Cost-Effective Cloud Edge Traffic Engineering with CASCARA

Rachee Singh Sharad Agarwal Matt Calder Paramvir Bahl
Microsoft

Abstract

Inter-domain bandwidth costs comprise a significant amount of the operating expenditure of cloud providers. Traffic engineering systems at the cloud edge must strike a fine balance between minimizing costs and maintaining the latency expected by clients. The nature of this tradeoff is complex due to *non-linear* pricing schemes prevalent in the market for inter-domain bandwidth. We quantify this tradeoff and uncover several key insights from the link-utilization between a large cloud provider and Internet service providers. Based on these insights, we propose CASCARA, a cloud edge traffic engineering framework to optimize inter-domain bandwidth allocations with non-linear pricing schemes. CASCARA leverages the abundance of *latency-equivalent* peer links on the cloud edge to minimize costs without impacting latency significantly. Extensive evaluation on production traffic demands of a commercial cloud provider shows that CASCARA saves 11–50% in bandwidth costs per cloud PoP, while bounding the increase in client latency by 3 milliseconds[1].

1 Introduction

Cloud wide-area networks (WANs) play a key role in enabling high performance applications on the Internet. The rapid rise in traffic demands from cloud networks has led to widespread adoption of centralized, software-defined traffic engineering (TE) systems by Google [19] and Microsoft [17] to maximize traffic flow *within* the cloud network.

In the quest to overcome BGP's shortcomings, recent efforts have focused on engineering *inter*-domain traffic, which is exchanged between the cloud WAN and other networks on the Internet [27, 33]. These systems can override BGP's best-path selection, to steer egress traffic to better performing next-hops. However, this focus on performance overlooks a crucial operating expenditure of cloud providers: the *cost* of inter-domain traffic determined by complex pricing schemes. While the prices of inter-domain bandwidth have declined in the past decade, the decrease has been outpaced by exponential growth in demand [29] from cloud networks serving high-definition video, music and gaming content. In fact, the inter-domain bandwidth costs incurred by the cloud provider we analyze increased by 40% in the March 2020 billing cycle as a consequence of the increase in demand fueled by work from home guidelines in various parts of the world. [2]

[1] Code and experiments at: http://cascara-network.github.io.

[2] We do not disclose the fraction of total cloud operation expenditure contributed by inter-domain bandwidth costs due to confidentiality reasons.

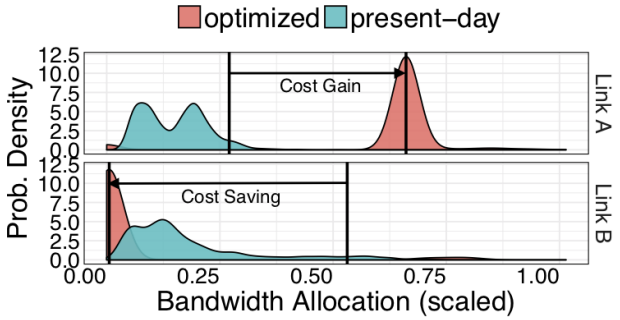


Figure 1: Present-day and CASCARA-optimized bandwidth allocation distributions for one week, across a pair of links between a large cloud provider and tier-1 North American ISPs. Costs depend on the 95th-percentile of the allocation distributions (vertical lines). CASCARA-optimized allocations reduce total costs by 35% over the present-day allocations while satisfying the same demand.

In this work, we show that recent increases in interconnection and infrastructure scale enable significant potential to reduce the costs of inter-domain traffic. These advances include the deployment of several new cloud points of presence (PoP) near clients and direct peering with an increasing fraction of the Internet's autonomous systems [5]. As a result, most clients are reachable over several short and *latency-equivalent* paths from the cloud provider [26]. We illustrate the cost saving potential due to latency-equivalent links with an example in Figure 1. We plot the distributions of bandwidth allocated over one week to links A and B, which connect a large cloud provider to tier-1 North American ISPs. Both links are located at inter-connection points within 30 km of each other, and offer comparable latency due to their geographical proximity. In this example, the bandwidth price per Mbps of Link B is 33% higher than that of Link A. Link costs are a function of the 95th percentile of the bandwidth allocations to each link. The *present-day* allocations (in blue) represent the current bandwidth assigned to the links by the cloud provider under study. In contrast, the CASCARA-*optimized* allocations (in red) meet the same or higher demand as the present-day allocations, while reducing total bandwidth costs by 35%.

