Understanding Cross-Cloud Interconnects: Hands-On Measurements and Cost Optimization

Eitan Eliav[1], Isaac Keslassy[1,2], David Breitgand[3], Dean H. Lorenz[3], Avi Weit[3]
[1] *Technion* [2] *UC Berkeley* [3] *IBM Research – Haifa*

***Abstract*—New services such as Google Cross-Cloud Interconnect (CCI) address the rise in fast and large-scale cross-cloud data transfers. CCI offers dedicated high-throughput links with low per-GB transfer costs, but also involves high fixed leasing fees and multi-day provisioning delays. This combination makes cost optimization difficult because traffic patterns are unpredictable.**

This paper presents the first comprehensive study of CCI-like services. We begin with an empirical characterization of CCI and its alternatives using direct measurements across AWS–GCP interconnects. We then introduce TOGGLECCI, a new dynamic cost-optimization algorithm designed to handle provisioning delays and uncertainty in future demand. TOGGLECCI adapts by switching between VPN and CCI based on cost trends observed over a sliding time window. We prove that TOGGLECCI achieves asymptotic optimality under sustained high-demand or low-demand regimes. Finally, using real-world traffic traces, we show that TOGGLECCI consistently tracks the best static policy for each scenario and delivers substantial cost savings.

***Index Terms*—cross-cloud interconnect, multi-cloud, cost optimization, cloud measurements, cloud transfers**

I. INTRODUCTION

Organizations increasingly deploy massive and complex applications across multiple cloud providers to improve availability, reduce vendor lock-in, and optimize cost and performance. This multi-cloud trend brings new technical challenges, particularly in the area of cross-cloud connectivity. Moving data between clouds is not only a performance concern but also a major contributor to operational costs [1]–[6].

The simplest way to transfer data across clouds is via the Public Internet. Although this approach is flexible and easy to set up, it suffers from variable performance, limited security guarantees, and potential privacy risks. To address these privacy concerns, cloud providers offer private networking options based on VPNs [7]–[10]. However, this method can become costly for users who transfer large volumes of data.

Another approach is to use a dedicated physical connection that enables users to establish a direct link across clouds. A notable recent advancement in this space is Google's *Cross-Cloud Interconnect (CCI)* [11]–[13]. CCI enables users to manage cross-cloud connectivity directly through the GCP Console, with a native Google Cloud API and typical 99.99% availability SLA guarantees. CCI can be seen as an evolution of the hybrid cloud, in which the user's premises are connected to the cloud via a private dedicated link bypassing the Public Internet [14]–[16]. CCI is the first cloud native solution for direct collocated cross-cloud connectivity that automates the process of establishing communication links. Competing solutions either require manual efforts or rely on intermediary partners to establish connectivity. Azure also has recently provided another collocated solution with native connectivity to Oracle, but it does not include other clouds yet [17], [18]. CCI is also an alternative to third-party vendors such as Megaport and Aviatrix who offer physical and virtual private connectivity solutions for cross-cloud scenarios [19]–[21].

In this paper, we focus on exploring the tradeoffs and challenges involved with the cloud-native CCI scenario, as there is no available academic work or objective evaluation about CCIs. In particular, we seek to explore two core metrics:
(1) **Performance:** How does CCI compare to VPN and the Public Internet? Does it indeed provide its advertised bandwidth? Do factors like the number of connections or the cloud region impact its performance?
(2) **Cost:** The cost tradeoff between CCI and VPN is hard to model, because of three main challenges. (a) *Time lag*: unlike VPNs, which are pay-as-you-go and can be provisioned instantly, CCI requires advance provisioning that typically takes several business days. Therefore the user essentially needs to bet on future demand. (b) *Complex pricing structure:* Cloud provider pricing is often complex and varies significantly across regions and services, especially in multi-cloud setups, leading users to rely on specialized calculators to estimate costs [22], [23]. CCI further increases this complexity because it is a shared organizational resource: the organization pays a base leasing fee for the interconnect, and each connection incurs an additional charge. So the user needs to consider all of its applications at once to decide whether to lease a CCI link. (c) *Different per-GB pricing mechanisms:* Comparing the CCI pricing model with VPN is non-trivial, as VPN pricing is based on tiered egress pricing, where the per-GB cost decreases with higher usage, whereas CCI has a fixed per-GB cost.

Contributions. We start by exploring the *performance* of CCI with the first hands-on measurement study of CCI. Our experiments reveal practical behaviors that are not captured by provider documentation. We find that CCI links strictly enforce their nominal bandwidth at the link level. Nevertheless, individual users sharing the same CCI link may sometimes observe throughput exceeding the capacity they provisioned.

Second, we focus on the *cost* tradeoff between CCI and VPN. While performance can be directly measured, cost is more complex, as it depends on the algorithm that the user adopts to decide on whether to use CCI or VPN. We prove that there is no optimal online algorithm, and in fact that no

algorithm can even guarantee a fixed competitive ratio that is independent of the problem settings. Instead, we argue that in practice we can use past demand to model future demand, and introduce a heuristic online algorithm, TOGGLECCI, which dynamically switches between VPN and CCI based on recent demands. TOGGLECCI is designed to deal with the complex provisioning delays and leasing commitments of CCI, along with the complex pricing structure of CCI and VPN that depends on both time and volume. We prove that TOGGLECCI achieves near-optimal cost in both low and high demand scenarios.

Finally, we evaluate TOGGLECCI on real-world workloads, MIRAGE and Puffer, as well as synthetic workloads, and show that it consistently outperforms baseline strategies. For example, with the MIRAGE workload, at the breakeven traffic rates where the total VPN and CCI costs are equal, TOGGLECCI reduces the total cost by an average factor of $1.8\times$. We evaluate many configurations to analyze the algorithm sensitivity: transfers from GCP to AWS and in the reverse direction; from GCP to Azure and back; both in single-continent and in multi-continent scenarios. Across all configurations, the results consistently show that TOGGLECCI either outperforms alternative methods or approaches the best performance.

The source code is publicly available [24].

II. Related Work

While there is a rich body of literature on cross-cloud networking and measurement, none of the prior works study the performance or cost optimization of CCI.

For example, recent efforts such as [25], [26] evaluate multi-cloud networking performance, and CloudCast [27] presents a system for monitoring and comparing cross-cloud latency and throughput across AWS, GCP, and Azure. Others focus more narrowly, e.g., on characterizing throughput dynamics within GCP [28]. However, to the best of our knowledge, no prior work conducts hands-on measurements or develops cost-aware frameworks for CCI usage.

Similarly, there is extensive research on multi-cloud task scheduling and resource management [29], [30]. Another work, called Paraglider [31], provides an abstraction layer for multi-cloud networking, enabling cloud-agnostic specification of networking and simplifying deployment across multiple cloud providers. However, it does not consider privacy or cost optimization. In addition, Skyplane [1] formulates the cost-throughput trade-off as a Linear Programming (LP) problem. It dynamically plans and routes cross-cloud data transfers in real-time, achieving substantial speedups while controlling egress costs. However, Skyplane relies solely on Public Internet paths and does not incorporate any privacy guarantees. CloudPilot [32] also looks at optimizing the cost of cloud-proxy placement. Finally, many commercial systems also aim to reduce multi-cloud expenses [33]–[35]. They offer integrated solutions that optimize total cloud billing. Still, none of these studies and systems consider the unique characteristics or pricing constraints of CCI links to optimize a CCI vs. VPN tradeoff, rendering them inapplicable in our setting.

III. Background

Google Cross-Cloud Interconnect (CCI). CCI is a service that offers high-bandwidth, dedicated connectivity between Google Cloud and other cloud providers [11]. This connectivity is established through designated *colocation facilities*, each tied to a specific external provider and geographically restricted, typically within a single continent. CCI is a shared resource, allowing multiple connections by attaching separate VLAN attachments. Deployments are available in 10 Gbps or 100 Gbps capacities and are backed by Google's 99.99% availability SLA, with no additional hardware required.

Establishing a CCI connection is not immediate: the process includes email correspondence with Google's service center and may take a few business days to complete [36]. Once established, the CCI link is billed at a fixed hourly rate, regardless of the volume of traffic transmitted. In addition, there is a fixed price per GiB of data egress from GCP.

