Controller must perform on-demand flow placement. This difference arises due to the dynamic and unpredictable nature of 5G traffic [24] compared to typical WAN workloads. Unfortunately, re-running the Optimizer too frequently incurs a large delay. While the Optimizer computes newly optimal flow allocations, stale routes remain in-use, potentially leading to service demand violations [93]. To handle this, the Controller implements a greedy heuristic that approximates the optimal (*i.e.,* invoking the Optimizer on every new flow) configuration. It greedily allocates flows to available paths and destinations to provide the best demand satisfaction, while adhering to resource constraints. The Controller then periodically runs the full Optimizer in the background to optimally place new flows and those that were greedily assigned since the previous call to the Optimizer. The Controller can then shift those flows to achieve a globally optimal allocation, or it can pin flows to specific paths or destinations to avoid packet re-ordering. While shifting flows *could* result in an individual flow experiencing lower service demand satisfaction compared to its initial greedy allocation, reallocation still improves overall satisfaction across more flows, as shown in §7.

Computational complexity of OTTER. OTTER's optimizer is a linear program. Off-the-shelf optimization solvers implement polynomial time algorithms to solve linear programs, making it possible to solve OTTER's optimization quickly. While linear programs are quick to solve in practice, increasing the size of the problem can increase the run time of the solver. OTTER has two ways of dealing with high run times. First, OTTER reduces the set of possible destinations for each flow to a set of size n prior to solving the optimization. We estimate the n "best" destination sites using a flow's service demands ($\mathcal{D}_f^m$) to rank candidate destinations by preference. This effectively reduces the number of variables in the linear program. Second, depending on the size of the linear program, OTTER can run the full optimization at longer periodic intervals (*e.g.,* ten minutes) since the greedy flow allocation can assign flows to paths without waiting for the next interval. We evaluate the performance of greedy flow allocation between consecutive runs of the Optimizer, as well as the Optimizer, in detail in §7.

5 OTTER Orchestrator

The Orchestrator includes the forwarding mechanism necessary to steer traffic along routes computed by the Controller (§4). It also manages compute as per the Controller's optimal placement. We design the Orchestrator as a multi-WAN overlay on a cloud WAN and an operator WAN. To evaluate OTTER without impacting an operational 5G network, we substitute the operator WAN with GCP (Google Cloud Platform). Figure 5 shows one small deployment of OTTER, and Figure 19 in Appendix A shows a larger one. The Orchestrator uses only existing, native cloud functionality, which satisfies our design goal of not requiring private WAN data (§3).

Connectivity in the OTTER Orchestrator overlay. We host multiple VMs in each Azure region to represent 5G NF and application servers. VMs in GCP serve as 5G cell sites with far edges, while virtual private network (VPN) gateways behave like 5G near edge sites. We create private subnets in each GCP region and virtual private cloud (VPC) peer-



(a) Operator WAN (GCP) (b) Cloud WAN (Azure)

Figure 5: OTTER Orchestrator's multi-WAN topology.

ing connections in full-mesh between GCP regions. We use virtual network (VNet) peering between clusters of VMs in different regions to connect them to each other and to VPN gateways. Given that much of the 5G infrastructure, including cell sites and NFs, use private IP address ranges to protect communication, VPN tunnels are necessary to establish private connectivity among them. We have configured the VPN gateways to allow transit with the 5G operator WAN where both clouds have regional DCs and underlying peering (see Appendix A).

Forwarding in the OTTER Orchestrator overlay. The OTTER Orchestrator leverages native functionality in each cloud to achieve high forwarding performance, ease of scale, and lower deployment complexity. To control the path that a 5G flow takes in the overlay, we use a combination of route tables with user-defined routes (UDRs) and choice of multiple VNets in the destination region, something that both GCP and Azure support without requiring the customer to deploy their own BGP speakers and packet forwarders. Other design choices, such as using SNAT and DNAT with VPN gateways and firewalls [51], or MP-TCP on source and destination VMs, incur different tradeoffs in flexibility and overheads. We configure advanced parameters on VNet peering and network firewalls to allow VMs in one region to use a VPN gateway in another and allow transit traffic between regions. Each cloud component supports scale-up and scale-out, allowing the entire topology to easily cover the continental US, for example, without any code changes to individual components. We use symmetric paths for a flow but can easily use different paths in each direction. The use of one system to identify, manage, and use routes across WANs allows us to satisfy the need for multi-WAN coordination (§ 2).

Secure multi-WAN routing. We use private VNets to host 5G cell sites, edge sites, and NFs. We use VNet peering to allow only authorized traffic to transit between VNets. We use VPN gateways to enforce encryption when transiting WANs. Our design choices were inspired by product demonstrations from industry 5G NF providers [61].

Extensible multi-WAN topology. This design allows both scale and flexibility. Given an arbitrary source in a GCP region and a destination in an Azure region, it can use any VPN connection and any intermediate Azure and/or GCP region to send and receive traffic. We achieve this through a combination of VPN connections, VNet peering, and selective route announcements. We trivially increase inter-WAN forwarding throughput by scaling out VPN gateways and tunnels.

6 System Implementation

We have implemented the entire OTTER system shown in Figure 4 to operate on top of Azure and GCP, with no special support from either underlying cloud. The Orchestrator is built with HashiCorp's Terraform [86]. Our evaluation topology (available in an anonymous repository [1]), listed in Table 3 and described in §7.1, is built in ∼4,100 LoC of HCL (HashiCorp Configuration Language), with ∼2,600 LoC for module deployments, ∼600 LoC for GCP resource declarations, and ∼900 LoC for Azure resource declarations.

We implemented the Measurement Coordinator in ∼1,200 LoC in C#. It uses the Azure and GCP SDKs to automate periodic execution of measurement tools for path bandwidth, RTT, jitter, and loss. We use iPerf3 [23] to measure throughput, configured to use TCP CUBIC with 60 parallel connections, 2 seconds of warm up, and 5 seconds of measurement. For packet loss, we use iPerf3 in UDP mode. For RTT and jitter, we use sockperf [49]. We use round-trip values, and jitter is calculated as the standard deviation of RTT.

The Measurement Coordinator analyzes the output of the Orchestrator's Terraform execution to determine the deployment topology and the names and IP addresses of its components, including source and destination VMs. It then creates a non-overlapping schedule of measurements. The output of the measurement tools is parsed and converted to JSON. That JSON is then inserted as a new document record into an Azure Cosmos DB NoSQL database [54].

The Optimizer issues simple SQL queries on that database to obtain sliding window values of median throughput, RTT, jitter, and loss for all paths. 5G flow requirements are modeled from a subset of application profiles in the 3GPP standard [79], which we describe in §7.2. However, we expect to integrate with network APIs [28, 34] such as Azure Programmable Connectivity [53, 57] that allow 5G applications to specify flow QoS requirements. We have implemented the Optimizer using the Gurobi [35] solver. The Optimizer is an LP of ∼500 LoC. We run our implementation on a VM with a 16 core 3GHz CPU and 48 GB of memory.

7 Evaluation

We evaluate OTTER in two ways. First, in §7.1, we demonstrate the value of orchestrating paths across two WANs, compared to the default path from traditional Internet routing. Second, in §7.2, we evaluate how OTTER's optimization algorithm performs under a wide variety of situations at scale, and in comparison to alternative algorithms.

Cloud	Element	Deployment strategy	Total
GCP	regions	cover US	8
	VPC	1 per region	8
	subnet	1 per region	8
	cloud router	1 per region	8
	VPC peering	full mesh	56
	32 core VM [31]	1 per VPC	8
	classic VPN GW	4 per VPC	32
	cloud VPN tunnel	2 per VPN GW	64
Azure	regions	cover US	8
	VpnGw5 GW [14]	1 per region	8
	VPN GW interface	8 per VPN GW	64
	32 core VM [13]	8 per region	64
	VNet	1 per VM and VPN GW	72
	VNet peering	each VPN GW VNet to VM VNet	64

Table 3: Scale of OTTER topology across US regions.

7.1 OTTER Orchestrator Evaluation

§5 briefly describes how OTTER's Orchestrator works, with a secure, private, scalable design that allows flexible path selection to cater to multiple QoS requirements. We evaluate this orchestration at scale on two commercial cloud WANs.

We have scaled out the topology shown in Figure 5 to cover the US and support a large number of flows. Table 3 lists how many of the main components we use, though there are many other sub-components that we do not list, such as VM drives, NICs, custom routes, firewalls, and security rules. We use the biggest VPN GWs and multiple connections. We create VPN connections between the two clouds in nearby regions, listed in Appendix A. This topology is sized similarly to that of large 5G operators [21]. We use the largest VMs that have sufficient CPU and network bandwidth such that the VMs themselves are not bottlenecks in any of our measurements. We validate this through intra-cloud testing, where for example we achieve >25Gbps throughput between any two VMs in GCP in our topology, more than what is possible between two VMs across the two clouds.

We have run experiments on different weekdays and observed qualitatively similar results. For brevity, we show results from a 24 hour period starting on 09 Feb 2023 13:00PM (PT), where we repeatedly measure $8^3 = 512$ paths over 20 times. When comparing any two paths between the same source on GCP and the same destination on Azure, we only consider measurements immediately adjacent in time to each other. This ensures an "apples to apples" comparison, since comparing a path at 2am with one at 2pm is not useful for real-time demands.

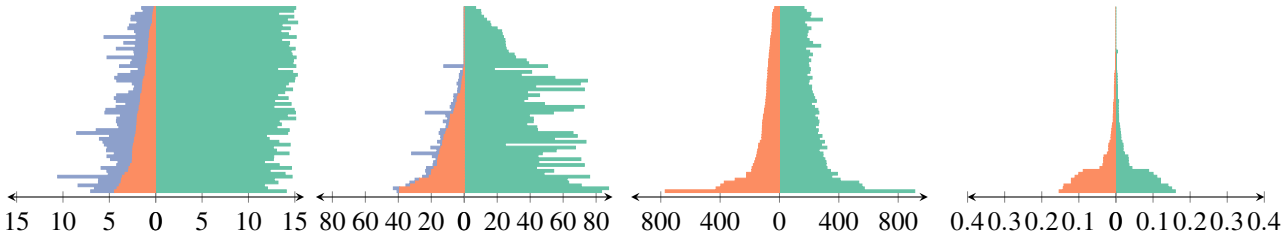
Figures 6a, 6b, 6c, and 6d show the network performance improvements that the OTTER Orchestrator is able to achieve. Each horizontal bar corresponds to one of 64 unique source-destination (Src-Dst) pairs, with the source in GCP and destination in Azure. The bar on the right represents the median performance observed for a Src-Dst pair when the OTTER Orchestrator is not invoked, *i.e.,* the default path [32] that a flow experiences when we do not specify which VPN connection to use and do not force traffic through an intermediate region (see Figure 5). The first bar on the left shows the median improvement that the OTTER Orchestrator achieves for the same Src-Dst pair when considering the best performing path across all possible combinations of VPN connections and intermediate regions. This median is computed as follows: for each round of measurements, we calculate the performance improvement of OTTER orchestration over the default path, and then use the median value across the ~20 rounds that happen over 24 hours. The second bar on the left shows the maximum improvement observed. The Src-Dst pairs in each figure are sorted by median improvement.