Bandwidth allocations at the cloud edge impact both the client latency and inter-domain bandwidth costs to the cloud provider. At one extreme, traffic allocations may disregard the latency impact to drive bandwidth costs to near-zero while at the other extreme, allocations may incur very high bandwidth costs by greedily assigning traffic to the lowest latency peers. Balancing this *cost-latency tradeoff* is central to our work. However, it is made challenging by industry-standard pricing schemes that use 95th percentile of the bandwidth dis-

tribution over monthly time-periods. Complex relationships between bandwidth allocations, costs and client latency lead to computationally hard optimization problems.

We tackle these challenges by first analyzing the utilization of edge links from a large commercial cloud provider. We find that the majority of traffic from the cloud is exchanged with transit ISPs, with outbound traffic being twice in volume compared to inbound traffic. Thus, outbound traffic to transit ISPs dominates the inter-domain bandwidth costs of the cloud. Three such North American ISPs incur a majority of the total expenditure on inter-domain bandwidth in the continent (§3). Using these insights, we make three main contributions:

1. Quantify the opportunity of saving bandwidth cost. We formulate cloud edge TE as an optimization with the goal of minimizing percentile bandwidth costs. Despite the non-convex nature of the objective, the optimization is tractable in engineering outbound traffic to peer links with only the small number of ISPs that contribute majority of the costs. We show that cost-optimal allocations can **save up to 65% of the cloud provider's inter-domain bandwidth costs**, quantifying the upper bound on savings (§3) and offering a significant improvement over related approaches in [12, 20, 35].

2. Practical and cost-efficient online edge TE. Since optimizing percentile costs is NP-Hard [20], finding optimal solutions can take several hours. We leverage insights from the offline optimal solution to design an efficient, heuristic-based online TE framework, CASCARA. CASCARA leverages the cloud provider's rich diversity of latency-equivalent BGP peers to offer cheaper options to outbound traffic. Through extensive experiments we demonstrate that CASCARA provides near-optimal cost saving in practice and can be deployed **safely** and **incrementally** in cloud WANs (§4).

3. Flexibility to balance the cost-latency tradeoff. CASCARA incorporates the latency of primary and alternate peer paths from the cloud [4, 27] to strike a balance between bandwidth cost savings and client latency. CASCARA provides the flexibility to pick the operating point on this tradeoff and finds allocations that bound the increase in client latency by 3 ms while saving 11-50% of bandwidth costs per cloud PoP (§5).

Client latency requirements vary based on the types of application traffic, *e.g.,* software updates and large file transfers are more delay tolerant than live video. In fact, majority of all outbound traffic from the cloud provider is marked as best-effort, making it tolerant to small changes in latency. We conclude this study by discussing the generalizability of our results, the implications of CASCARA on peering contracts and bandwidth pricing models on the Internet (§6).

2 CASCARA controller overview

CASCARA's goal is to engineer outbound traffic allocations from the cloud edge to achieve near-optimal saving in inter-domain bandwidth costs. It does so by providing operational safety knobs to the operator: configurable variation in the

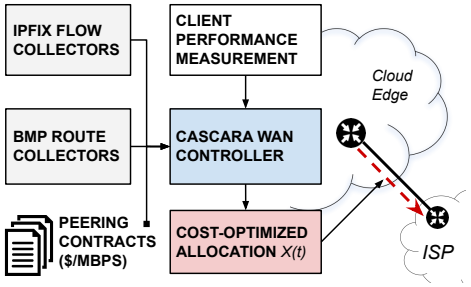


Figure 2: The design of CASCARA.

traffic allocations to peer links, incremental deployability and bounded impact on client latency. Figure 2 shows the different components of CASCARA. At the core is the CASCARA WAN controller that allocates cost-optimized flow to outbound peer links of the cloud network.

IPFIX Flow Collectors. We feed IP Flow Information Export (IPFIX) [31] logs to CASCARA to infer the utilization of edge links of the cloud network in five minute intervals of the billing cycle. These allocations to peer links are used both for offline cost analysis (§3) and online allocation to meet demands by CASCARA (§4 and §5).

BMP Route Collectors. We gather route announcements made by BGP peers at points of presence (PoP) of the cloud provider using BGP Monitoring Protocol (BMP) collectors. These routes inform CASCARA of the choices of peer links for outbound demand towards clients.