Virtual Private Cloud (VPC) is a logically isolated virtual network within a public cloud [7]–[10]. Users deploy all workloads, such as virtual machines (VMs) and Kubernetes (K8s) clusters, inside VPCs, which are not accessible to other users unless explicit permission is granted.

VLAN attachments [37] provide the logical link that connects a specific VPC to a CCI within the same geographic area. Each VPC that wants to use an existing CCI must lease its own VLAN attachment. These attachments are available in a range of bandwidth options and incur an hourly charge based on the selected capacity [38]. Unlike the CCI provisioning process, VLAN attachments are fully automated and become active immediately upon request. At least one VLAN attachment is required for an active CCI connection, and multiple attachments can be associated with the same CCI.

Establishing a CCI link involves several steps: (1) Lease a physical port from both Google and another cloud provider at the same colocation facility; (2) Create a virtual connection from VPCs to the physical port (a VLAN attachment in GCP and a Virtual Interface (VIF) in AWS); (3) Establish routing policies and BGP sessions to enable inter-cloud traffic. Full technical details of the setup process are available in [36].

Virtual Private Network (VPN) is a service supplied by all major cloud providers [39]–[42]. VPNs offer high flexibility and are easy to deploy, making them a default choice for many multi-cloud scenarios. Cost-wise, VPNs typically involve a fixed hourly charge for maintaining the connection and variable fees for data transferred out of the network.

IV. Hands-on evaluation of cross-cloud connectivity

In this section, we study whether CCI provides its full capacity in a fair manner, and how it compares to the VPN and Public Internet alternatives, especially when users try to send traffic beyond the nominal capacity.

A. Preliminary Experiments

Prior to performing the main experiments, we need to make sure that VMs do not have bottlenecks on memory or CPU. We performed experiments with `iperf3` within the same VPC in the same region in the same cloud for AWS and GCP respectively, testing connectivity between pairs of VM instances of different types. In both GCP and AWS, we were able to obtain the nominal NIC bandwidth guarantees.

Beyond VM NIC nominal capacity. When considering cross-cloud communication, there are caveats that should be taken into account. When ordering a VM instance of a given flavor, it is important to be alert to the fact that this is a virtual, i.e., logical appliance. For example, a VM NIC is, in fact, an elastic cloud resource that shares the underlying physical NIC with other tenants on the same physical host subject to some policy, which is not always fully transparent when cross-cloud communication happens. A NIC of a given nominal capacity can perform from slightly below to considerably higher than its nominal capacity. For example, *we observed a* $2\times$ *higher throughput* on short-lived (60 seconds) bursty traffic obtaining 4.16 Gbps on a nominally 2 Gbps NIC.

Interestingly, this happens across the board for both CCI and Public Internet (premium and regular tiers). It appears that cloud vendors strive to provide the nominal capacity guarantee (even though this is not always achieved on the short-lived connections) and are willing to share spare spot capacity when it is available. Even though this does not entail change in the cost of NIC, the traffic volume is still billed as usual. Thus, this is an advantageous strategy for cloud providers. The details of the throttling and capacity sharing policies are not documented though. In general, *our experiments suggest that these distributed mechanisms require some warm-up time* and kick in after some relatively long time, such as $3-5$ minutes. At this point, the observed throughput converges to the nominal capacity. Fully explaining this behavior is out of scope for this paper, but we observed that in a single cloud, convergence to the nominal capacity occurs much faster.

Beyond VLAN nominal capacity. For our evaluation, we used a CCI link of 10 Gbps and varied the number of VLAN attachments and their capacities between $1-10$ Gbps. Similarly to NICs, VLAN attachments are not physical VLANs, but logical cloud resources. A VLAN attachment actual performance typically adheres to its nominal capacity, but occasionally for short-lived bursty traffic patterns, it can accommodate much more traffic than allowed by the nominal specification. *We observed up to* 70% *higher throughput* than the nominal specification. We did not observe VLAN attachment performance below the nominal specification. Note that we explored VLAN attachments and CCI performance using default traffic policies. We were not interested in validating policies for segregating traffic of the same user because they do not affect cost, which is based on the hourly fee per CCI link and VLAN attachment capacities, and the volume of the egress traffic.

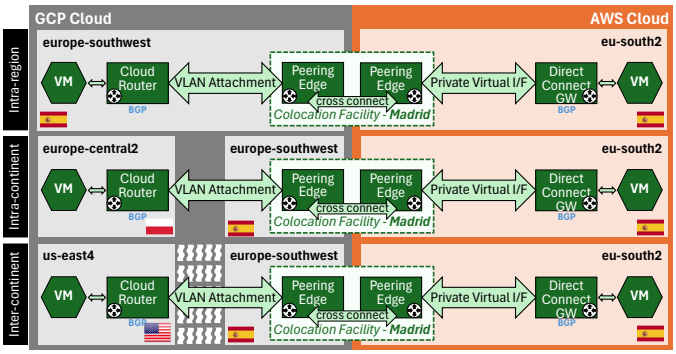


Fig. 1. Measurements testbed: measuring network performance across AWS and GCP.

Beyond CCI nominal capacity. In contrast to the virtual NIC and VLAN attachment resources, we observed that a CCI link never allows to exceed its nominal capacity. This is an expected behavior, since the CCI links are physical resources.

Overbooking VLANs. In another set of experiments, we want to understand what happens to the traffic of the user under the default policy when VLAN attachments and/or CCI links are overbooked. It turns out that if a VLAN attachment is overbooked, but the total egress/ingress traffic is below the CCI link's capacity, the TCP connections using this VLAN attachment receive their fair shares. Sometimes the TCP connections can exceed the fair shares, because the total capacity of the VLAN attachment is exceeded thanks to transparent spot capacity sharing by the vendor. Furthermore, when multiple VLANs share the same CCI link, the link capacity is shared fairly between the VLANs, and occasionally VLANs can obtain more than their entitlement.

We also run heavy overbooking experiments with total VLAN capacity sharing the same CCI link twice exceeding the capacity of the CCI link (i.e., the total VLAN capacity used was 20 Gbps). In this case, the CCI link allows exactly 10 Gbps throughput minus 5% L4+L2 overhead, and each of the two 10 Gbps VLANs receive about 5 Gbps fair share.

An additional consideration in cross-cloud connectivity is that *cloud vendors might apply policies to egress and ingress traffic differently*, without making the details completely transparent. All the factors above underline our need to obtain the baseline measurements through experimentation rather than simply assuming that the cloud resources will always perform as their nominal specifications, and use the cheapest VM and VLAN configurations to measure performance of CCI. The next section describes the testbed configuration that we settled on following the preliminary experiments.

B. Testbed and Methodology

Fig. 1 depicts our testbed. In all experiments, VMs run `Ubuntu 24.04` and we use `iperf 2.1.9` to measure throughput. We arbitrarily set an *anchor* VPC in AWS region `eu-south-2` (Madrid, Spain). With respect to this anchor VPC, we use three subnets in the

GCP VPC[1] in `europe-southwest1` (Madrid, Spain: intra-region), `europe-central2` (Poland: intra-continent), and `us-east4` (Virginia, USA: inter-continent). We use `n2-standard-32` VM type in GCP. It has `32 VCPUs`, `128 GB` of memory, and up to `32 Gbps` default egress bandwidth. In AWS, we use `m5.12.xlarge` VM type. This machine type has `48 VCPUs`, `192 GB` of memory, and `12 Gbps` of default egress bandwidth.

For each connectivity option (CCI, VPN, Public Internet Premium, and Public Internet Standard), we measure throughput in two directions, i.e., by using the anchor VPC in AWS as `iperf3 server` and `iperf3` client alternately.

For each experiment configuration (i.e., connectivity option, direction, and collocation option), we also control the utilization level of the cross-cloud link to model (a) low utilization (below 30%), (b) high utilization (around 70%), and (c) complete saturation (100%). The saturation level is being controlled by the number of `iperf3` connections and bandwidth limit per-connection. To illustrate:

```
iperf3 -J -Z -n 20G -i 0 -b 500M \
                    -c 10.xxx.xxx.xxx -P 20
```

completely saturates the 10 Gbps CCI connection, albeit for a short period of time.