For example, the bottom bar in Figure 6a shows that this Src-Dst pair can achieve 14.1 Gbps, in the median across the 24 hour period, but the OTTER Orchestrator can improve that by 4.4 Gbps to achieve 18.5 Gbps in the median, and in the best case improves throughput to 21.1 Gbps. The bottom bar in Figure 6b shows a Src-Dst pair with 83.2 ms median RTT, but the OTTER Orchestrator can reduce that by 39.4 ms to achieve 43.8 ms in the median, and in the best to 42.4 ms.

Consider one Src-Dst pair example in Figure 7. From a source in GCP's US West 3 region to a destination in Azure's South Central US region, 13.6 Gbps throughput and 40.6 ms RTT is the median performance, on the typical [32] hot potato path, where traffic is handed off to Azure at the nearest location in Azure's West Central US region. OTTER has two other paths between the same source and destination that the Orchestrator can use with different RTT and throughput tradeoffs. When sending traffic via GCP's US South 1 region, the median RTT is significantly better at 35.0 ms, but at a slight degradation in throughput (13.5 Gbps). Instead, via GCP's US West 4 and Azure's West US 3 region, there is significantly higher throughput of 19.4 Gbps, but at slightly worse RTT of 42.8 ms. There is not one path with the best throughput, RTT, jitter and loss—different paths have different performance. OTTER is able to exploit the spectrum of paths to send traffic along the appropriate path for a flow, using scalable mechanisms involving VPN tunnels, VNet peering, and user-defined routes (§5).

Appendix B has additional presentations of these experimental results. We summarize our findings as:

- For flows prioritizing throughput, the OTTER Orchestrator offers higher throughput than the underlying networks. On average, it is 13% higher and in the best case 136% higher. In some cases, the throughput increases by 6-10 Gbps. The peak throughput OTTER achieves between two VMs across the clouds is >20 Gbps.
- For RTT, the average reduction is 15%, with a maximum of 56%. In some cases this is a RTT reduction of 42 ms, within the continental US.
- The average jitter reduction is 45%, with a maximum of 99%, which in some cases is a reduction of over 10 ms.
- Excluding some outliers, packet loss is reduced from 0.06% on average to less than 0.001%. In some cases



(a) Throughput (Gbps) (b) RTT (ms) (c) Jitter (μs) (d) Loss (%)

Figure 6: Improvements that the OTTER Orchestrator achieves per Src-Dst pair for each of 4 performance metrics. Bars on right show performance of default path, bars on left show median improvement and max improvement. Max improvements for jitter and loss are excluded for clarity.

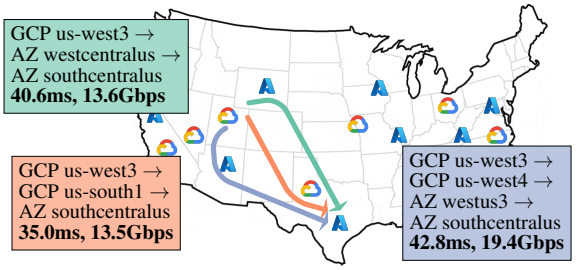


Figure 7: Examples of improvements in performance OTTER can achieve between the same Src-Dst pair.

the maximum reduction is more than 0.4%.

- Our scalable, secure, and private multi-WAN architecture achieves higher throughput than a simple architecture of VMs with public IPs. While some cloud components such as VPN tunnels can slightly worsen RTT and jitter, the OTTER Orchestrator compensates by finding better paths, providing the highest throughput, lowest RTT, jitter, or loss for any Src-Dst pair.

Prior work [40, 48, 91] has observed inefficiencies in the default intra- and inter-WAN paths provided by existing networks. Here, we comprehensively identify what OTTER can achieve across multiple objectives in the context of a deployment topology that is secure, private, and scalable, and achieves higher throughput than previously shown [40]. As seen in many TE algorithms [39, 41], the primary objective function is satisfying as much traffic demand as possible, with a possible secondary objective of minimizing RTT. This makes sense in a commercial WAN, where the cost of building and operating it has to be amortized over as much customer demand as possible. In contrast, OTTER's goal is to provide best performance to individual flows, and as such utilizes multiple paths that offer different QoS, without having to scale to the TE demands of the underlying networks. Further, OTTER utilizes virtual routing techniques (see §5), to react to real-time performance changes without requiring the underlying network infrastructure to issue route updates and converge, forcing all traffic to use those routes.

7.2 OTTER Controller Evaluation

Setup. We evaluate the Controller on the same continental US-wide, multi-WAN topology on which the Orchestrator is evaluated (§5). In Appendix F, we also evaluate synthetic smaller and larger topologies, including those that match the scale of contemporary TE systems, and show qualitatively similar results. To avoid limiting findings to specific network performance values, we evaluate the Controller on a distribution of values listed in Table 4. Path attributes are sampled at random from the distribution of measured path metrics in our multi-WAN topology, while destination and flow attributes are chosen uniformly at random from the specified range to mimic 5G traffic [79].

Flows arrive according to a Poisson process with a mean arrival rate. We evaluate multiple flow arrival rates as listed in Table 4. We use 10 seconds as the mean flow duration, and we show qualitatively similar results for varying durations in Appendix C. We randomly assign an application profile to each flow and generate a demand function for each performance metric using one of three function types: step, linear, or rational, with inflection points in each function derived from the guideline service demands in Table 5. Those demands are based on 5G application profiles from the relevant 3GPP standard [79], described in more detail in Appendix D. To account for flow diversity, we select inflection points at random from within $\pm$25% of the guideline metric. We also evaluate the effect of assigning application profiles from varying distributions based on RTT requirements, instead of random assignments, and show qualitatively similar results in Appendix E.

Heuristics. We implement several heuristics in the Controller, with different levels of optimality, compute needs, and impact on ongoing flows:

- GREEDY: Flows are allocated greedily in arrival order to the best paths based on the tolerance coefficient.
- PER: Same as GREEDY, but the Optimizer also periodically runs to shift all active flows to optimal paths. While the Optimizer is computing allocations, incoming flows are allocated greedily.

Topology			value/range		units
Num sources				8	#
Num destinations (DC)				2	#
Num destinations (edge)				6	#
Num paths per src/dst pair				8	#
Path Attributes	min	max	avg	std	units
Throughput	4.0	20.8	13.7	1.6	Gbps
RTT (DC)	69.2	150.8	92.2	18.2	ms
RTT (edge)	3.2	34.6	24.8	7.2	ms
Jitter	0.008	38.0	0.62	1.38	ms
Loss	0.0	4.3	0.06	0.16	%
Destination Attributes			value/range		units
CPU capacity (edge)			128-4096		cores
Memory capacity (edge)			128-4096		GB
Storage capacity (edge)			1024-32768		GB
Flow Attributes			value/range		units
Flow arrival rate			20K-40K		flows/s
Flow duration mean			10		s
Flow duration std			10		s
Application profile			1-8		index
Demand function			{step, linear, rational}		type
CPU requirement			1-128		#
Memory requirement			1-128		GB
Storage requirement			8-1024		GB

Table 4: Evaluation Parameters. Values are sampled at random from the specified ranges. Path attribute values are actual measurements from our multi-WAN topology.

Application name	**Throughput** (Mbps)	**Latency** (ms)	**Jitter** (ms)	**Loss** (rate)
Conversational Voice	0.3	100	-	10^{-2}
Live Streaming Video	1.5	150	-	10^{-3}
Buffered Streaming Video	15	300	-	10^{-6}
MC-PTT Voice	1	75	-	10^{-2}
Augmented Reality	20	10	-	10^{-6}
Real-time Gaming	10	50	10	10^{-3}
Remote Driving	1	5	-	10^{-4}
Intelligent Transport	1	30	-	10^{-5}

Table 5: Application profiles for flows (see Appendix D for details).

- PER+PATHPIN: Same as PER, except RTT and jitter-sensitive flows are pinned to their destinations and paths and not shifted even if the Optimizer finds a better path.
- PER+DSTPIN: Same as PER, except RTT and jitter-sensitive flows can be assigned to a new path without changing the destination.
- OPT+PATHPIN: An unattainable version of PER+PATHPIN that uses an infinitely fast optimizer. This removes the need to greedily allocate during optimizer compute time.
- OPT+DSTPIN: An unattainable version of PER+DSTPIN that uses an infinitely fast optimizer.
- OPT: An unattainable version of PER that uses an infinitely fast optimizer, which invokes a re-optimization of the network on every flow arrival and can re-allocate flows to optimal paths with no restrictions.

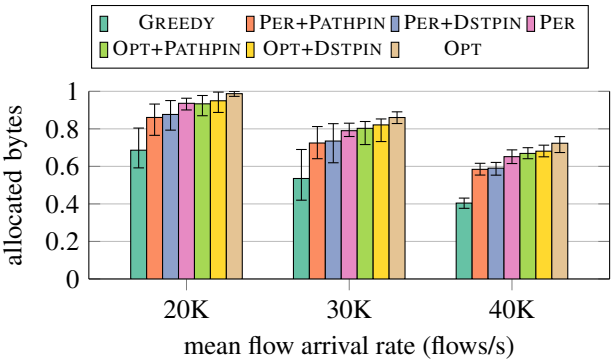


Figure 8: Average fraction of allocated bytes across all flows. Error bars show min-max over multiple runs.