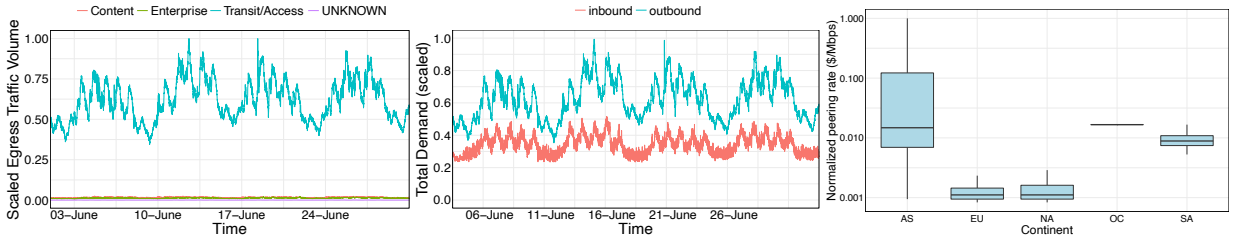
Peering Contracts. We feed the billing models and peering rates for all BGP peers of the cloud provider to CASCARA. Since peering rates remain stable over short durations of time, we use snapshot of this information from June 2019.

Client latency measurements. CASCARA makes latency-aware decisions limiting the impact of outbound traffic allocation on client latency. We feed CASCARA the median latency to all clients of the cloud provider over both the primary and alternate BGP paths at the PoPs.

Cloud providers have developed software-defined edges for fine-grained control of outbound path selection from their networks [27, 33]. These systems provide the infrastructure to steer outbound traffic to desired peer paths. The CASCARA controller allocates flow to peer links in every 5 minute interval and can leverage the software-defined edge to optimize the inter-domain bandwidth costs. We first quantify the potential of bandwidth cost saving in a large cloud provider (§3), then develop an efficient, online and near-optimal algorithm for CASCARA to realize the saving potential (§4). Finally, we put CASCARA to test with realistic client performance and route availability constraints in §5.

3 Quantifying the Opportunity

Cloud networks occupy a central position in the Internet ecosystem due to the large volume and variety of popular content they serve to users. To make this possible, cloud providers



(a) Outbound traffic types. (b) Inbound vs. Outbound traffic. (c) Norm. peering rates by continent.

Figure 3: (a) Outbound traffic from a large cloud provider towards BGP peers of different types; majority of outbound traffic is towards transit/access networks. (b) The distribution of inbound vs. outbound traffic volume from the cloud network. (c) Large differences in the cost per unit bandwidth in different parts of the world *e.g.,* the median peering cost in Asia is over 10X the median peering cost in North America.

peer with a large number of networks or Autonomous Systems (ASes) on the Internet, including transit ISPs and eyeball networks. The cloud provider we analyze has over 7,000 BGP peers, including transit networks, access networks, content providers and Internet Exchange Points (IXPs). These links span over one hundred geographical locations, collectively carrying terabits of traffic per second. We analyze the utilization and bandwidth costs incurred at the peering edge of the commercial cloud provider using IPFIX flow records collected from June 2018 to July 2019. Aggregated across all edge links, Figure 3a shows the outbound traffic volume per five minute interval from the cloud towards transit/access networks, cloud providers and enterprise networks, categorized by CAIDA's AS types classification [3].

3.1 Dominant contributors to bandwidth cost

A BGP peer of the cloud network charges for the traffic exchanged between them according to the billing model negotiated in their *peering contract*. There are three billing models for inter-domain traffic prevalent on the Internet today: (1) Settlement-free (2) Per-port and (3) Per-Megabit [9]. Settlement-free peers (SFP) agree to exchange traffic with each other at no cost (*e.g.,* between cloud providers). In per-port peering, a peer bills another for each network port used at their facility (*e.g.,* connections at IXPs). Per-Megabit is a utilization-based model where a network charges its peer based on the utilization of the link between them over monthly billing cycles. There can be a commit clause in this contract *i.e.,* regardless of the actual usage, the customer *commits* to pay at least some pre-arranged amount to the provider.