We repeat each experiment 30 times to obtain a data point, compute the average total throughput, and its variance. After a set of 30 runs, we completely clean the system and start from a clean slate to prevent spurious correlations and side effects across different experiment configurations.

Thus, overall, we run 4 (connectivity configurations) x 2 (directions) x 3 (collocation options) x 3 (utilization levels) x 30 (replications) $= 2160$ experiments in our main experimentation study. We show a subset of measurements in this paper and will share the full set publicly upon publication.

To make sure that the experiments are not affected by transient conditions in the clouds, we repeat the experiments at different random dates. As one can appreciate, such experiments takes time and budget. They are also difficult to complete in one pass because now and then some exceptions and errors happen. Because of the volume of the experiments, the sometimes painstaking discovery process of undocumented behavior of the cloud VPN networking, the configuration changes required as explained in the next subsection, and time and the costs involved, we were not able to run every experiment under exactly the same conditions. To that end, we segregate the data sets obtained under different conditions and discuss them separately.

C. Cross-cloud Measurements

The VPN tunnel bandwidth is limited *nominally* by AWS Site-to-Site VPN quota, which allows 1.25 Gbps [43] per tunnel. The GCP CloudVPN quota allows 3 Gbps per tunnel. Since we wanted to keep the costs down, we only used one VPN tunnel in the VPN-based experiments and, therefore, we expected that the best throughput attainable in our experiments, will be limited to 1.25 Gbps.

Bypassing VPN throttling. We discovered that CloudVPN can behave very differently for different traffic profiles. In many cases, we obtained throughput that exceeds the upper limit of the AWS Site-to-Site VPN. With the help of the AWS team, we discovered that *if the traffic flow is short lived, the throttling mechanisms do not kick in fast enough.*

Slow gateway auto-scaling. Likewise, we observed that in many cases, even for the inbound traffic, we obtained very low throughput as shown in Fig. 2. With the help of the AWS team, we discovered that because of some of the elasticity mechanism configurations in AWS, auto-scaling of the gateways kicks in after at least 5 minutes of sustained high volume traffic going into the AWS cloud from GCP. These features were not documented at the time of our experimentation, and *we consistently obtained sub-optimal results for VPN-based connectivity*, as our experiments lasted less than 5 minutes. Following the recommendation of the AWS team, we extended the time of an experiment to be over 5 mins, as in:

```
iperf3 -Z -t 600 -i 0 -b 125M \
                          -c 10.1.1.6 -P 10
```

This allowed to obtain the 1.25 Gbps upper limit promised by the AWS Site-to-Site VPN quota.

Likewise, when exploring with the AWS customer support team why we do not obtain the upper limit on the AWS outbound traffic, we were able to collaboratively fix this issue on the AWS side.

Throughput in long VPN connections. The results of the long runs are shown in Fig. 3. For the intra-region case, the throughput of VPN connectivity is very close to 1.25 Gbps and the variance is very low. In the inter-region case, the throughput is a bit lower, because of the increased delay, as expected. The variance also becomes larger. It should be stressed that the data points of the measurements were collected at different days and at different hours of the day. In both cases, there are several outliers occasionally crossing the 1.25 Gbps threshold slightly (as could be expected from our preliminary experiments).

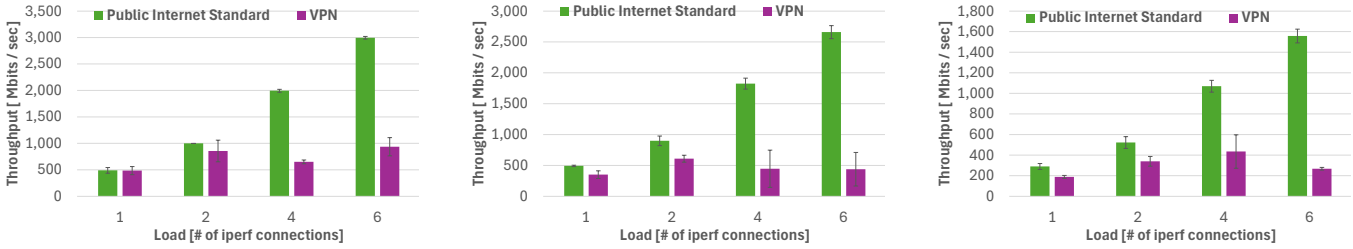
D. CCI Measurements

Fig. 4 shows throughput measurements for CCI and Public Internet (standard and premium tiers). As explained above, we only show CCI performance for 30%, 70%, and 100% link utilizations. However, we show more data points for the Public Internet options. We make the following observations.

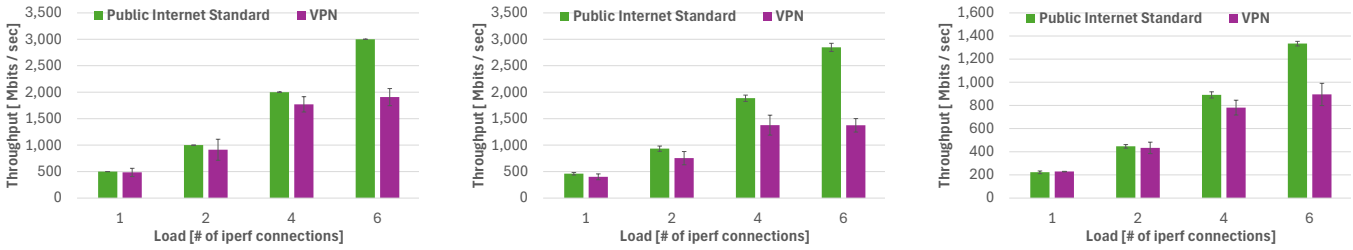
Guaranteed CCI capacity. At 100% utilization level, CCI attains its nominal throughput capacity as expected (minus 5% L2+L4 overhead) in both intra-region and intra-continent settings. In the inter-continent configuration, a larger delay causes throughput to drop consistently with the bandwidth-delay product.

Low CCI utilization. At the lower levels of utilization, CCI does not achieve better than standard or premium Internet.

[1]Google VPCs are multi-regional global resources comprised of regional private subnets.



(a) AWS Spain → GCP Spain (b) AWS Spain → GCP Poland (c) AWS Spain → GCP VA, USA



(d) GCP Spain → AWS Spain (e) GCP Poland → AWS Spain (f) GCP VA, USA → AWS Spain

Fig. 2. Throughput: VPN Service vs. Unencrypted Public Internet Standard

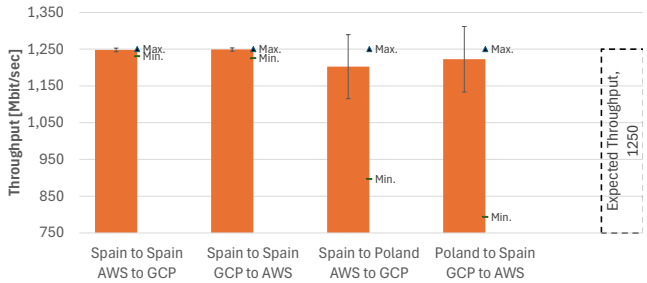


Fig. 3. Throughput in long VPN connections

Internet tier. Occasionally, Public Internet standard tier outperforms the premium one. This happens for example in the intra-continent setting when traffic is sent from Poland (GCP) to Madrid (AWS). The premium public Internet tier offering promises to carry traffic inside the GCP network and emit it at the closest Point of Presence (POP) of AWS to hand it to AWS. On the other hand, public standard Internet tier is emitted as soon as possible (i.e., Poland) and enters AWS at the closest POP in Central Europe. The traffic is then carried by the AWS network. Hence, the obtained result might reflect the relative speeds of the cloud vendors and peculiarities of routing. Of course, this phenomenon does not happen in the intra-region setting, because both vendors have presence in Madrid, Spain, and there are no routing options that would create an asymmetry between the standard and premium traffic. In this case, a user can save costs and receive similar service by using the cheaper standard tier rather than a more expensive premium one.