Traffic allocation. Figure 8 shows the average fraction of bytes allocated across all flows for each allocation method for varying flow arrival rates. We define allocated bytes for a flow as the amount of requested bandwidth bw_f successfully allocated multiplied by its duration. We see that OTTER with periodic re-optimizing (PER) always allocates more traffic compared to the baseline GREEDY method, while still reacting quickly to on-demand flows. As expected, as the flow arrival rate increases, the number of allocated bytes decreases across all allocation methods. This is explained by the increase in resource contention when more active flows are present on networks of the same size with similar link and destination capacities. We also see that PER+PATHPIN and PER+DSTPIN on average allocate 26–45% more bytes than GREEDY, and just 10% fewer bytes than their unattainable optimal counterparts. In Appendix F, we scale to mean flow arrival rates of up to 512K flows/s without observing a large drop-off in gap-to-optimal performance.

For intuition into how the OTTER Controller allocates flows, Figure 9 has a time series of the average fraction of requested bandwidth that is successfully allocated by each method. It shows results for a mean flow arrival rate of 40K flows per second. Note that for instantaneous bandwidth allocation, we do not weight the allocation by the size of the flow, and only show the unweighted average across all flows at each point in time. OTTER's periodic re-optimizing methods exhibit a sawtooth behavior, with a jump in allocated bandwidth after each optimizer run and a steady decrease during the solve time due to falling back to GREEDY allocation for new flows. PER+PATHPIN and PER+DSTPIN provide a less optimal solution after each Optimizer run due to pinning, as shown by smaller sawtooths, but can run more frequently since the additional constraints reduce solve time.

These results also show how OTTER's Optimizer better maps flows to suitable paths without starving flows that may be considered "lower priority" in other schemes. While GREEDY also uses our notion of tolerance when ranking paths, it will greedily assign a flow to the best available path

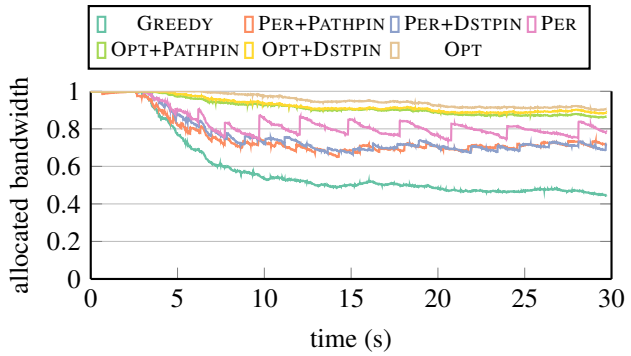


Figure 9: Average fraction of allocated bandwidth across all flows over time.

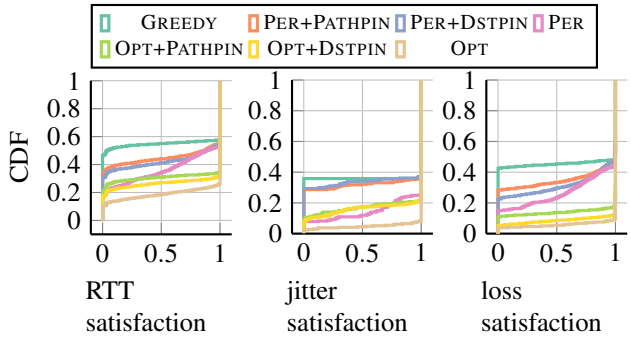


Figure 10: Service demand satisfaction for RTT, jitter, and packet loss. Value of 1 means perfect satisfaction.

even if there are alternate paths with worse performance but still just as satisfactory. The other allocation methods spread flows across available paths instead of assigning them to a single, attractive path, and those details are in Appendix G.

Service demand satisfaction. We now show that OTTER meets the design goal of placing fine-grained flow service demands. We choose a mean flow arrival rate of 40K flows/sec, which is the most difficult workload we evaluate, to show a lower bound on service demand satisfaction. All other parameters remain as in Table 4. For each flow, we compute its average "satisfaction" for each performance metric over its assigned paths, defined as the tolerance coefficient normalized to the fraction of bytes successfully allocated.

Figure 10 shows flow satisfaction distribution for RTT, jitter, and packet loss. Throughput satisfaction is the fraction of bytes successfully allocated, previously shown in Figure 8. Across all three service metrics, OTTER's periodic re-optimizing methods map a similar fraction of flows to paths with perfect satisfaction compared to GREEDY, but allocate far fewer flows to paths with extremely poor (<0.01) satisfaction. For RTT, PER+PATHPIN and PER+DSTPIN map approximately 47% of flows to paths with perfect RTT satisfaction, while GREEDY does so for 41% of flows. PER+PATHPIN and PER+DSTPIN allocate significantly fewer flows to paths that are entirely unsatisfactory (35% vs. 44%). Note that even with optimal allocation, OTTER still assigns at least 10% of flows to paths with poor RTT satisfaction (<0.01). Most of these flows belong to application profiles with service demands that are extremely difficult to meet alongside a large number of other flows.

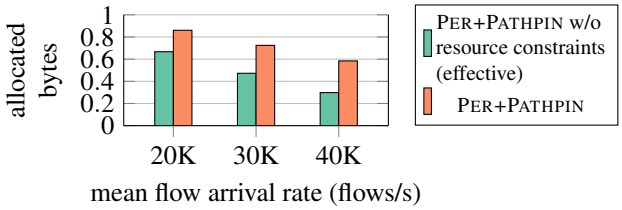


Figure 11: Impact of resource constraints on allocated bytes.

Best heuristic. Of the four realistic heuristics, we choose PER+PATHPIN for OTTER. It limits disruption of extremely performance sensitive flows, while still offering high allocated bytes and satisfaction.

Impact of edge resource constraints. In the disaggregated 5G network architecture, edge computing plays a central role in meeting service demands. However, TE systems do not account for the heterogenous availability of compute at edge sites, and may over-subscribe flows to destinations.

To demonstrate the value of explicitly considering destination resource capacities when routing 5G flows, we modify OTTER's Optimizer to ignore destination resource constraints (eliminating constraint $c4$ in Table 2). This effectively gives all destinations infinite resource capacity. The LP is restricted only by link capacities. While the unconstrained LP is seemingly able to allocate more bytes, the actual amount is much lower due to resource contention at destination sites. To estimate the amount of traffic that can actually be serviced, we reduce allocated bytes from flows that are assigned to over-subscribed destination sites. We follow random priority when de-allocating, *i.e.,* for each destination, we repeatedly select a flow at random and decrease the number of bytes allocated until the resource limitations are met. Figure 11 compares the effective fraction of bytes allocated between OTTER using PER+PATHPIN with and without edge resource constraints for different traffic loads. As we can see, ignoring destination resource constraints in the Optimizer formulation results in 23–50% fewer bytes allocated than PER+PATHPIN. This shows that the added complexity of integrating resource capacity constraints allows OTTER to better map flows to destinations that have the capacity to serve them, allowing more traffic to be supported.

8 Miscellany

Why does BGP not get in the way? Cellular network operators tend to be large ISPs that directly peer with public clouds in multiple locations. Our evaluation setup in §7 represents that, with both WANs directly peering and our use

of VPN gateways at the overlay layer. Not only do we expect 5G traffic to flow over direct connections, we expect 5G cloudification to spur operators to increase the quantity of peering. This has two implications for OTTER. First, many of the oddities of BGP in the wild do not apply here, where intermediate ASes enforce their policies between source and destination ASes. Second, we expect to see even more diversity of inter-domain paths that OTTER can exploit.

Why not optimize for WAN costs? OTTER does not explicitly incorporate monetary cost as an optimization criteria. Cloud compute and network costs tend to be uniform within countries. Most 5G network deployments are at national scales. Operators and clouds that they depend on use direct peering. Hence, optimizing for WAN costs is not a priority for OTTER and we leave that for future work.

Will OTTER negatitvely impact existing WAN TE? OTTER works within the confines of TE in both WANs — the routes available to OTTER are what the underlying TE has exposed to applications. OTTER then can only influence the traffic matrix, which TE systems effectively optimize over. If OTTER-induced demands are sufficiently high, TE systems can change the routes available to applications, including OTTER. OTTER will in turn adapt to these new routes and still pick the best ones for 5G workloads. While there is a feedback loop between OTTER and TE, it is not one that can be exploited for reducing the efficiency of existing TE. In Appendix H, we explore the value of using paths that underlying TE systems create but do not expose to applications.

9 Related Work

Overlay networks. Researchers have proposed overlay networks to improve underlying network availability, performance, or security [4, 5, 17, 22, 36, 59, 70, 77]. Recently, Skyplane [40] proposed a cloud overlay for inter-cloud bulk transfers between object stores, and selects routes that balance price and throughput. Routes and VMs are chosen using static throughput and cost measurements and participate in store-and-forward queuing. Hercules [29] is another bulk-transfer tool for overlays. CRONets [18] deploys overlay nodes on cloud networks and leverages MPTCP through overlay paths to improve throughput. CloudCast [72] focuses on latency improvement using cloud overlays.

OTTER also uses an overlay network to interconnect a 5G operator WAN and a cloud WAN. Unlike prior work, OTTER optimizes for various 5G NFs and applications, each of which can have different performance requirements, not just bulk data transfer. OTTER optimizes network paths online by using dynamic measurements for a wide range of metrics, including RTT, jitter, and loss. We achieve higher inter-cloud, per-VM throughput than prior work [40], in part because our security and privacy oriented topology design does not suffer from limits that some clouds impose on bandwidth to public IPs (Appendix B). OTTER also allocates both the path of flows as well as the compute destination.

Cloud TE. Existing cloud TE systems (such as Taiji [19], COPE [87], SWAN [39], B4 [41], BwE [46], OneWAN [45], Cascara [76], Espresso [94]) maximize network utilization or bandwidth fairness, or minimize cost for traffic grouped by coarse-grained priority classes. OTTER is not a replacement for existing TE, as underlay paths will still be subject existing TE. OTTER is an enhancement that can optimize at the granularity required for 5G applications by selecting overlay paths between multiple destinations across WANs. Additionally, OTTER performs dynamic flow placement and considers server resource capacities at endpoints, while TE periodically allocates batched traffic predictions and only considers network resource constraints. While some recent TE work [50] to optimize more fine-graned flows for applications, this optimization still takes place within a single WAN, and does not consider the multi-WAN setting of OTTER.