Utilization-based, per-megabit billing is the industry standard for paid peer and transit ISP contracts and it is the focus of our work. Our goal is to minimize bandwidth costs accrued on peering links billed by their utilization. To translate network utilization into the corresponding inter-domain bandwidth cost, ISPs measure the average utilization of peering links in five minute intervals in both inbound and outbound directions. Let the edge link from peer p_1 to peer p_2 have average outbound utilizations of $B = \{B_1, B_2, .., B_n\}$ megabits in 5-minute intervals of a given month. Let B_{out} be the 95^{th} percentile of the outbound utilizations, B. Similarly, B_{in} is the 95^{th} percentile of average inbound utilizations of the $p_1 - p_2$ link. The link cost for a billing cycle is given by, $B = c_i \cdot \text{MAX}\{B_{out}, B_{in}\}$, where c_i is the peering rate negotiated by p_1 and p_2 as part of their peering agreement. This model of billing bandwidth, also called *burstable billing*, has evolved as an industry standard on the Internet [9].

Bulk of the traffic is exchanged with Transit/Access ISPs. The large majority of traffic at the cloud edge is outbound to Transit/Access networks (Figure 3a). Therefore, traffic exchanged with transit ISPs is the main contributor to bandwidth costs incurred by the cloud provider.

Outbound traffic is twice the inbound. For the cloud WAN, outbound traffic volume is nearly twice the inbound (Figure 3b), highlighting that the cost computation based on link utilizations can be simplified to $c_i \cdot B_{out}$ for clouds networks.

Links with only three ISPs contribute majority of costs. Due to the large variance in peering rates (seen in Figure 3c) and skewed distribution of traffic towards a few large ISPs in the North American region of the cloud, edge links to three large networks incur a majority of the total spend on inter-domain bandwidth in North America.

3.2 Optimal inter-domain bandwidth costs

In this section we formalize the task of optimizing inter-domain bandwidth costs of a cloud network. As outbound traffic to paid peers is significantly higher than inbound (Figure 3b), we focus on engineering outbound traffic to minimize the overall inter-domain bandwidth cost. To quantify the potential cost savings, we formulate the offline version of the problem where traffic demands are known in advance.

Let $L = \{l_1, l_2, ..l_m\}$ be the set of all edge links from the WAN. Edge links to the same peer at different points of presence (PoP) are billed individually according to their percentile utilization. Let a five-minute interval in the monthly billing period be t_j where $j \in \{1, 2, .., n\}$. For instance, the month of January has $8,928$ five-minute intervals.

Decision variables. The traffic allocation scheme assigns network flow to peering links in L, for every time slot $t_j, j \in [1,..,n]$. Let x_{ij} be the decision variable, where x_{ij} is the flow assigned to peering link l_i in time slot t_j.

Objective function. The goal of our allocation scheme is to find a traffic assignment to edge links over the entire billing period such that the total inter-domain bandwidth cost is minimized. The cost incurred on peering link l_i is the product of the peering rate (c_i) and the 95^{th} percentile utilization of that link (denoted by z_i). The goal is to minimize the total cost incurred across all links in the WAN:

$$\text{minimize } Z = \sum_{i=1}^{m} c_i \cdot z_i$$

Constraints. The traffic allocations are subject to constraints on link capacities. Since, the offline setting assumes knowledge of traffic demands, the traffic scheme must allocate flow in a way that the egress traffic demand is met in all time slots.

Formulating percentile cost as k-max. The cost function consisting of the sum of 95^{th} percentile utilization of links is non-convex. Previous work has shown that optimizing percentile cost functions is NP-HARD [20]. We later show that techniques from previous work are not effective in saving bandwidth costs of edge links (§4.3). We formulate the exact 95^{th} percentile of traffic allocations as part of the objective function. We note that the 95^{th} percentile of a distribution of n numbers is the same as their **k-max** where $k = n/20$.

Key insight. The key insight of our formulation is that link utilization during 5% of time slots do not contribute to its 95^{th} percentile cost. This means that 5% of time in any billing month is *free* regardless of the traffic it carries. We capture this insight in the optimization formulation using binary integer variables λ_{ij} for each decision variable x_{ij}. λ_{ij}s are also decision variables of the optimization which reflect whether their corresponding x_{ij}s contribute to the link cost. This is expressed with the indicator constraint:

$$(\lambda_{ij} == 0) \implies z_i \geq x_{ij}, \forall i, j \tag{1}$$

We note that only 5% of all $\{x_{i1}, x_{i2}, .., x_{in}\}$ can have their corresponding $\lambda_{ij} = 0$ since we can get away with considering 5% of allocations as *free*. This is expressed using Big-M constraints in the formulation [14]. The minimization objective ensures that of all λ_{ij}s, the ones corresponding to the top $k-1$ of the allocations (x_{ij}) at a link do not contribute to its cost.