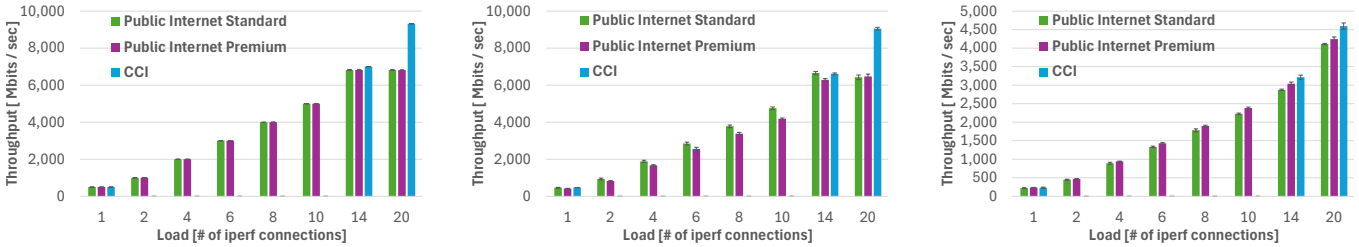
In addition, it appears that *the egress Public Internet is capped at* 7 *Gbps*, as the same NIC is capable of filling the 10 Gbps CCI link when we use CCI.

V. Model and Problem Formulation

In the previous section, we analyzed the performance characteristics of CCI. We now want to compare its cost against the VPN alternative. To do so, we develop an automated algorithm that dynamically decides when to establish or release either a dedicated physical CCI link or VPN tunnels, in order to minimize the total cost of serving the traffic demands. We start by formulating this as a formal problem.

Assumptions. We now attempt to formulate an optimization problem in the hypothetical case where an oracle knows all demands in advance. We later show that if future demands are unknown, there is no optimal decision, and thus we later offer a heuristic algorithm. As discussed in Section III, several practical considerations influence the cost model and its estimation: (1) The cost of using VPN is not constant; it follows a tiered pricing scheme in which the per-GB cost decreases with the total volume of data transferred. (2) Establishing a CCI link involves a provisioning delay of at least a few business days, denoted by the parameter D. From our hands-on experience with CCI, we assume $D = 72$ hours. To reflect practical management constraints, we also assume a fixed leasing period of $T_{\text{CCI}} = 1$ week that prevents turning off CCI right away. In contrast, we assume that a VPN tunnel can be both provisioned and released immediately. To highlight the fundamental differences between CCI and VPN, we focus on a simplified scenario in which all regions are located within the same continent and share a single CCI. As a result, the decision reduces to toggling between using VPN and CCI.

Problem framework. We consider a set of user pairs $\mathcal{P}$, where each pair $p \in \mathcal{P}$ consists of two VPCs located in different clouds: one in region r_1 of provider A and the other in region r_2 of provider B. Our goal is to minimize the total cost over a



(a) GCP Spain → AWS Spain (b) GCP Poland → AWS Spain (c) GCP VA, USA → AWS Spain

Fig. 4. Throughput: Public Internet Standard vs. Public Internet Premium vs. CCI

time horizon of T hours. At each hour, the cost includes two components: an hourly leasing cost and a per-GB data transfer cost. We decompose each cost component into per-pair costs.

$$\min \sum_{p\in\mathcal{P}} \sum_{t=1}^{T} \Big(\text{leasing cost}(p,t) + \text{transfer cost}(p,t)\Big) \quad (1)$$

CCI costs. If p uses CCI at time t, its leasing cost is $\frac{L_{\text{CCI}}}{P^t} + V^p_{\text{CCI}}$, where L_{CCI} is the shared cost of activating CCI, P^t is the number of pairs sharing CCI at time t, and V^p_{CCI} is the per-pair cost for a VLAN attachment. The transfer cost is $d^{p,t} \cdot c^p_{\text{CCI}}$, where $d^{p,t}$ is the data volume in GB, transferred by p at time t and c^p_{CCI} is the per-GB transfer cost over CCI.

VPN costs. If p uses VPN instead, its leasing cost is L^p_{VPN}, a fixed amount per pair, and the transfer cost is $d^{p,t} \cdot c^{p,t}_{\text{VPN}}$. The VPN transfer cost $c^{p,t}_{\text{VPN}}$ depends on the accumulated volume from the beginning of the month, so we assume a function $f(p, \sum_{t'=\text{start of month}}^{t} d^p_{t'})$ that returns the current per-GB rate for each pair p based on its total volume to date.

Problem formulation. Eq. (1) becomes

$$\min_{\{x_t\}} \sum_{t=1}^{T} \Bigg[x_t \cdot \Bigg(L_{\text{CCI}} + \sum_{p\in\mathcal{P}} \big(V^p_{\text{CCI}} + c^p_{\text{CCI}} \cdot d^{p,t}\big) \Bigg) + (1-x_t) \cdot \sum_{p\in\mathcal{P}} \big(L^p_{\text{VPN}} + c^{p,t}_{\text{VPN}} \cdot d^{p,t}\big) \Bigg] \quad (2)$$

The variable x_t indicates whether the CCI link is active at time t. We assume that when CCI is active ($x_t = 1$), all pairs use CCI instead of VPN.

VI. Algorithm

In this section, we present a heuristic algorithm to solve the problem defined in § V when future demands are unknown.

Comparison with ski-rental approach. The *ski-rental problem* is a foundational online problem that models the trade-off between an ongoing rental cost and a one-time purchase cost, under uncertainty about the future [44], [45]. Formally, a skier rents skis at cost r per day or buys them outright for B, but the number of skiing days d is unknown. The optimal deterministic algorithm rents for B/r days, then buys, achieving a competitive ratio of 2. Our setting differs in several ways: (1) Leasing CCI ("buying") is temporary, and the user can switch back to VPN ("renting"). (2) The cost structure includes multiple interacting components (e.g., fixed lease, per-unit transfer, and tiered costs), unlike the simple rent-or-buy setup. (3) There is a built-in provisioning delay D between the decision and activation of the resource.

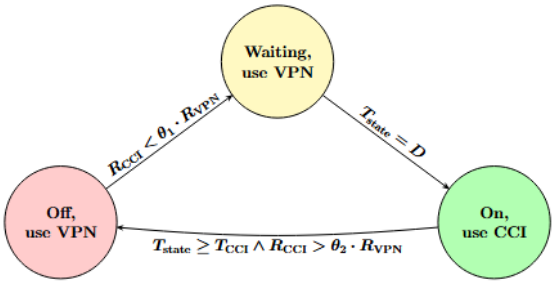


Fig. 5. TOGGLECCI's state diagram.

TOGGLECCI. We present an online algorithm, TOGGLECCI, that dynamically decides when to activate a dedicated CCI link, based on recent demand and cost trends. It maintains a sliding window of length h to track the aggregated cost of using VPN, denoted R_{VPN}, and the cost of using CCI, denoted R_{CCI}. Its behavior is governed by a finite-state process, illustrated in Fig. 5. It switches between three states, OFF, WAITING, and ON, using two thresholds, θ_1 and θ_2. During the initial time steps $t \in [0, h)$, TOGGLECCI uses the cumulative cost from the past t steps only. For our evaluations, we set $T_{\text{CCI}} = h = 168$ hours (one week).

In the OFF state (which is the initial state), all traffic is routed via VPN. The algorithm remains in this state as long as the observed cost of CCI is not significantly lower than the VPN cost. Once the aggregated CCI cost becomes attractive (specifically, when $R_{\text{CCI}} < \theta_1 \cdot R_{\text{VPN}}$), the algorithm requests CCI provisioning, and transitions into the WAITING state.

While in the WAITING state, traffic continues over VPN during the provisioning delay period. The algorithm remains in this state until the delay D elapses, after which it transitions to the ON state and begins routing traffic over CCI. We denote by T_{state} the time elapsed since TOGGLECCI entered its current state. In the ON state, CCI remains active for at least T_{CCI} time units. After this minimum commitment period, the algorithm continues using CCI as long as it remains cost-effective.