Multi-WAN collaboration. Unlike TE systems, OTTER optimizes traffic that spans two WANs. Prior work (Nexit [48], Wiser [47], P4P [90], Flow Director [68]) attempt to facilitate coordination between two independent entities by allowing them to exchange information while preserving privacy. Nexit and Wiser allow two ISPs to route traffic efficiently, despite competing interests. Nexit uses a negotiation-based approach, while downstream ISPs using Wiser tag routing advertisements with costs used by upstream ISPs. P4P improves peer-to-peer application performance by allowing ISPs to share performance and topology information with applications. Flow Director uses ALTO [2] to share information between ISPs and CDNs to optimize content placement. However, these works require negotiation to ensure mutually beneficial routing. OTTER makes use of the fact that, for 5G networks, these incentives are already aligned – operators pay to deploy resources on cloud networks to facilitate better QoS for 5G flows. OTTER does not require any buy-in from cloud platforms, so routing and resource placement can be done without coordination.

Better routing. Attempts to improve interdomain routing include multipath extensions to BGP [38, 92] and interdomain bandwidth reservation such as N-Tube [88]. New routing protocols that allow nodes to select paths that optimize multiple metrics include Routing on Multiple Optimality Criteria [78]. Such approaches may improve path diversity or even path selection. However, they must still optimize routing based on static metrics such as BGP AS path length or width, and cannot adjust based on dynamic metrics such as loss or latency. Deploying any of these technologies requires changes to BGP, which has proven to be a major practical impediment. SCION [95] is a novel Internet routing architecture that seeks to provide high security and availability. However, it requires ISP buy-in to deploy border routers to participate. RouteScout [7], PAINTER [43] and SCULP-

TOR [6] reroute traffic based on traffic performance degradation. However, such optimizations are performed locally or per prefix. In contrast, OTTER optimizes the entire overlay according to the performance demands of all flows, allocates per flow, and does not require deploying new devices.

Better path selection at the edge. In Tango [16], edge networks coordinate to expose more paths, enable better measurements, and route traffic efficiently. Rather than coordinate paths between edge networks, OTTER directly sets up overlay paths between the cloud provider and 5G operators by orchestrating resources on the cloud network. OTTER also routes traffic according to application-specific service demands and relies only on existing cloud network technologies. Other work [42] selects paths between a private WAN and the public Internet for video conferencing applications (e.g., Teams) based on extensive measurement data. Like OTTER, this work jointly optimizes the path and destination DC selection. However, unlike OTTER, it neither considers the general setting of 5G applications nor the coordination of paths among multiple ASNs. Any of these mechanisms that expose multiple paths to an endpoint could be utilized at the transport layer by multipath TCP.

Virtual Network Function (VNF) placement. Systems for VNF placement [62, 63, 82] optimize for metrics such as cost and CPU utilization of NFs. However, unlike OTTER, they do not consider the QoS needs of individual flows, and are evaluated on simulated setups. VNF placement systems periodically optimize for coarse-grained objectives, according to predicted future load, but OTTER dynamically places flows and resources, on-demand, according to customizable fine-grained service objectives.

10 Conclusions

The next generation of 5G networks includes new frequency bands and radio technology, equipping users with access links so fast that their E2E performance will become bottlenecked by inter-domain paths [91] rather than last mile access links. Moreover, next generation 5G networks will rely on hyperscalers to deploy software-based 5G NFs [61] and interactive applications. Thus, we need to improve the performance of paths across operator and cloud WANs.

We demonstrate the value of constructing such paths over two commercial WANs. OTTER achieves 13% higher average throughput or 6-10 Gbps in the maximum, RTT reduction as much as 42 ms within the US, jitter reduction by 45% on average, and 0.06% lower packet loss. We designed and implemented a scalable algorithm to jointly optimize the placement of workloads on available paths and compute resources. It allocates 26%–45% more bytes on the network than good baselines while also satisfying the service demands of more flows. While we focus on the urgency of 5G, our research contributions may be relevant in other upcoming access technologies such as 6G.

OTTER unlocks avenues of future work. Underlay paths that are private to one WAN can provide even more diversity in end-to-end path performance than we have shown – how can they be leveraged? How will OTTER work in topologies where a 5G operator relies on two or more hyperscalers?

References

[1] 3GPP. Service requirements for enhanced V2X scenarios. https://www.etsi.org/deliver/etsi_ts/122100_122199/122186/16.02.00_60/ts_122186v160200p.pdf, 2020.

[2] R. Alimi, R. Penno, Y. Yang, S. Kiesel, S. Previdi, W. Roome, S. Shalunov, and R. Woundy. Application-Layer Traffic Optimization (ALTO) Protocol. RFC 7285, September 2014.

[3] The Telco Global API Alliance. Quality of Demand (QoD) in CAMARA. https://camaraproject.org/quality-on-demand/, 2025.

[4] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. *SIGOPS Oper. Syst. Rev.*, 35(5), Oct 2001.

[5] David G. Andersen, Alex C. Snoeren, and Hari Balakrishnan. Best-Path vs. Multi-Path Overlay Routing. In *Internet Measurement Conference*, IMC '03, New York, NY, USA, 2003. ACM.

[6] anonymous. SCULPTOR Has Your Back(up Path): Carving Interdomain Routes to Services. In *under submission to USENIX NSDI 2025*.

[7] Maria Apostolaki, Ankit Singla, and Laurent Vanbever. Performance-Driven Internet Path Selection. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, SOSR '21, New York, NY, USA, 2021. Association for Computing Machinery.

[8] AT&T. AT&T Moves 5G Mobile Network to Microsoft Cloud. https://about.att.com/story/2021/att_microsoft_azure.html, 2021.

[9] ATT, BT, CenturyLink, China Mobile, Colt, Deutsche Telekom, KDDI, NTT, Orange, Telecom Italia, Telefonica, Telstra, and Verizon. Network Functions Virtualisation (NFV) White Paper. In *SDN and OpenFlow World Congress*. ETSI https://portal.etsi.org/nfv/nfv_white_paper.pdf, October 2012.

[10] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. In *RFC 2702*, 1999.

[11] D. O. Awduche. MPLS and traffic engineering in IP networks. *IEEE Communications Magazine*, 37(12), 1999.

[12] AWS and Verizon. Architecting Geo-Distributed Mobile Edge Application With Consul. https://www.youtube.com/watch?app=desktop&v=At6cHrL6s0Q, 2022.

[13] Azure. Azure Ev3 and Esv3-series VMs. https://learn.microsoft.com/en-us/azure/virtual-machines/ev3-esv3-series, 2024.

[14] Azure. Azure VPN Gateway SKUs. https://learn.microsoft.com/en-us/azure/vpn-gateway/about-gateway-skus, 2024.

[15] Microsoft Azure. What is Azure Programmable Connectivity? https://learn.microsoft.com/en-us/azure/programmable-connectivity/azure-programmable-connectivity-overview, 2025.

[16] Henry Birge-Lee, Maria Apostolaki, and Jennifer Rexford. It Takes Two to Tango: Cooperative Edge-to-Edge Routing. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, HotNets '22, New York, NY, USA, 2022. ACM.

[17] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat. Opus: an overlay peer utility service. In *IEEE Open Architectures and Network Programming Proceedings. OPENARCH 2002*, 2002.

[18] Chris X. Cai, Franck Le, Xin Sun, Geoffrey G. Xie, Hani Jamjoom, and Roy H. Campbell. Cronets: Cloud-routed overlay networks. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, 2016.

[19] David Chou, Tianyin Xu, Kaushik Veeraraghavan, Andrew Newell, Sonia Margulis, Lin Xiao, Pol Mauri Ruiz, Justin Meza, Kiryong Ha, Shruti Padmanabha, Kevin Cole, and Dmitri Perelman. Taiji: Managing Global User Traffic for Large-Scale Internet Services at the Edge. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, New York, NY, USA, 2019. ACM.

[20] Cisco. The Cisco 5G Strategy Series: Packet Core, Transport, and Identity Management. https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/ultra-services-platform/5g-strategy-series.pdf, 2016.

[21] Cloud and Colocation. AT&T Data Center Locations. https://cloudandcolocation.com/colocation-provider/att/, 2024.

[22] Andy Collins. The detour framework for packet rerouting. 1998.

[23] Jon Dugan, Seth Elliott, Bruce A. Mah, Jeff Poskanzer, and Kaustubh Prabhu. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. https://iperf.fr, 2024.

[24] Ericsson. Traffic patterns and network design: To achieve optimum 5G performance, both coverage and capacity must be available throughout the network according to location-specific needs. https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/traffic-patterns-drive-network-evolution, 2024.

[25] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40(10), 2002.

[26] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000*, volume 2, 2000.

[27] Nate Foster, Nick McKeown, Jennifer Rexford, Guru Parulkar, Larry Peterson, and Oguz Sunay. Using Deep Programmability to Put Network Owners in Control. In *SIGCOMM CCR*, 2020.

[28] The Linux Foundation. CAMARA: The Telco Global API Alliance. https://camaraproject.org/, 2024.

[29] Marten Gartner, Jean-Pierre Smith, Matthias Frei, Francois Wirz, Cédric Neukom, David Hausheer, and Adrian Perrig. Hercules: High-Speed Bulk-Transfer over SCION. In *IFIP Networking*, 2023.

[30] Google. Cross-Cloud Interconnect overview. https://cloud.google.com/network-connectivity/docs/interconnect/concepts/cci-overview, 2024.

[31] Google. Google n2-standard-32 VMs. https://gcloud-compute.com/n2-standard-32.html, 2024.

[32] Google. Routing in Google Cloud. https://cloud.google.com/vpc/docs/routes, 2024.

[33] GooRAM. 5G: What's the NRF? https://gooram.medium.com/5g-whats-the-nrf-7db7a83cb1bd, 2023.

[34] GSMA. Open Gateway API Descriptions. https://www.gsma.com/futurenetworks/gsma-open-gateway-api-descriptions/, 2024.

[35] Gurobi Optimization. Gurobi optimizer reference manual. http://www.gurobi.com, 2019.

[36] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 4, 2005.

[37] Red Hat. Automatic & Zero-Touch Provisioning for the RAN. https://www.youtube.com/watch?v=2m1CVD5ytgI, 2023.

[38] Jiayue He and Jennifer Rexford. Toward internet-wide multipath routing. *IEEE Network*, 22(2), 2008.

[39] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM*, 2013.

[40] Paras Jain, Sam Kumar, Sarah Wooders, Shishir G. Patil, Joseph E. Gonzalez, and Ion Stoica. Skyplane: Navigating the Cost and Performance Trade-off for Cloud Bulk Transfers. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023.

[41] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a Globally-Deployed Software Defined WAN. In *ACM SIGCOMM*, 2013.