Implementation details. Algorithm 1 formulates the traffic cost optimization problem as a Mixed Integer Linear Program (MILP), which is computationally hard to solve. We implement the formulation using the CVX [8] framework and solve it with the commercial optimization solver, GUROBI [15] on a machine with 12 cores and 48 Gb RAM. Our choice of solver is motivated by the computational complexity of Algorithm 1. Commercial solvers like GNU LPK [11] and CPLEX [18] were orders of magnitude slower than GUROBI in solving our formulation.

Algorithm 1: WAN Egress Traffic Allocation

Inputs:

n: number of five-minute time slots in a month
m: number of peering links in the WAN
l_i: Peering link i
C_i: capacity of peering link l_i
c_i: peering rate (USD/Mbps) for link l_i
d_j: egress demand from the WAN in time slot t_j
$k = \frac{n}{20}$
M: large integer constant

Outputs:

x_{ij}: traffic allocation to link l_i in time slot t_j
λ_{ij}: binary variables that discount top-k x_{ij}s
z_i: billable bandwidth on link l_i

Minimize: $\sum_i z_i \cdot c_i$

subject to:

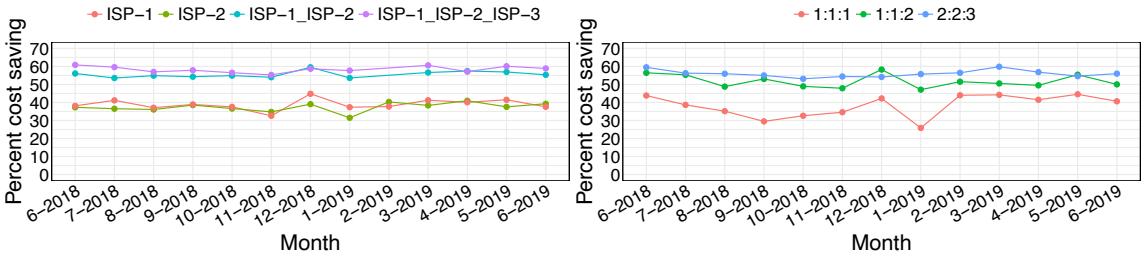
$0 \leq x_{ij} \leq C_i, \quad \forall i, \forall j$
$\sum_i x_{ij} = d_j, \quad \forall j$
$\sum_j \lambda_{ij} = k - 1, \quad \forall i$
$z_i > x_{ij} - M \cdot \lambda_{ij}, \quad \forall i, \forall j$

3.3 Generalizable and large saving potential

Using the set of peering links (L), peering rates (c_i), link capacities (C_i) and real egress traffic demands (d_j) from a large commercial cloud network, we formulate instances of Alg. 1. The egress traffic demands (d_j) are collected from June 2018 to June 2019 and consist of flow (megabits) that traversed the BGP peering links in each 5-minute interval. Peering rates remain constant during the course of our study. This provides 12 instances of Alg. 1, one for each 1-month billing period. We discuss the implementation details and assumptions in §3.4 and offer a preview of the results here. We compare the cost of allocations computed by Alg. 1 with the real allocation cost incurred by the cloud provider and find that Algorithm 1 reduces the combined cost of the three ISPs that contribute a majority of the bandwidth cost (ISP-1, ISP-2 and ISP-3 peer links) by 65% on average (Figure 4a).

Impact of participating links. When the input to the optimization is a *single* peer's links and traffic matrix, we observe lesser, yet significant, cost savings. This can be seen in the trends for ISP-1 and ISP-2 in Figure 4a. This shows that our cost optimization techniques can be deployed incrementally in the cloud WAN by engineering the traffic flow to a few ISPs at first. The fraction of savings increase as more outbound links are included in the optimization.

Impact of peering rates. We show the impact of relative peering rates of the three participating ISPs in the cost optimization. For the optimization instances demonstrated in



(a) Cost savings with different sets of participating links.

(b) Impact of peering rates on cost saving.

Figure 4: (a) Cost savings for 12 billing cycles using traffic matrices of ISP-1, ISP-2, ISP-3 and their combinations. (b) Cost saving with ISP-1, ISP-2 and ISP-3 for different peering rate ratios.

Figure 4a, the ratio of peering rates of the ISPs is 2:2:3. While the exact peering rates are confidential, their ratio shows that links belonging to two ISPs cost less than the third ISP. To ensure that the cost savings are not simply a function of this specific cost ratio, we compare the savings from the optimization when the peering rates are in 1:1:1 and 1:1:2 ratios and demands are the same as before. Figure 4b shows that savings are significant ($\approx 40\%$) even when all links have the same peering rate. Significant cost savings with different peering ratios demonstrate the generality of our results.