Specifically, it stays in the ON state while $R_{\text{CCI}} \leq \theta_2 \cdot R_{\text{VPN}}$. Once $R_{\text{CCI}} > \theta_2 \cdot R_{\text{VPN}}$, CCI is deactivated and the algorithm transitions back to the OFF state, reverting to VPN. This approach allows TOGGLECCI to toggle between VPN and CCI based on current cost dynamics, while incorporating real-world constraints such as provisioning delay. The use of two distinct thresholds, θ_1 for initiating CCI provisioning and θ_2 for renewal, with $\theta_1 < \theta_2$ ensure stability and prevent oscillation between states, while a single equal threshold can lead to more oscillations, as a well-known hysteresis phenomenon. In the evaluations, we set $\theta_1 = 0.9$ and $\theta_2 = 1.1$, reflecting a conservative policy that avoids premature activation of CCI and tolerates mild cost fluctuations.

Theoretical properties. We show that TOGGLECCI is near-optimal at consistently high or low rates.

Property 1. *(i) If traffic rates remain low, i.e., below the activation threshold of* TOGGLECCI*, then* TOGGLECCI *achieves optimal cost.*
(ii) If traffic rates remain high, i.e., above the activation condition of TOGGLECCI*, then* TOGGLECCI *achieves asymptotic optimal cost.*

Proof. (i) If traffic rates are sufficiently low, then over any sliding window of length h, $\theta_1 \cdot R_{\text{VPN}} < R_{\text{CCI}}$, so TOGGLECCI never transitions to the WAITING or ON state. Since the optimal offline algorithm faces the same cost comparison and finds no benefit in activating CCI, it also remains with VPN. Thus, the behavior and costs are identical.

(ii) Assume that for every time t, the aggregated cost over the past h steps satisfies $R_{\text{CCI}} < \theta_1 \cdot R_{\text{VPN}}$. This ensures that the algorithm transitions from the OFF state to the WAITING state. After a provisioning delay of D steps, CCI becomes available. If traffic remains high, the condition $R_{\text{CCI}} < \theta_2 \cdot R_{\text{VPN}}$ is also satisfied in all future windows, ensuring that the algorithm remains in the ON state indefinitely, just like the optimal offline solution. However, due to the provisioning delay D and the initial time required until the condition for θ_1 is met, the algorithm continues to use VPN. Thus, the only cost difference between the algorithm and the offline optimum arises during this transition period. Let

$$\gamma = \sum_{t=0}^{h+D-1} \sum_{p \in \mathcal{P}} \left(d^{p,t} \cdot (c_{\text{VPN}}^{p,t} - c_{\text{CCI}}^{p}) + L_{\text{VPN}}^{p} - \frac{L_{\text{CCI}}}{|\mathcal{P}|} - V_{\text{CCI}}^{p} \right).$$

γ represents the extra cost incurred by TOGGLECCI due to using VPN while waiting for the CCI provisioning to complete. After this delay, both the online and offline algorithms use CCI continuously, so their costs become identical. Therefore, the competitive ratio satisfies $\frac{\text{Cost}_{\text{TOGGLECCI}}}{\text{Cost}_{\text{OPT}}} \leq 1 + \frac{\gamma}{\text{Cost}_{\text{OPT}}}$. As traffic continues over a long time horizon, the additive gap γ becomes negligible relative to the total cost, and the algorithm becomes asymptotically optimal. □

We now show that unlike the famous competitive ratio of 2 in the ski rental problem [44], *there can be no constant competitive ratio* that is independent of the problem parameters in our online problem, no matter the algorithm.

Theorem 1. *For any constant $\alpha > 0$, there exists a traffic pattern and a set of cost parameters such that no online algorithm can guarantee a cost within a factor α of the optimal offline solution.*

Proof. Assume by contradiction that there exists an online algorithm $\mathcal{A}$ and a constant $\alpha > 0$ such that for any traffic pattern, the cost of $\mathcal{A}$ is at most α times the cost of the optimal offline algorithm. We construct an adversarial one-step scenario. At time $t = -D$, algorithm $\mathcal{A}$ must choose between using VPN or activating a CCI link, without knowing future demand. Thus, at time $t = 0$, either CCI is activated and we pay for it, or we use VPN.

- If $\mathcal{A}$ picks VPN, the adversary immediately injects at time $t = 0$ a large demand volume $d^{p,t} \gg 0$. The optimal choice would have been to use CCI (with a sufficient CCI capacity parameter), incurring cost $\frac{L_{\text{CCI}}}{P^k} + V_{\text{CCI}}^{p} + c_{\text{CCI}}^{p} \cdot d^{p,t}$, while $\mathcal{A}$ pays $L_{\text{VPN}}^{p} + c_{\text{VPN}}^{p,t} \cdot d^{p,t}$. As $d^{p,t} \to \infty$, the cost ratio becomes arbitrarily large:
$$\frac{\text{Cost}_{\mathcal{A}}}{\text{Cost}_{\text{OPT}}} = \frac{L_{\text{VPN}}^{p} + c_{\text{VPN}}^{p,t} \cdot d^{p,t}}{\frac{L_{\text{CCI}}}{P^k} + V_{\text{CCI}}^{p} + c_{\text{CCI}}^{p} \cdot d^{p,t}} \to \frac{c_{\text{VPN}}^{p,t}}{c_{\text{CCI}}^{p}} > \alpha$$
for suitable $d^{p,t}$, $c_{\text{VPN}}^{p,t}$, c_{CCI}^{p}.
- If $\mathcal{A}$ chooses to activate CCI, the adversary sends no traffic at $t = 0$. The optimal cost is zero, while $\mathcal{A}$ incurs a cost of at least L_{CCI}, and $\frac{\text{Cost}_{\mathcal{A}}}{\text{Cost}_{\text{OPT}}} = \infty > \alpha$.

In both cases, the adversary forces the cost ratio beyond any constant α, contradicting the assumption. □

VII. EVALUATION

A. Methodology

Costs. We design an evaluation environment to mimic the cost dynamics of cross-cloud data transfers between AWS, GCP and Azure. We use official pricing data from AWS [46], [47], GCP [38], [48] and Azure [49], [50] to get all tiered pricing components. Our model reflects the structure and pricing logic of real-world deployments, including transfer fees, VPN overheads, CCI leasing models, and volume-based cost thresholds. Although a full production setup includes additional components, such as load balancers and redundancy configurations, we focus here on the main components that distinguish between CCI and VPN.

Traffic. Each simulation scenario comprises multiple region pairs, where each pair represents a directional traffic flow between a region in GCP and a region in AWS or Azure, or vice versa. For this study, all regions are randomly selected within those of a single continent, defined as either Europe or the US. Traffic can be routed either through VPN or via CCI. Routing decisions are made hourly, as a higher frequency would not gain much given the 72-hour CCI provisioning delay. Traffic traces are either synthetic or derived from real-world measurements, as detailed below.

Algorithms. We compare the TOGGLECCI against four baselines: (1) ALWAYS-CCI, (2) ALWAYS-VPN, (3) AVG(ALL), which makes decisions based on the average traffic over the entire history, and (4) AVG(MONTH), which makes decisions based on the average traffic over the last month.

Workloads. There are no publicly-available workloads that specifically characterize cross-cloud transfers. Therefore, we leverage datasets from related domains and adapt them to our setting. We use four workloads to evaluate our algorithm. Two use real-world datasets with different traffic characteristics: MIRAGE-2019 [51], with bursty mobile app traffic, and PUFFER [52], with more stable video traffic. The other two are synthetic: one with a constant traffic rate, and one that models bursty traffic volumes.

B. MIRAGE real-world dataset

Description. The MIRAGE-2019 dataset [51] was collected at the University of Napoli between 2017 and 2019. It includes traffic logs from over 280 participants using rooted Android devices under realistic mobile usage.

Pre-processing. The raw dataset provides second-by-second traffic traces. In the preprocessing phase, we aggregate the original traffic data into hourly intervals, producing a continuous 2-year trace. We model a network with K users, each representing an individual mobile device. For each user, we generate a traffic trace by sampling from the MIRAGE dataset: Each day, we randomly select one of the available device traces and assign its hourly traffic volume to that user. We assume that each application server is deployed in a random distinct region within GCP, while users are randomly spread across same continent regions within AWS or Azure—and vice versa.