[42] Bhaskar Kataria, Palak LNU, Rahul Bothra, Rohan Gandhi, Debopam Bhattacherjee, Venkata N Padmanabhan, Irena Atov, Sriraam Ramakrishnan, Somesh Chaturmohta, Chakri Kotipalli, et al. Saving private wan: Using internet paths to offload wan traffic in conferencing services. *arXiv preprint arXiv:2407.02037*, 2024.

[43] Thomas Koch, Shuyue Yu, Sharad Agarwal, Ethan Katz-Bassett, and Ryan Beckett. PAINTER: Ingress Traffic Engineering and Routing for Enterprise Cloud Networks. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23. Association for Computing Machinery, 2023.

[44] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. Decentralized cloud wide-area network traffic engineering with BlastShield. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021.

[45] Umesh Krishnaswamy, Rachee Singh, Paul Mattes, Paul-Andre C Bissonette, Nikolaj Bjorner, Zahira Nasrin, Sonal Kothari, Prabhakar Reddy, John Abeln, Srikanth Kandula, Himanshu Raj, Luis Irun-Briz, Jamie Gaudette, and Erica Lan. OneWAN is better than two: Unifying a split WAN architecture. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023.

[46] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. *SIGCOMM Comput. Commun. Rev.*, 45(4), Aug 2015.

[47] Ratul Mahajan and David Wetherall. Mutually Controlled Routing with Independent ISPs . In *4th USENIX Symposium on Networked Systems Design and Implementation (NSDI 07)*, Cambridge, MA, apr 2007. USENIX Association.

[48] Ratul Mahajan, David Wetherall, and Thomas Anderson. Negotiation-Based Routing between Neighboring ISPs. In *2nd USENIX Symposium on Networked Systems Design and Implementation*, NSDI'05, USA, 2005. USENIX Association.

[49] Mellanox. sockperf: Network Benchmarking Utility. https://github.com/Mellanox/sockperf, 2024.

[50] Congcong Miao, Zhizhen Zhong, Yunming Xiao, Feng Yang, Senkuo Zhang, Yinan Jiang, Zizhuo Bai, Chaodong Lu, Jingyi Geng, Zekun He, et al. Megate: Extending wan traffic engineering to millions of endpoints in virtualized cloud. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 103–116, 2024.

[51] Microsoft. About NAT on Azure VPN Gateway. https://learn.microsoft.com/en-us/azure/vpn-gateway/nat-overview, 2023.

[52] Microsoft. Azure Operator Voicemail: Take the first step to move voice workloads to the cloud. https://techcommunity.microsoft.com/t5/azure-for-operators-blog/azure-operator-voicemail-take-the-first-step-to-move-voice/ba-p/3751315, 2023.

[53] Microsoft. Azure Programmable Connectivity. https://techcommunity.microsoft.com/t5/azure-for-operators-blog/azure-programmable-connectivity-powers-network-aware/ba-p/3751024, 2023.

[54] Microsoft. Azure Cosmos DB. https://azure.microsoft.com/en-us/products/cosmos-db, 2024.

[55] Microsoft. Azure Operator Nexus. https://azure.microsoft.com/en-us/products/operator-nexus/, 2024.

[56] Microsoft. Microsoft Azure Private 5G Core. https://azure.microsoft.com/en-us/products/private-5g-core, 2024.

[57] Microsoft. Write network-aware applications with Azure Programmable Connectivity. https://build.microsoft.com/en-US/sessions/2c7f3bc6-e021-4f5c-8749-a2544f34395d, 2024.

[58] Netflix. Internet connection speed recommendations. https://help.netflix.com/en/node/306, 2023.

[59] NetFoundry. OpenZiti. https://docs.openziti.io, 2024.

[60] PR Newswire. AT&T to run its mobility network on Microsoft's Azure for Operators cloud, delivering cost-efficient 5G services at scale. https://www.prnewswire.com/news-releases/att-to-run-its-mobility-network-on-microsofts-azure-for-operators-cloud-delivering-cost-efficient-5g-services-at-scale-301323256.html, 2021.

[61] Nokia. Nokia 5G Core Software as a Service in practice. https://www.youtube.com/watch?v=_4DHzjrtB34, 2023.

[62] Jianing Pei, Peilin Hong, Miao Pan, Jiangqing Liu, and Jingsong Zhou. Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks. *IEEE Journal on Selected Areas in Communications*, 38(2):263–278, 2020.

[63] Jianing Pei, Peilin Hong, Kaiping Xue, and Defang Li. Efficiently Embedding Service Function Chains with Dynamic Virtual Network Function Placement in Geo-Distributed Cloud System. *IEEE Transactions on Parallel and Distributed Systems*, 30(10):2179–2192, 2019.

[64] Larry Peterson, Tom Anderson, Sachin Katti, Nick McKeown, Guru Parulkar, Jennifer Rexford, Mahadev Satyanarayanan, Oguz Sunay, and Amin Vahdat. Democratizing the Network Edge. In *ACM SIGCOMM Computer Communication Review*, 2019.

[65] Larry Peterson and Oguz Sunay. 5G Mobile Networks: A Systems Approach. 2022.

[66] Phone.com. How Much Bandwidth Do I Need for VOIP? https://www.phone.com/much-bandwidth-need-voip/, 2014.

[67] Petar Popovski, Kasper F. Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. 5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View. In *IEEE Access*, 2018.

[68] Enric Pujol, Ingmar Poese, Johannes Zerwas, Georgios Smaragdakis, and Anja Feldmann. Steering Hyper-Giants' Traffic at Scale. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, CoNEXT '19, New York, NY, USA, 2019. Association for Computing Machinery.

[69] Yakov Rekhter, Susan Hares, and Tony Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.

[70] Adolfo Rodriguez, Charles Killian, Sooraj Bhat, Dejan Kostić, and Amin Vahdat. MACEDON: Methodology for Automatically Creating, Evaluating, and Designing Overlay Networks. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, USA, 2004. USENIX Association.

[71] RootMetrics. Mobile cloud gaming: the real-world cloud gaming experience in Los Angeles. https://rootmetrics.com/en-US/content/us-LA-gaming-report-2020, 2020.

[72] Noga H. Rotman, Yaniv Ben-Itzhak, Aran Bergman, Israel Cidon, Igor Golikov, Alex Markuze, and Eyal Zohar. CloudCast: Characterizing Public Clouds Connectivity. *CoRR*, abs/2201.06989, 2022.

[73] Mahadev Satyanarayanan. The Emergence of Cloud Computing. In *IEEE Computer*, 2017.

[74] Mahadev Satyanarayanan, Guenter Klas, Marco Silva, and Simone Mangiante. The Seminal Role of Edge-Native Applications. In *IEEE International Conference on Edge Computing (EDGE)*, 2019.

[75] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The End-to-End Effects of Internet Path Selection. In *SIGCOMM*, 1999.

[76] Rachee Singh, Sharad Agarwal, Matt Calder, and Paramvir Bahl. Cost-effective Cloud Edge Traffic Engineering with Cascara. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021.

[77] Ramesh K. Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. Overlay Networks: An Akamai Perspective. In *Advanced Content Delivery, Streaming, and Cloud Services*, 2014.

[78] João Luís Sobrinho and Miguel Alves Ferreira. Routing on Multiple Optimality Criteria. SIGCOMM '20, New York, NY, USA, 2020. Association for Computing Machinery.

[79] 3GPP Technical Specification. 3GPP TS 23.501 version 16.6.0 Release 16. https://www.3gpp.org/ftp//Specs/archive/23_series/23.501/23501-g60.zip, 2020.

[80] 3GPP Technical Specification. 5G System Network Function Repository Services. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3345, 2023.

[81] Neil Spring, Ratul Mahajan, and Thomas Anderson. Quantifying the Causes of Path Inflation. In *SIGCOMM*, 2003.

[82] Jie Sun, Feng Liu, Huandong Wang, and Dapeng Oliver Wu. Joint VNF Placement, CPU Allocation, and Flow Routing for Traffic Changes. *IEEE Internet of Things Journal*, 10(2):1208–1222, 2023.

[83] Zoom Support. Zoom system requirements: Windows, macOS, Linux. https://support.zoom.us/hc/en-us/articles/201362023-Zoom-system-requirements-Windows-macOS-Linux, 2023.

[84] Qualcomm Technologies. Making 5G NR a reality. Technical report, 2016.

[85] Qualcomm Technologies. VR and AR pushing connectivity limits. https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/presentation_vr_and_ar_are_pushing_connectivity_limits_web_0.pdf, 2018.

[86] Terraform. Automate infrastructure on any cloud. https://www.terraform.io/, 2024.

[87] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: Traffic Engineering in Dynamic Networks. In *ACM SIGCOMM*, 2006.

[88] Thilo Weghorn, Si Liu, Christoph Sprenger, Adrian Perrig, and David Basin. N-Tube: Formally Verified Secure Bandwidth Reservation in Path-Aware Internet Architectures. In *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*, 2022.

[89] Xipeng Xiao, A. Hannan, B. Bailey, and L. M. Ni. Traffic Engineering with MPLS in the Internet. *Netwrk. Mag. of Global Internetwkg.*, 14(2), March 2000.

[90] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. P4p: Provider Portal for Applications. SIGCOMM '08, New York, NY, USA, 2008. Association for Computing Machinery.

[91] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. Understanding Operational 5G: A First Measurement Study on Its Coverage, Performance and Energy Consumption. In *ACM SIGCOMM*, 2020.

[92] Wen Xu and Jennifer Rexford. MIRO: Multi-Path Interdomain Routing. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, New York, NY, USA, 2006. Association for Computing Machinery.

[93] Zhiying Xu, Francis Y. Yan, Rachee Singh, Justin T. Chiu, Alexander M. Rush, and Minlan Yu. Teal: Learning-accelerated optimization of wan traffic engineering. In *Proceedings of the ACM SIGCOMM 2023 Conference*. Association for Computing Machinery, 2023.

[94] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, Victor Lin, Colin Rice, Brian Rogan, Arjun Singh, Bert Tanaka, Manish Verma, Puneet Sood, Mukarram Tariq, Matt Tierney, Dzevad Trumic, Vytautas Valancius, Calvin Ying, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *ACM SIGCOMM*, 2017.

[95] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G. Andersen. SCION: Scalability, Control, and Isolation on Next-Generation Networks. In *2011 IEEE Symposium on Security and Privacy*, 2011.