Impact of engineered traffic volume It may not be desirable to allow all traffic from edge links to be engineered for saving network costs. For instance, it may be important to egress some portion of the traffic on the same edge link where the client request entered the cloud for performance or geopolitical reasons. We find the impact of the fraction of traffic that can be engineered on a per-link basis by computing the cost gains for the month of June 2018 when the fraction of engineered outbound traffic on the edge links of ISP-1, ISP-2 and ISP-3 is 50%. We find that the resulting cost savings are 37.5%. We note that the solution took longer than our time limit for the solver and therefore the LP gap was higher than 15%. Similarly, when the fraction of traffic engineered on a link is reduced to 40%, the overall cost saving is 28.6%.

3.4 Computing optimal traffic allocations

We now discuss the details of our implementation of Alg. 1.

Managing the scale of the problem. Due to the non-convex nature of the problem, even state-of-the art optimization solvers can take an impractical amount of time to approximately solve Algorithm 1. We take advantage of our findings from §3.1 and only engineer peer links to the three North American ISPs (ISP-1, ISP-2 and ISP-3, anonymized for confidentiality) which incur a majority of the inter-domain bandwidth costs to the cloud. Each of the 3 ISPs peers with the cloud provider at tens of locations in North America, contributing 56 peer links between the cloud network and the three ISPs. We solve Algorithm 1 for different sets of peering links: first considering links with ISP-1 and the egress demand (d_j) that gets served over links with ISP-1. Similarly, we solve problem instances with links and demands of ISP-2, ISP-3, ISP-1 and ISP-2 and ISP-1, ISP-2 and ISP-3 as input.

Efficient computation of the lower-bound. Cutting-edge optimization solvers use a combination of techniques to solve general Mixed Integer Programs (MIPs). At a high level, the first step is *relaxing* the MIP to an efficiently solvable Linear Program (LP) by removing the integral constraints. If a feasible solution to the LP is not found, the MIP, in turn, is also infeasible. If a feasible solution is found, the solution of the LP is a lower bound to the solution of the original MIP. Therefore, in a minimization problem like Algorithm 1, the LP solution provides the lower bound on bandwidth cost without having to solve the MILP.

Running time of the optimization solver. We note that Algorithm 1 has $O(mn)$ Real decision variables and just as many binary variables. Predicting the difficulty of Integer programs in terms of the number of variables and constraints is hard. Indeed, *increasing* the number of links (size of set L) *reduces* the algorithm's running time. The rationale behind this counter-intuitive behavior is that higher number of peering links make it easier for the optimization to meet demands without raising the 95^{th} percentile utilization of the links.

Once the LP relaxation has been solved, MIP solvers use a branch-and-bound strategy to find feasible solutions to the MIP from an exponential number of possibilities. As a result, some instances of the optimization can take several hours to solve. We use two techniques to bound the time of the solver. First, using the efficiently computable LP relaxation, we compute the proximity of the MIP solution to the theoretical lower bound. Second, we configure the branch-and-bound algorithm to return the current-best feasible solution after a fixed amount of time has elapsed. We configure the solver to stop if the current best feasible solution to the MIP is within 15% of the LP optimal or if the solver has run for 15 hours.

Some instances of the optimization problem took 1-2 hours to find solutions while for others, the solution space had to be explored for 15 hours. On average, instances of Algorithm 1 took 6 hours to finish. The variance in run-time is due to differences in traffic demands of months. One strategy that was effective in speeding the optimization involved using

the values of decision variables from the previous month as initial values of the corresponding decision variables for next month's model. We found that using this *warm-start* strategy reduced the running time by 3X with instances taking 2 hours to solve on average. We describe other approaches that did not reduce the running time in Appendix (§A.1).

Gap from LP optimal. While the optimal solution to the LP relaxation provides a lower bound on the minimum cost of allocations, this lower bound is not always feasible. To improve the run time, we set a break condition while solving the problem instances to either reach within 15% of the LP optimal or spend 15 hours in solving the MIP using branch-and-bound. For the instances we solved, the average gap of the final MIP solution from the LP optimal is 9% *i.e.,* the solutions are very close to the theoretical lower bound.