Results. Fig. 6 shows the total cost as a function of the number of users. Its results are similar to those on the synthetic traffic. TOGGLECCI consistently tracks the cheapest strategy across varying traffic phases, avoiding unnecessary leasing costs during low-traffic intervals and selectively activating CCI contracts during sustained bursts. Figs. 6(a) to 6(d) present all four evaluation settings: traffic sent from GCP to AWS and vice versa, in both Europe and the USA. While there are minor variations due to regional pricing and traffic direction, the overall trend is consistent across all scenarios. TOGGLECCI reduces the total cost by an average factor of $1.8\times$ compared to both strategies at breakeven traffic rates.

Fig. 7 decomposes the total cost for a representative case ($K = 100,000$ users) into leasing and traffic components, showing that TOGGLECCI achieves a balanced cost split.

GCP-Azure. Fig. 8 presents results for transfers between GCP and Azure in both directions, to evaluate TOGGLECCI's robustness. The principles of the cost model are similar to those in the GCP-AWS setting, with only minor differences in pricing [49]. The results are similar to the GCP-AWS case. This is because, although the absolute costs differ, the underlying principles remain the same.

Inter-Continental Scenario. In many practical settings, organizations operate multi-cloud resources across different continents, where inter-continental data transfer costs may be substantial. We consider a simple broadcast use case: a sender located in Europe (i.e., Paris) sends data from GCP to AWS regions distributed across both Europe and the United States. We assume that the colocation facility is given in this setting, and the user must decide whether to send the traffic through CCI at this colocation or through a VPN. We compare two configurations depending on the location of the colocation facility: (1) near the sender (in Paris), and (2) far from the sender (in Ohio). Of course, the egress cost of CCI differs significantly depending on the location of the colocation facility. When the facility is not located near the sender, traffic must first traverse the cloud provider's inter-continental backbone before reaching the CCI, thereby increasing the overall cost.

Fig. 9 shows that in both cases, whether the colocation is nearby or distant, TOGGLECCI adapts to price changes and remains cost-effective in inter-continental settings.

C. PUFFER real-world dataset

Description. The PUFFER dataset [53] was collected as part of a real-world video streaming experiment led by Stanford University. Its goal was to study how users interact with adaptive bitrate (ABR) video delivery over the internet. The dataset contains fine-grained, second-level traffic traces from users watching live and on-demand video through a custom ABR system. The PUFFER dataset provides realistic, continuous traffic patterns generated by users streaming video content. Compared to the bursty mobile app traffic of MIRAGE, it represents more stable, session-based load.

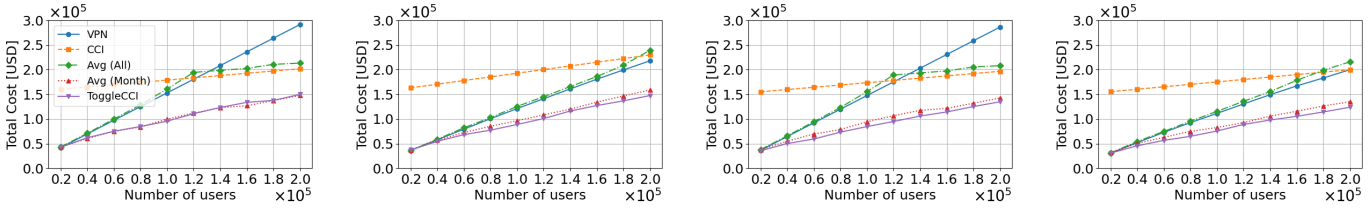
Pre-processing. Similarly to our processing of the MIRAGE dataset, we aggregate the data into hourly bins to match the time resolution of our evaluation. The dataset includes seven video channels, each assigned to a distinct region in Europe, with transfers directed from GCP to AWS.

Results. The PUFFER dataset is characterized by stable, session-based traffic with observable daily and weekly cycles. However, due to the overhead and setup time required to establish a CCI connection, TOGGLECCI cannot effectively exploit these periodic patterns. As a result, dynamic switching between transfer modes provides limited benefit, and the optimal choice between VPN and CCI primarily depends on the overall traffic volume.

Fig. 10(a) shows how TOGGLECCI effectively identifies the more cost-efficient option under these conditions. CCI emerges as the most efficient strategy for the PUFFER workload, and TOGGLECCI quickly aligns with it. In Fig. 10(b), we show the decomposition of total cost into leasing and traffic components. As expected, CCI dominates in leasing cost, while VPN dominates in traffic cost. In this scenario, TOGGLECCI closely tracks the cost behavior of CCI.

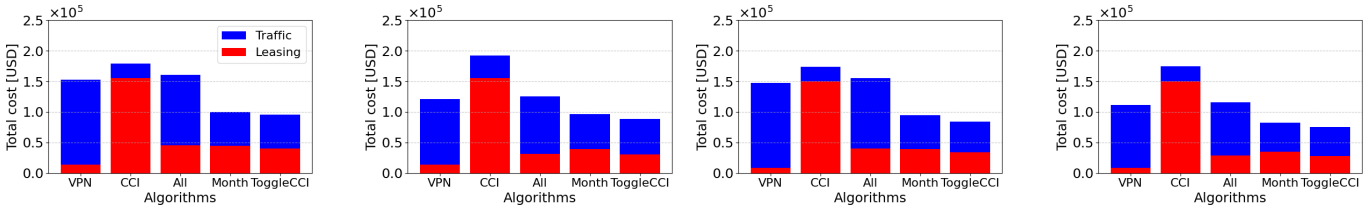
D. Sensitivity Analysis

Constant traffic. For additional insights and sensitivity analysis, we also evaluate TOGGLECCI on synthetic workloads. The first is a constant-rate workload, generated over a full year



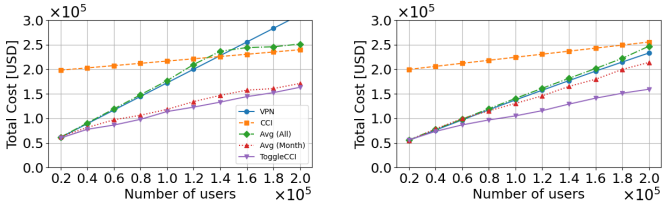
(a) EU: GCP → AWS (b) EU: AWS → GCP (c) USA: GCP → AWS (d) USA: AWS → GCP

Fig. 6. Performance comparison using the MIRAGE dataset. Cost as a function of the number of users, both in Europe and in the US, from GCP to AWS and vice versa. In all cases, TOGGLECCI is close to optimal.



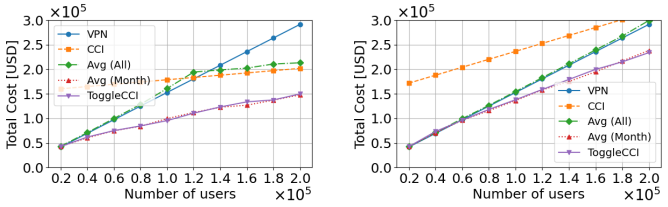
(a) EU: GCP → AWS (b) EU: AWS → GCP (c) USA: GCP → AWS (d) USA: AWS → GCP

Fig. 7. Cost breakdown for a specific scenario with 100,000 users on the MIRAGE dataset, in Europe and in the US, from GCP to AWS and vice versa. Results are roughly similar in all scenarios. TOGGLECCI uses a balanced cost split.



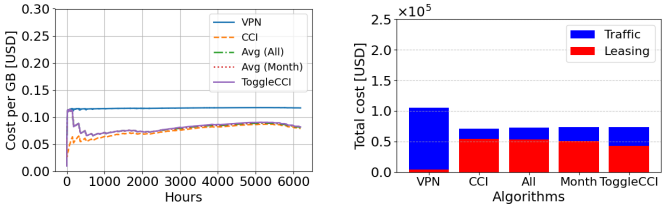
(a) EU: GCP → AZURE (b) EU: AZURE → GCP

Fig. 8. Performance comparison using the MIRAGE dataset. Cost as a function of the number of users, from GCP to AZURE and vice versa.



(a) Colocation in Europe (b) Colocation in the US

Fig. 9. Inter-continental data transfer cost comparison under different CCI colocation placements (Europe vs. US).