A Multi-WAN US Topology

Figure 19 shows all the Azure and GCP cloud components (VMs, VPN gateways, etc.) and the connections between them on the continental US-wide topology we have deployed for evaluating OTTER. It is generated from the output of our Terraform deployment, using a Terraform visualizer called Rover. While difficult to read, the figure indicates the complexity and scale of our evaluation topology across 8 Azure and 8 GCP regions, as it contains many cloud components and connections between them.

Table 6 shows the 8 sets of VPN connections that we use in our evaluation topology. For example, the first row shows that GCP us-central1 is peered with Azure centralus. Each peering consists of 4 VPN connections, each with 2 tunnels, for improved throughput and resilience. This number can be increased or decreased as needed, or further augmented [30], but we have found that 8 tunnels gives us a good tradeoff

GCP Region name	GCP Region location		Azure Region name	Azure Region location
us-central1	Iowa	⟺	centralus	Iowa
us-east1	South Carolina	⟺	eastus	Virginia
us-east4	Virginia	⟺	eastus2	Virginia
us-east5	Ohio	⟺	northcentralus	Illinois
us-south1	Texas	⟺	southcentralus	Texas
us-west3	Utah	⟺	westcentralus	Wyoming
us-west2	California	⟺	westus	California
us-west4	Nevada	⟺	westus3	Arizona

Table 6: VPN connections between Azure and GCP regions.

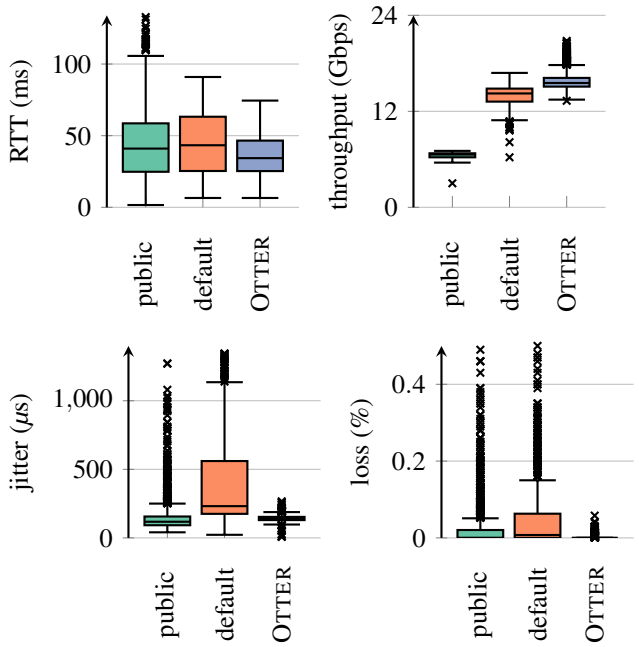


Figure 12: Network performance of paths available by different mechanisms. The jitter vertical axis is chopped at 1200 μs for clarity, and the loss vertical axis is chopped at 0.5%.

between cost and throughput. In picking which Azure region peers with which GCP region, our goal was to mimic how a 5G operator would peer with a cloud provider. Such peerings occur at carrier hotels or "meet-me" points, where two or more ASes may co-locate their routing equipment. Hence we strove to pick pairs of regions that were physically close to each other. Note that when we evaluated OTTER, there was one additional US region offered by both clouds, however we were unable to acquire sufficient compute capacity in those regions.

B Additional Analysis of OTTER Orchestrator Performance

Figure 12 summarizes the characteristics of the paths available by different mechanisms. The "default" box plots show the performance spread between all Src-Dst pairs in our topology (§7.1) across the 24 hour measurement period, when not controlling the path and allowing the two WANs to pick the path [32]. The "public" box plots show the performance spread without any of our security and privacy infrastructure (i.e., no private VNets, no VPN gateways, no VNet peering), where the source and destination use public IP addresses and nothing specified by us in between. The OTTER box plots show the best performing OTTER Orchestrator path for each Src-Dst pair for the relevant network performance metric.

We make several observations. Our secure, private, and scalable deployment topology offers far higher throughput than when using public routing. The {min, average, max} throughput for public is {3.0 Gbps, 6.5 Gbps, 7.1 Gbps}, while for default it is {6.3 Gbps, 14.0 Gbps, 16.8 Gbps}. As prior work [40] noted, some clouds such as GCP limit outbound transfers to 7 Gbps, but those limits do not apply to our use of private VNets and VPN connections. OTTER is able to further improve these numbers to {13.3 Gbps, 15.8 Gbps, 20.8 Gbps}. The median, 25th, and 75th percentile RTTs are slightly higher for default compared to public, but with far less spread above the 75th percentile. OTTER offers the lowest RTT at the 25th and higher percentiles. OTTER also offers the tightest jitter and lowest loss.

Figure 13 shows the performance improvement when comparing different mechanisms. "O-D" shows for every Src-Dst pair, for every round of measurements, the additional performance that the OTTER Orchestrator achieves compared to the default path on our topology. "O-P" provides OTTER's improvement over the public path when using public IP addresses and no private networking functions. "P-D" shows the improvement of the public path over the default private path. Figure 14 shows the same data, but as a percentage improvement. When considering the "O-D" boxes, all the values are 0% or higher, because the OTTER Orchestrator will never pick a path that has worse performance for the desired metric. The "O-P" and "P-D" boxes show that for RTT, jitter, and loss, the use of private networking functions introduces occasional performance regressions. We believe these functions are run by cloud providers in VMs, and that introduces small amounts of occasional jitter and packet loss as traffic goes up and down the software stack. Nonetheless, OTTER improves on this, by, for example, picking paths that

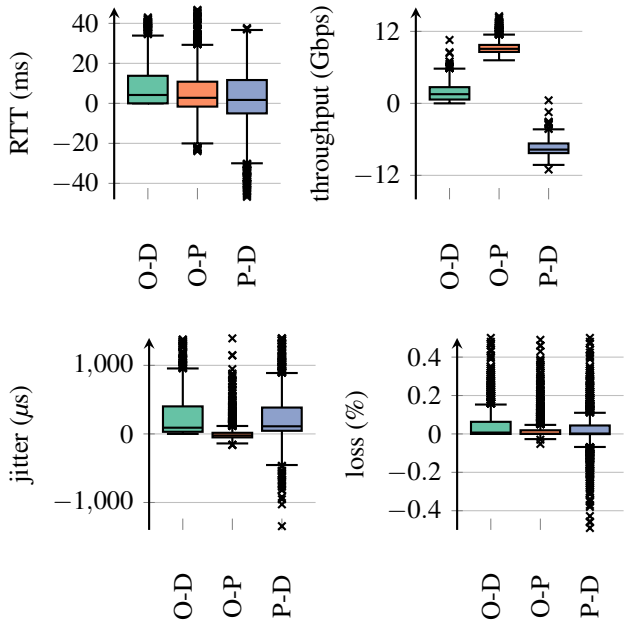


Figure 13: Improvement in network performance. On the horizontal axis, "O-D" shows the performance improvement of OTTER over default, "O-P" is OTTER over public, and "P-D" is public over default.

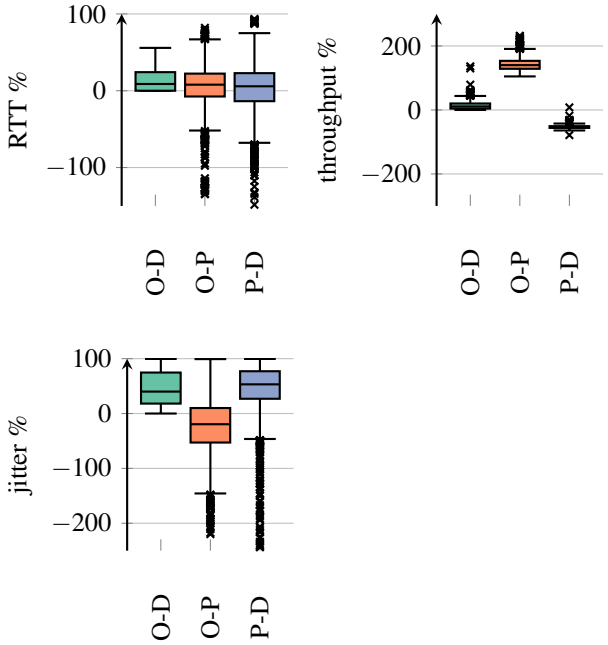


Figure 14: Percentage improvement in network performance. On the horizontal axis, "O-D" shows the performance improvement of OTTER over default, "O-P" is OTTER over public, and "P-D" is public over default. Loss is not presented because the large spread of outliers on small values makes the graph difficult to read.

go through other instances of VPN gateways that are not suffering from noise at the moment. This is seen in Figure 12, where on each metric of throughput, RTT, jitter, and loss, OTTER offers the best path.

C Evaluation of OTTER Optimizer on Different Mean Flow Durations

Table 7 summarizes the results of how well OTTER allocates flows of varying durations using the heuristics described in §7.2: GREEDY, PER+PATHPIN, PER+DSTPIN, PER, OPT+PATHPIN, OPT+DSTPIN, and OPT. These results, like results in §7.2, show that heuristics relying on a greedy allocation (GREEDY, PER+PATHPIN, PER+DSTPIN, PER) typically allocate less traffic, but are more practical for real-time decisions. We also see that with longer flow durations, the amount of bytes allocated per flow decreases for each algorithm. This is because, with longer flows, there are less resources available for later flows, because earlier flows are still active.

D Service Demand Generation

The traffic composition on 5G networks is dynamic and can evolve over time, making it difficult to predict what kinds of demands will be present. Applications will also have diverse service requirements across various performance metrics. To mimic 5G traffic in our evaluations, we generate flows by assigning them an "application profile" based on service guidelines from the 3GPP and other sources.

We selected a set of example applications outlined by the 3GPP standard [79] that represent a diverse range of performance demands that 5G networks are expected to support. These applications are listed in the first column of Table 5. In § 5.7.4 of the relevant 3GPP standard [79] are guidelines for the packet delay budget and packet error rate for each application, which we use to determine RTT and packet loss demands. These are shown in the latency and loss columns in Table 5. To obtain RTT demands, we double the one-way latency guidelines listed in that 3GPP standard. However, it does not provide information on throughput and jitter, so we infer them from other sources [1, 58, 66, 71, 83, 85]. In cases where the throughput demands are unclear or conflicting, we set a default of 1 Mbps for a single 4-tuple flow. We chose real-time gaming as one application with explicit jitter demands, doubling the 10 ms one-way jitter guideline from [71] to match the round-trip path attributes measured on our topology.