4 Online cost-optimization with CASCARA

Results of the offline allocation scheme (3.2) show that there is significant potential for optimizing bandwidth cost at the cloud edge. There are two caveats to the scheme's use: first, it assumes knowledge of outbound demand for every time slot of the billing cycle. In practice, an online algorithm that can allocate network flow to peer links without the knowledge of future demands is required. Second, the optimization formulation (Algorithm 1) takes two hours on average to provide optimal traffic allocations for the entire month. However, state-of-the-art TE controllers compute traffic allocations every 5-10 minutes, making it crucial to have an online solution that is efficient and effective. In this section we develop a heuristic-based online traffic allocation framework that uses insights from the offline optimal solutions to Algorithm 1. Despite the complexity of the cost optimization problem, we show that a simple and efficient algorithm with few hyperparameters governs the closeness of the heuristic solution to the offline optimal. The heuristic allocations achieve bandwidth costs savings within 5% of the optimal.

Consider the set of edge links from the cloud, L = $\{l_1, l_2, ..l_m\}$. Let L_i be a subset of L, such that links in L_i are each priced at p_i per Mbps. For example, the setup in (3.2) has two such subsets, L_1 and L_2 where links in L_1 are priced at p_1 and those in L_2 are priced at p_2. Since the peering rates of links to ISP-1, ISP-2 and ISP-3 are in the ratio 3:2:2, $p_1 = \frac{3}{2}p_2$. From the results of Section 3.4, we derive three key insights about the optimal traffic allocations:

Lower utilization of expensive links. When $p_2 < p_1$, the optimal traffic allocations use links in L_1 minimally. This means that barring capacity considerations, it is always cheaper to use links in L_2 to meet the demand and only use links in L_1 for their *free* 5% time slots.

Maximize the utilization of free slots. Figure 5 shows the density distribution of optimal allocations on an edge link by Algorithm 1. We note that the optimal allocations reduce the link's 95^{th} percentile utilization to $\approx$ 15% of its capacity.

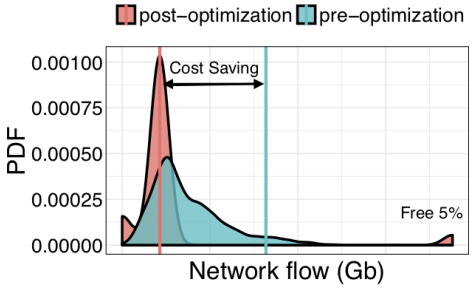


Figure 5: Optimized allocations on an edge link for a month. The vertical lines show the pre- and post-optimization 95^{th} percentile utilization on the link. (X-axis labels removed.)

However, during 5% of time slots the link is utilized nearly at full capacity without contributing to the billable-bandwidth. Optimal allocation on all links show similar patterns.

Link utilization below the 95^{th} percentile is uneconomical. Let u_j be the 95^{th} percentile utilization of an egress link l_j. Assigning less than u_j flow to link l_j in any time slot is wasteful, *i.e.,* the link will get billed for u_j even if its utilization in other time slots is lower (Appendix Figure 13a).

4.1 Online traffic allocation

Using insights derived from the optimal allocations, we propose an online traffic allocation scheme, CASCARA, for the cloud edge. CASCARA *pre-decides* the fraction of the total network capacity that will be the billable bandwidth for the month (C_f). Given the billable bandwidth, finding the optimal pre-decided 95^{th} percentile utilization of link l_j (u_j) is a special case of the bin-packing problem. Thus, greedy assignment of C_f to links in the increasing order of their peering rates minimizes the total bandwidth cost of the network. Since subsets of links ($L_i \subset L$) have the same peering rate, we assign u_j to links in the same subset using the progressive filling algorithm to ensure max-min fairness [2] within link subsets.

When a billing period begins, every link has a 95^{th} percentile utilization (u_i) assigned to it. As new outbound demands arrive, if they can be met with $\sum u_i = C_f$ capacity, CASCARA allocates corresponding flows to the links. However, if the outbound demand exceeds C_f, CASCARA chooses to utilize one or more links at near full capacity to meet the demand. Since 5% of billing slots do not contribute to the links' costs, CASCARA ensures it only runs a link at near capacity for 5% or fewer billing time slots.