(a) Cumulative average cost per GB (b) Total cost

Fig. 10. Evaluation with the PUFFER dataset. TOGGLECCI tends to stick with CCI due to the high traffic volume.

using hourly time steps (i.e., 8,760 hours). This trace provides initial intuition by fixing the traffic volume at every time step. It corresponds to scenarios involving short recurring transfer cycles (e.g., hourly or daily batches for backups), which appear almost constant to TOGGLECCI.

Fig. 11 presents results for this constant trace. For low traffic rates, it is optimal to stick to VPN, while for higher rates, CCI is preferable. For low and high rates, TOGGLECCI quickly adopts the optimal choice and achieves the lowest possible cost, as stated in Property 1. When using CCI, it only misses the first D days due to the CCI setup time. Moreover, because the switching threshold is slightly below the breakeven point ($\theta_1 = 0.9$), TOGGLECCI remains conservative just below the breakeven point when CCI and VPN costs are equal.

Bursty traffic. We also evaluate a *bursty trace* for one year. It captures a more dynamic behavior. Burst arrivals follow a Poisson process. Burst durations and intensities are sampled from Gaussian distributions with configurable means. We start with an initial configuration that uses a Poisson arrival rate of $\lambda = 1/730$, corresponding to roughly one burst per month on average; a mean burst duration of about one week; and an average traffic intensity of 400 GB/hour. All results are averaged over 20 randomized repeats.

Fig. 12(a) shows the results for different values of mean burst intensity. As for the constant trace, for small traffic volumes, VPN is cheaper. For high volumes, CCI becomes the most cost-effective option. In the intermediate range, TOGGLECCI outperforms both static strategies by dynamically adapting to the observed traffic patterns. It selectively activates CCI during high-demand burst periods and falls back to VPN when demand drops. Fig. 12(b) shows the average cumulative cost per GB over time given a mean volume of 400 GB/hour. The VPN cost is near-constant, while CCI cost

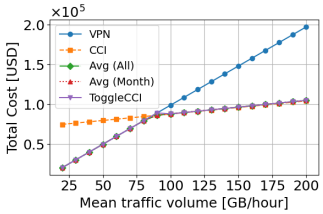
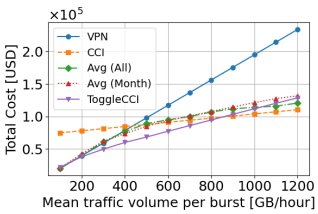
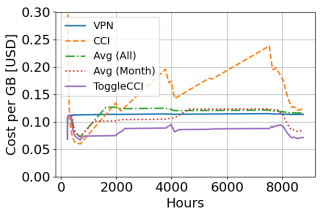


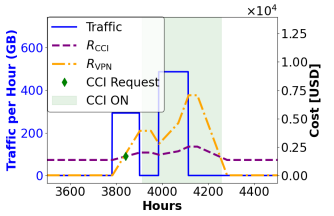
Fig. 11. Total cost for the trace with constant traffic rate as a function of rate. ToggleCCI is near-optimal.



(a) Total cost

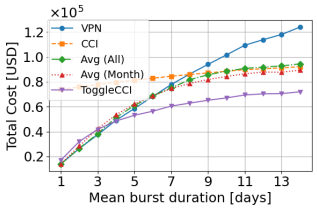


(b) Cumulative average cost per GB

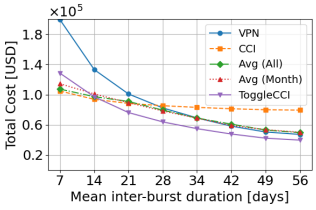


(c) Dynamic behavior

Fig. 12. Bursty trace. (a) compares total costs across different mean traffic volumes. Given a mean volume of 400 GB/hour, (b) shows ToggleCCI's ability to achieve the best cost per GB compared to the baselines, while (c) visualizes ToggleCCI's dynamic behavior over time, with a zoom-in between time 3500 and 4500.

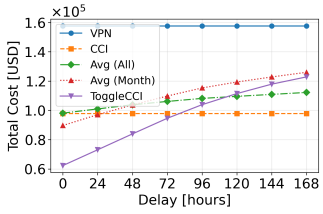


(a) Cost vs. burst duration

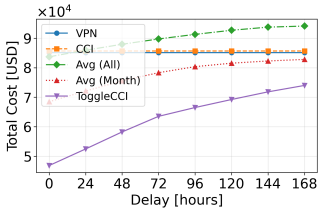


(b) Cost vs. inter-burst time $\frac{1}{\lambda}$

Fig. 13. Sensitivity analysis for the bursty traffic parameters. (a) varies the burst duration, given one burst per month on average. (b) varies the inter-burst interval, given a burst duration of 7 days on average.



(a) High traffic.



(b) Breakeven traffic.

Fig. 14. Sensitivity of ToggleCCI to the provisioning delay D. The plots show the total cost as a function of the provisioning delay under (a) high traffic and (b) mid-range traffic near the VPN/CCI breakeven point.

drops during bursts and rises in silent periods. ToggleCCI achieves the lowest cost by adaptively switching between the two. Fig. 12(c) illustrates the behavior of ToggleCCI over time. We observe that during the idle period (3500-3800), there is no traffic and thus $R_{\text{VPN}} = 0$, while R_{CCI} remains stable around 1500$, which corresponds to the cumulative CCI leasing cost over the sliding window h. Once the burst begins, R_{VPN} increases sharply due to rising traffic, whereas R_{CCI} grows only moderately. ToggleCCI does not activate CCI until the condition $R_{\text{CCI}} < \theta_1 \cdot R_{\text{VPN}}$ is satisfied. The actual activation occurs after a provisioning delay of D hours. After activation, CCI remains On for T_{CCI} hours, and is renewed if the stay condition still holds. Otherwise, CCI is deactivated.

Fig. 13 presents a sensitivity analysis for bursty traffic. In Fig. 13(a), we vary the burst duration while keeping the average frequency fixed at one burst per month. When the duration is low, ToggleCCI is more expensive than VPN due to the fixed contract duration time T_{CCI}. However, when the duration exceeds D days, ToggleCCI becomes more cost-effective than VPN. In Fig. 13(b), we fix the average burst duration to 7 days and vary the interval between bursts. In this scenario, ToggleCCI outperforms VPN for any value of λ because the duration is sufficiently long. When the interval between bursts becomes short, CCI performs the best due to the limitation of D. Once the time between bursts becomes large (around 21 days), ToggleCCI outperforms both.

We next examine the robustness of ToggleCCI by varying the provisioning delay D. Fig. 14(a) presents a high-traffic scenario, where CCI is cheaper than VPN. In this regime, ToggleCCI outperforms the static policies when the provisioning delay is short, since the algorithm can respond quickly to traffic changes and activate CCI in time. However, when the delay becomes large, ToggleCCI reacts slowly, and Always-CCI becomes a better option. Fig. 14(b) shows the same experiment for the breakeven scenario, where VPN and CCI have similar costs. In this case, ToggleCCI remains beneficial even under relatively long provisioning delays.

VIII. Conclusion

We presented the first comprehensive study of CCI, a high-throughput option for inter-cloud transfers. While CCI offers low per-GB costs, it uses fixed leasing fees and takes a few days to establish. We conducted a methodical study of CCI performance and compared it to other connectivity options. To the best of our knowledge these results are first of a kind and they provide some valuable insights into the actual (also undocumented) behavior of cross-cloud connectivity. We further proposed ToggleCCI, an online algorithm that dynamically switches between VPN and CCI based on recent cost trends. We proved that ToggleCCI achieves near-optimal performance in persistent high or low demand regimes. Evaluations on synthetic and real-world traces show that ToggleCCI closely tracks the best static choice and significantly reduces total cost in practice.

Several directions remain for future work. In particular, while we evaluate a single VPN tunnel as a representative baseline, real deployments may use more advanced networking solutions, with more complex cost considerations.

Acknowledgments. This work was partly supported by the Louis and Miriam Benjamin Chair in Computer-Communication Networks.

REFERENCES

[1] P. Jain and et al., "Skyplane: Optimizing transfer cost and throughput using cloud-aware overlays," in *USENIX NSDI*, 2023, pp. 1375–1389.