As mentioned in §4, OTTER models service demands as a vector of demand functions which map path performance characteristics to a tolerance coefficient in the range $(0, 1]$, where 1 is best. A demand function is able to capture com-

mean flow duration(s)	mean flow arrival rate (flows/s)	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	20K	0.95	0.99	0.99	0.99	0.99	1.0	1.0
5	30K	0.61	0.84	0.84	0.93	0.92	0.95	0.98
5	40K	0.60	0.78	0.82	0.85	0.85	0.87	0.91
10	20K	0.64	0.80	0.82	0.91	0.87	0.90	0.96
10	30K	0.55	0.74	0.76	0.83	0.81	0.83	0.87
10	40K	0.44	0.64	0.64	0.66	0.68	0.69	0.70
20	20K	0.50	0.71	0.71	0.79	0.79	0.79	0.84
20	30K	0.30	0.47	0.47	0.54	0.54	0.54	0.58
20	40K	0.17	0.34	0.34	0.41	0.40	0.41	0.45
30	20K	0.36	0.54	0.55	0.68	0.66	0.66	0.72
30	30K	0.21	0.38	0.38	0.47	0.43	0.44	0.50
30	40K	0.23	0.35	0.35	0.37	0.37	0.38	0.39
60	20K	0.45	0.64	0.63	0.67	0.67	0.68	0.69
60	30K	0.32	0.45	0.46	0.49	0.47	0.49	0.50
60	40K	0.21	0.32	0.33	0.35	0.34	0.35	0.36

Table 7: Fraction of allocated bytes across flows for varying mean flow durations.

plex service demands, which we believe will be necessary to capture the QoS characteristics expected by 5G applications. In our implementation, a demand function can be either a step, linear, or rational function, with inflection points randomly selected from within ±25% of the service guidelines mentioned above. For flows that are indifferent to the metric (*e.g.,* jitter), the demand function is constant at 1.

The service demand generation process is as follows: each flow is first randomly assigned an application profile. For each performance metric, we randomly assign a function type, then generate inflection points. For example, an augmented reality flow may have a step-type demand function for RTT that looks like the following:

$$\mathcal{D}_f^{rtt}(\sigma_p^{rtt}) = \left\{ \begin{array}{ll} 1, & \text{if } \sigma_p^{rtt} \leq 21.2\ ms \\ \varepsilon, & \text{if } \sigma_p^{rtt} > 21.2\ ms \end{array} \right\}$$

We then randomly select a value within ±25% of the application's guideline throughput demand.

E Evaluation of OTTER Optimizer on Different Flow Distributions

Table 8 summarizes the results of how well OTTER satisfies flow RTT requirements for varying distributions of flows, using the heuristics described in §7.2. We vary the proportion of flows with strict RTT requirements (RTT $\leq$ 60ms) and report the average RTT satisfaction for those RTT-sensitive flows. We find that for lower arrival rates, the proportion of RTT-sensitive flows does not affect their satisfaction, as there is less contention for resources. Like the results in §7.2, the GREEDY heuristic provides the lowest satisfaction. The heuristics that pin flows to either paths or destinations also provide less satisfaction than PER and OPT, as RTT-sensitive flows may get pinned to a high-RTT path, even as capacity on lower-RTT paths becomes available.

F Evaluation of OTTER Optimizer on Different Network Sizes

Table 9 summarizes the results of how well OTTER allocates flows on networks of various sizes using the same mean flow arrival rates and the heuristics described in §7.2: GREEDY, PER+PATHPIN, PER+DSTPIN, PER, OPT+PATHPIN, OPT+DSTPIN, and OPT. These results are qualitatively similar to those described in §7.2, with OPT being able to allocate more traffic compared to the other algorithms, but the PER heuristics scaling better with higher traffic load on the same network topology. We can additionally see that as we increase the number of paths between each Src-Dst pair, all algorithms are able to improve the amount of bytes allocated per flow. This is explained by the increase in path diversity, which gives more options for flows to be mapped to paths that better suit their service demands. The same effect can be seen as the number of sources and destinations increases. This increases the overall resource capacity of the network, since there are more links and compute resources for the same traffic load.

We also evaluate OTTER on larger topologies with larger flow arrival rates. We choose topology sizes to match the scale of contemporary TE systems, B4 [41] (approximately 30 DC sites globally) and SWAN [39] (evaluated with up to 40 DC nodes). In each topology, we set 75% of the destinations to be edge nodes, and set the rest to be DC nodes. We generate path, destination, and flow attributes (except flow arrival rates) using the values in Table 4. We scale up the flow arrival rates because as the topology grows, smaller arrival rates do not produce enough traffic to cause resource contention, and OTTER is able to achieve 100% allocation. We set the mean flow arrival rate to the number of source nodes times the number of destination nodes. The results are summarized in Table 10. The allocation from the OPT+PATHPIN, OPT+DSTPIN, and OPT heuristics are qualitatively similar to

% flows with RTT $\leq$ 60ms	mean flow arrival rate (flows/s)	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
30	20K	0.34	0.38	0.48	0.77	0.50	0.57	0.85
30	30K	0.33	0.36	0.43	0.65	0.43	0.52	0.71
30	40K	0.22	0.17	0.22	0.39	0.23	0.31	0.43
50	20K	0.48	0.52	0.56	0.81	0.57	0.65	0.87
50	30K	0.31	0.33	0.37	0.58	0.38	0.46	0.64
50	40K	0.20	0.21	0.27	0.46	0.27	0.34	0.53
70	20K	0.48	0.52	0.58	0.79	0.55	0.66	0.85
70	30K	0.22	0.27	0.29	0.53	0.34	0.44	0.59
70	40K	0.21	0.24	0.26	0.46	0.29	0.38	0.53

Table 8: Average RTT satisfaction for flows with varying application distributions.

		Small (4 srcs, 4 dsts)						
paths per src/dst pair	mean flow arrival rate (flows/s)	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	20K	0.18	0.36	0.36	0.38	0.36	0.37	0.40
5	30K	0.16	0.27	0.27	0.28	0.28	0.27	0.29
5	40K	0.15	0.20	0.20	0.20	0.20	0.21	0.21
10	20K	0.31	0.56	0.57	0.62	0.59	0.60	0.64
10	30K	0.27	0.43	0.43	0.45	0.45	0.45	0.47
10	40K	0.23	0.31	0.31	0.31	0.32	0.32	0.32
20	20K	0.41	0.65	0.63	0.83	0.74	0.72	0.88
20	30K	0.34	0.54	0.56	0.63	0.61	0.62	0.68
20	40K	0.20	0.36	0.36	0.46	0.45	0.45	0.53
		Medium (8 srcs, 8 dsts)						
paths per src/dst pair	mean flow arrival rate (flows/s)	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	20K	0.78	0.87	0.89	0.93	0.91	0.92	0.95
5	30K	0.34	0.56	0.57	0.67	0.63	0.64	0.72
5	40K	0.40	0.55	0.56	0.56	0.58	0.58	0.60
10	20K	0.74	0.86	0.90	0.95	0.93	0.93	0.99
10	30K	0.48	0.69	0.71	0.75	0.76	0.77	0.82
10	40K	0.52	0.68	0.69	0.73	0.76	0.78	0.79
20	20K	0.92	0.99	0.99	0.99	0.99	0.99	1.0
20	30K	0.59	0.79	0.81	0.88	0.91	0.92	1.0
20	40K	0.52	0.68	0.70	0.79	0.83	0.84	0.94
		Large (12 srcs, 12 dsts)						
paths per src/dst pair	mean flow arrival rate (flows/s)	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	20K	0.95	0.98	0.99	0.99	0.99	0.99	1.0
5	30K	0.71	0.88	0.89	0.93	0.95	0.96	0.99
5	40K	0.61	0.77	0.80	0.87	0.87	0.91	0.97
10	20K	1.0	0.99	1.0	1.0	0.99	1.0	1.0
10	30K	0.93	0.99	0.99	0.99	0.99	0.99	1.0
10	40K	0.65	0.82	0.85	0.88	0.96	0.96	0.99
20	20K	1.0	0.99	1.0	1.0	0.99	1.0	1.0
20	30K	0.95	0.99	0.99	1.0	0.99	0.99	0.99
20	40K	0.79	0.93	0.93	0.94	0.99	0.99	1.0

Table 9: Fraction of allocated bytes across flows for varying network sizes.

(12 srcs, 12 dsts)								
paths per src/dst pair	**mean flow arrival rate (flows/s)**	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	72K	0.34	0.49	0.49	0.53	0.59	0.60	0.64
8	72K	0.38	0.54	0.55	0.59	0.66	0.67	0.72
10	72K	0.45	0.61	0.63	0.62	0.74	0.77	0.81
20	72K	0.49	0.64	0.66	0.62	0.84	0.84	0.95
(16 srcs, 16 dsts)								
paths per src/dst pair	**mean flow arrival rate (flows/s)**	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	128K	0.30	0.48	0.47	0.51	0.55	0.56	0.59
8	128K	0.29	0.45	0.45	0.50	0.55	0.57	0.64
10	128K	0.30	0.46	0.47	0.53	0.60	0.62	0.69
20	128K	0.42	0.57	0.59	0.57	0.79	0.82	0.90
(20 srcs, 20 dsts)								
paths per src/dst pair	**mean flow arrival rate (flows/s)**	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	200K	0.27	0.45	0.45	0.51	0.54	0.55	0.59
8	200K	0.30	0.48	0.48	0.54	0.58	0.60	0.67
10	200K	0.31	0.49	0.49	0.54	0.62	0.62	0.70
20	200K	0.34	0.50	0.52	0.55	0.67	0.69	0.77
(32 srcs, 32 dsts)								
paths per src/dst pair	**mean flow arrival rate (flows/s)**	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	512K	0.20	0.36	0.37	0.43	0.42	0.42	
8	512K	0.23	0.39	0.39	0.45	0.47	0.46	
10	512K	0.25	0.41	0.42	0.47	0.49		
20	512K	0.27	0.42	0.43	0.42			
(64 srcs, 64 dsts)								
paths per src/dst pair	**mean flow arrival rate (flows/s)**	GREEDY	PER+PATHPIN	PER+DSTPIN	PER	OPT+PATHPIN	OPT+DSTPIN	OPT
5	2M	0.17	0.25	0.26	0.24			
8	2M	0.18	0.25	0.25	0.26			
10	2M	0.19	0.22	0.23	0.20			
20	2M	0.18	0.21	0.23	0.19			

Table 10: Fraction of allocated bytes across flows for varying network sizes, with mean flow arrival rates scaled according to network size. Some values for OPT heuristics are missing because of the excessive and unrealistic compute times these heuristics need.

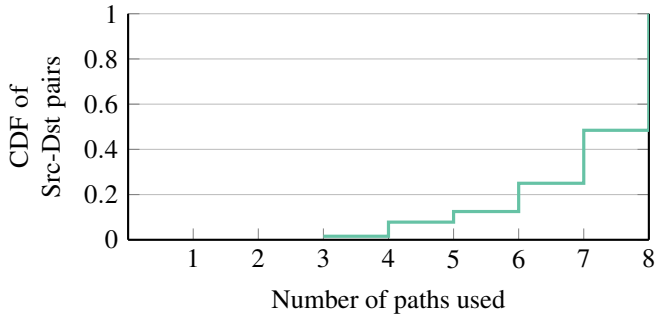


Figure 15: Number of paths between each Src-Dst pair that carries at least one flow.

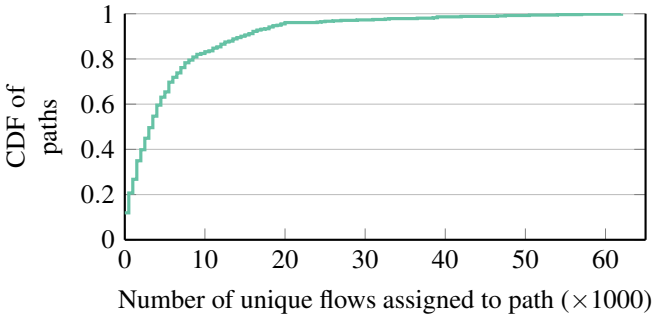


Figure 16: CDF of the number of unique flow aggregates assigned to each path in the network.

results in §7.2. As the flow arrival rate grows, there is a more significant gap between the PER+PATHPIN, PER+DSTPIN, and PER and the OPT heuristics. This is because the OTTER Optimizer has a longer runtime, which means the PER heuristics rely more on greedily allocating flows.

G OTTER's Path Usage

We present additional analysis results on the path usage of OTTER's allocation methods. This is in contrast to the service demand satisfaction and byte allocation metrics presented in the main body of the paper. We use this to demonstrate that the evaluated topologies have realistic capacity constraints and path metrics, and that the OTTER Optimizer does not simply assign the majority of flows to a single, particularly attractive, hypothetical path (*e.g.,* one that happens to have extremely low RTT, high throughput, and low loss).

Figure 15 shows the number of paths between each Src-Dst pair that is assigned at least one flow under the PER+PATHPIN allocation method. We use the default topology described in § 7.2, with eight sources, eight destinations, and eight paths between each Src-Dst pair. We can see that for more than half of all Src-Dst pairs, all eight paths are employed. In 75% of cases, at least seven paths are used. This demonstrates that OTTER does not simply favor a small number of well-provisioned paths, but that it balances traffic across those that are available to maximize satisfaction.

We further show in Figure 16 the distribution of the number of unique flows that are assigned to each path. We use a mean flow arrival rate of 40K flows per second for a period of 30 seconds from the time of the first flow. While there is a long tail in terms of number of flows assigned to each path, almost 90% of all paths in the network are allocated at least one flow within first 30 seconds.

H WAN Underlay Opportunity

The focus of OTTER is as an overlay, in setups which can be constructed by customers of public clouds without any special privileges. WANs, whether they be cloud WANs or 5G operator WANs, have a dedicated underlay network. This

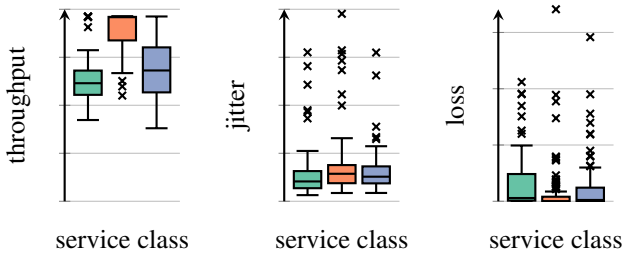


Figure 17: Relative network performance between two VMs across the US, with each box representing a different class of service. Jitter graph is chopped on vertical axis for clarity – outliers go up to 2X, 3X, 3X higher for each class from left to right.

underlay is typically comprised of layer 2 (such as MPLS or SONET) segments which are stitched together to form layer 3 links that interconnect various locations (DCs and/or PoPs). Some WANs maintain multiple paths in the underlay between any two locations [45], as well as different queues at the routers in the WAN. The combination of the two can be used to create different classes of service, with different properties (such as bandwidth, RTT, jitter, loss, reliability, cost). While typically not exposed to customers, we wonder if these underlay network capabilities could even further enhance the value of OTTER as part of a native offering by a WAN.

To briefly demonstrate the value of using these underlay capabilities, we show the relative network performance between two VMs, one in the US West coast and one in the US East coast, on a cloud WAN, over three different classes of service, on a two hour window in the middle of a US week day. We run approximately 100 iPerf3 measurements lasting 10 s each for each of UDP and TCP on each class of service.

Figure 17 shows that each class offers a different tradeoff. The class in the middle tends to offer more throughput but also slightly more jitter than the other two. The class on the right offers more throughput and less loss than the left, but more uncertainty in the throughput. As shown in Figure 18, these tradeoffs are not static, necessitating a dynamic approach to allocating traffic to underlay offerings. RTT is also different and time-varying between these classes – the

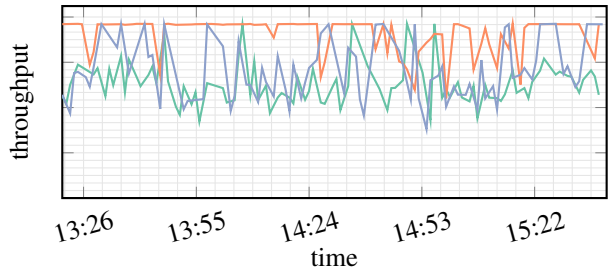


Figure 18: Relative throughput between two VMs in the US. Each line represents a different class of service.

minimum RTT is the same for all three, but the maximum is 1.7× higher for the middle class.

We have removed the units on the vertical axes on the graphs as these results are obtained from proprietary access to underlay paths that are not available to the public. However, the key takeaway is visible, that there are different underlay paths offering different tradeoffs in network performance. As future work, we are considering how to construct hybrid paths across one WAN with only overlay paths and another WAN with access to underlay paths.

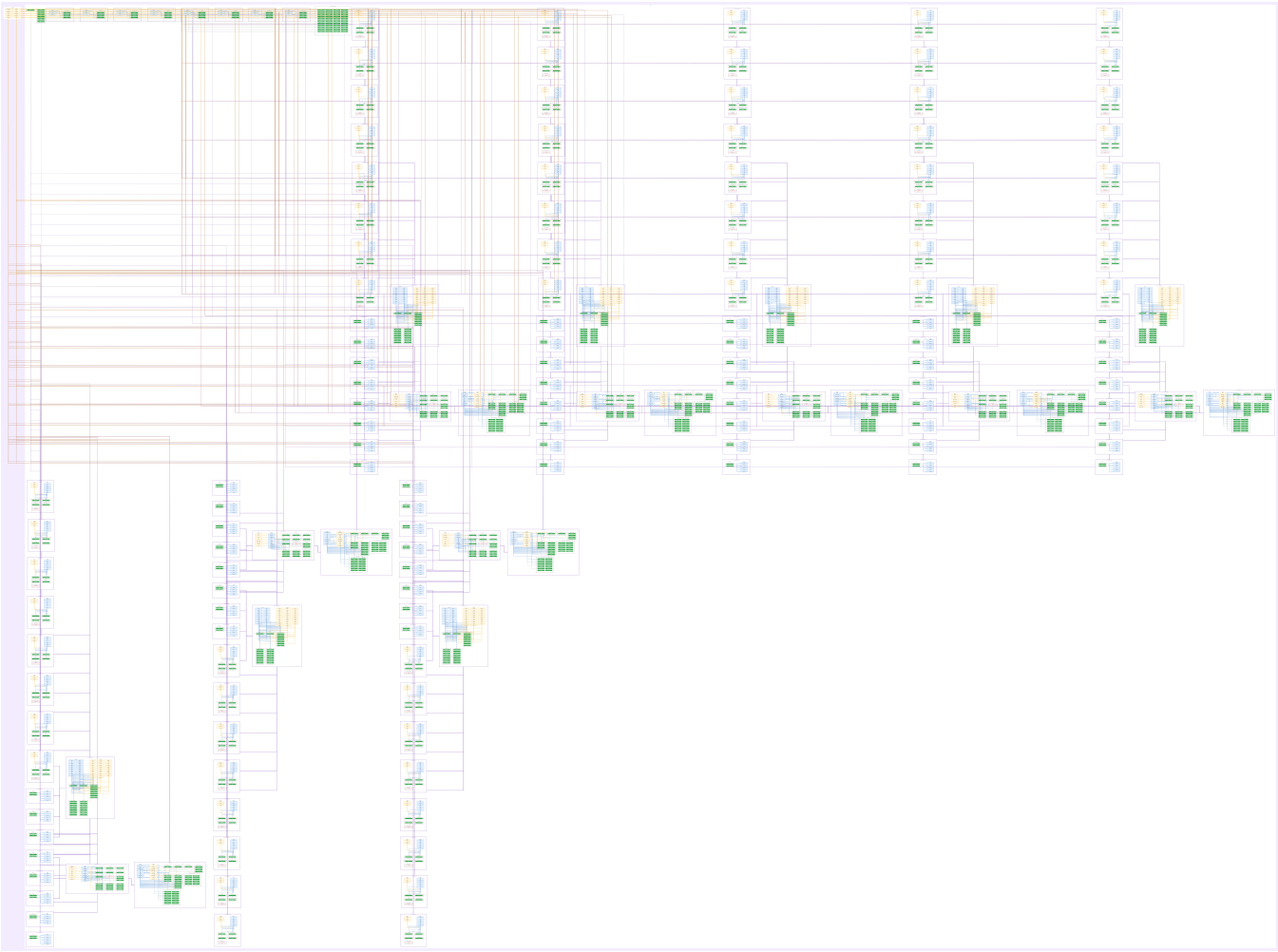


Figure 19: OTTER multi-WAN topology deployed on continental US scale across Azure and GCP.