[2] Matan Avneri, "Optimizing Data Transfer Costs: Top Strategies to Save Big and Improve Performance," 2025. [Online]. Available: https://seemoredata.io/blog/data-transfer-costs/

[3] Z. Yang and et al., "Skypilot: An intercloud broker for sky computing," in *USENIX NSDI*, 2023, pp. 437–455.

[4] Lucidity, "Multi-cloud cost optimization: A complete guide," 2024, accessed: 2025-07-30. [Online]. Available: https://www.lucidity.cloud/blog/multi-cloud-cost-optimization

[5] S. M. Data, "Snowflake data transfer costs: Everything you need to know," 2024, accessed: 2025-07-30. [Online]. Available: https://seemoredata.io/blog/snowflake-data-transfer-costs/

[6] Contrail, "Cloud-native contrail networking datasheet," 2024. [Online]. Available: https://www.juniper.net/us/en/products/sdn-and-orchestration/contrail/cloud-native-contrail-networking-datasheet.html

[7] Amazon Web Services, "How Amazon VPC works," 2024. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html

[8] Google Cloud, "Virtual Private Cloud (VPC)," 2023. [Online]. Available: https://cloud.google.com/vpc?hl=en

[9] IBM Cloud, "What is a virtual private cloud (VPC)?" 2024. [Online]. Available: https://cloud.ibm.com/docs/overview?topic=overview-endpoints-support

[10] Microsoft, "What is Azure Virtual Network?" 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview

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

[12] ——, "Announcing Cross-Cloud Interconnect: Connect Google Cloud to other cloud providers," https://cloud.google.com/blog/products/networking/announcing-google-cloud-cross-cloud-interconnect, May 2023.

[13] ——, "Extending Cross-Cloud Interconnect to AWS and partners," https://cloud.google.com/blog/products/networking/extending-cross-cloud-interconnect-to-aws-and-partners, Nov. 2025.

[14] Microsoft Learn, "Connect to public cloud services using azure expressroute," https://learn.microsoft.com/en-us/azure/expressroute/expressroute-connect-azure-to-public-cloud, 2024.

[15] Amazon Web Services, "AWS direct connect - VPC connectivity options," 2024, accessed: 2025-07-30. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/aws-direct-connect.html

[16] Google Cloud, "Dedicated interconnect overview," 2024, accessed: 2025-07-30. [Online]. Available: https://cloud.google.com/network-connectivity/docs/interconnect/concepts/dedicated-overview

[17] Oracle, "Oracle Database@Azure," https://www.oracle.com/cloud/azure/oracle-database-at-azure/.

[18] Microsoft Azure, "Oracle on Azure," https://azure.microsoft.com/en-us/solutions/oracle.

[19] Aviatrix, "Aviatrix secure cloud networking products," 2024. [Online]. Available: https://aviatrix.com/secure-cloud-networking-products/

[20] Megaport, "Megaport: Global network as a service provider," 2025, accessed: 2025-07-30. [Online]. Available: https://www.megaport.com/

[21] Fortinet, "Secure SD-WAN for multi-cloud infrastructure," 2024. [Online]. Available: https://www.fortinet.com/solutions/enterprise-midsize-business/sd-wan-multi-cloud

[22] C. Plewnia and H. Lichter, "Cross-provider cloud cost calculation," *arXiv preprint arXiv:2301.XXXX*, 2023.

[23] T. Aoshima and K. Yoshida, "Pre-design stage cost estimation for cloud services," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2020, pp. 61–66.

[24] E. Eliav *et al.*, "ToggleCCI code," https://github.com/eitttan/ToggleCCI.

[25] N. Kandregula, "Evaluating performance and scalability of multi-cloud environments: Key metrics and optimization strategies," *World Journal of Advanced Research and Reviews*, 2022.

[26] C. Rista, M. Teixeira, D. Griebler, and L. G. Fernandes, "Evaluating, estimating, and improving network performance in container-based clouds," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 00 514–00 520.

[27] Y. Ben-Itzhak, A. Bergman, I. Cidon, I. Golikov, A. Markuze, N. H. Rotman, and E. Zohar, "Cloudcast: Characterizing public clouds connectivity," *ArXiv*, vol. abs/2201.06989, 2022.

[28] R. K. P. Mok, H. Zou, R. Yang, T. Koch, E. Katz-Bassett, and K. C. Claffy, "Measuring the network performance of Google cloud platform," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21, 2021, p. 54–61.

[29] X. Tang, "Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2909–2919, 2021.

[30] Q.-H. Zhu, H. Tang, J.-J. Huang, and Y. Hou, "Task scheduling for multi-cloud computing subject to security and reliability constraints," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, pp. 848–865, 2021.

[31] The Linux Foundation, "Paraglider," 2024. [Online]. Available: https://paragliderproject.io

[32] K. Toledo, D. Breitgand, D. Lorenz, and I. Keslassy, "CloudPilot: Flow acceleration in the cloud," *Computer Networks*, vol. 224, p. 109610, 2023.

[33] Spot by NetApp, "Spot by netapp – optimize your cloud infrastructure costs," 2024. [Online]. Available: https://spot.io

[34] CAST AI, "CAST AI – Kubernetes cost optimization and automation," 2024, accessed: 2025-07-28. [Online]. Available: https://cast.ai

[35] CloudHealth by VMware, "Cloudhealth – multi-cloud cost management and optimization," 2024, accessed: 2025-07-28. [Online]. Available: https://www.vmware.com/products/cloudhealth.html

[36] Google Cloud, "Order AWS ports for Cross-Cloud Interconnect," 2024, accessed: 2025-07-31. [Online]. Available: https://cloud.google.com/network-connectivity/docs/interconnect/how-to/cci/aws/order-aws-ports#order_your_ports

[37] ——, "VLAN attachments," 2025. [Online]. Available: https://cloud.google.com/network-connectivity/docs/interconnect/how-to/dedicated/creating-vlan-attachments

[38] ——, "CCI pricing catalog," https://cloud.google.com/network-connectivity/docs/interconnect/pricing#cci-pricing.

[39] Microsoft, "VPN gateway security baseline," 2025. [Online]. Available: https://learn.microsoft.com/en-us/security/benchmark/azure/baselines/vpn-gateway-security-baseline?toc=%2Fazure%2Fvpn-gateway%2FTOC.json

[40] IBM Cloud, "Getting started with VPN for VPC," 2025. [Online]. Available: https://cloud.ibm.com/docs/iaas-vpn?topic=iaas-vpn-getting-started

[41] Amazon Web Services, "Aws site-to-site vpn," 2025. [Online]. Available: https://aws.amazon.com/vpn/

[42] Google Cloud, "Cloud VPN Overview," 2025. [Online]. Available: https://cloud.google.com/network-connectivity/docs/vpn

[43] AWS, "AWS VPN Quotas," https://docs.aws.amazon.com/vpn/latest/s2svpn/vpn-limits.html.

[44] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator, "Competitive snoopy caching," *Algorithmica*, vol. 3, no. 1, pp. 79–119, 1988.

[45] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, "Competitive randomized algorithms for nonuniform problems," *Algorithmica*, vol. 11, no. 6, pp. 542–571, 1994.

[46] AWS, "AWS pricing catalog," https://aws.amazon.com/ec2/pricing/on-demand/.

[47] ——, "AWS direct connect pricing catalog," https://aws.amazon.com/directconnect/pricing/.

[48] Google Cloud, "GCP premium tier pricing catalog," https://cloud.google.com/vpc/network-pricing?hl=en#internet_egress.

[49] Microsoft Azure, "Azure ExpressRoute Pricing," https://azure.microsoft.com/en-us/pricing/details/expressroute/.

[50] ——, "Azure VPN Pricing," https://azure.microsoft.com/en-us/pricing/details/vpn-gateway/.

[51] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapè, "Mirage: Mobile-app traffic capture and ground-truth creation," in *IEEE 4th International Conference on Computing, Communication and Security (ICCCS 2019)*, Oct 2019.

[52] Puffer Dataset, https://puffer.stanford.edu/data-description/.

[53] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *USENIX NSDI*, Santa Clara, CA, Feb. 2020, pp. 495–511.