Parameters to the online algorithm. It is crucial to select C_f such that all demands in the billing period are met within C_f or by augmenting C_f with the extra capacity of links in their 5% free time slots. Once a link's 95^{th} percentile utilization has been chosen to be u_i, using it for any lesser makes no difference to its final cost. The choice of C_f is critical to making a feasible allocation. If C_f is too low, the allocation may be infeasible or if it is too high, the bandwidth cost can be sub-optimally high. We discuss the choice of initial C_f and

how CASCARA improvises when the chosen C_f is too small to meet the demand during the billing cycle.

Order of choosing peer links. CASCARA decides the order of links to be augmented above their allocation u_i to meet 5-minute demands higher than C_f. Using a configurable parameter, CASCARA can allocate how close the augmented allocation is to the link's capacity to prevent sudden link performance degradation. The time slots in which CASCARA augments the allocation to a link are called *augmented* slots. The augmented slots are limited to 5% for each link, making the order in which links are augmented relevant to the feasibility of an allocation. CASCARA uses a priority queue of all edge links where a link's priority is a combination of the time since it was last augmented and its capacity. If a link was augmented in the previous slot, it must also be augmented in the following slot, if required, so that the allocations do not change sharply. By prioritizing links with lesser capacity for augmentation, CASCARA ensures that free slots of links with higher capacity are not used pre-maturely.

Link augmentation order does not impact feasibility. If CASCARA's assignment of u_is and the order of link augmentation leads to an infeasible allocation problem, any change to the order of link augmentation does not render the allocation feasible (Proof in Appendix A.2). Since u_is are derived from C_f, the key input parameter to CASCARA is C_f. Algorithm 2 shows the online traffic allocation scheme of CASCARA in brief (details in Appendix Algorithm 3).

Insufficient C_f and infeasible allocation. If the initial capacity fraction assigned by CASCARA ends up being insufficient to meet the demand in a time slot, despite augmenting the allocations to all edge links that have free slots remaining, we consider the allocation infeasible. This means that it is no longer possible to limit the billable bandwidth of this month to C_f and the C_f value must be increased. CASCARA increases the value of C_f by step size (β) to meet the demand. Until it becomes necessary to increase C_f in a billing cycle, CASCARA has under-utilized the links stay under C_f. Increasing C_f to $C_f + \beta$ renders the past efforts to keep C_f low, futile. Indeed these efforts may have wasted the augmentation slots of links before C_f is incremented. However, there is no choice but to increase C_f as traffic demands must always be met. In the ideal case, initial value of C_f is just enough to meet demands in the entire billing period using augmentation slots when needed. On the other hand, starting the billing cycle with a C_f that is higher than required leads to sub-optimally high bandwidth costs. We show that the *ideal* C_f value is sufficient in ensuring that CASCARA finds optimal cost allocations.

Improvising billable bandwidth preemptively. When CASCARA finds that the demand is too high to accommodate in the current C_f, it increases C_f by β. Increasing the billable bandwidth estimate, C_f is a tradeoff – increasing too late in the billing cycle leads to wasteful use of links' free slots until the increase and increasing it too early reduces the cost saving potential. We capture this tradeoff by introducing the third and final parameter of CASCARA: α. α is the increase in C_f during the monthly billing cycle before an infeasible allocation is encountered. The goal is to preemptively increase C_f if such an increase is inevitable later in the month.

Algorithm 2: Online Traffic Allocation Per-Timestep

```
Function allocate_timestep(d, f):
    if d ≤ C_f then
        allocate C_f to links in L
        return true
    else
        d = d − C_f
        while linkqueue do
            l = pop(linkqueue)
            augment l
            decrement l's priority and free slots
            decrement d by l's augmented capacity
            if d ≤ 0 then
                return true
        return false
```

4.2 Finding CASCARA's hyperparameters

We show that by setting C_f effectively, CASCARA's online traffic allocation (Algorithm 2) can be nearly as effective as the offline solutions of Algorithm 1. We set C_f to different fractions of the total network capacity, ranging from 0 to 1, in steps of 0.01. We compare the cost saving from the feasible allocation using the smallest C_f with the optimal cost saving[3] and find that on average, CASCARA with the optimal initial C_f achieves savings within 2% of the offline optimal allocation.

Setting C_f. CASCARA with the optimal C_f is called CASCARA-offline since it has prior knowledge of the lowest C_f for feasible allocations. CASCARA-online assumes no such knowledge and uses the optimal C_f of the previous billing cycle as the current month's initial C_f. This choice is motivated by strong daily, weekly and monthly seasonality in the outbound traffic demands. Previous month's C_f is the optimal value for the next month 64% of the time. For the rest, the average difference between optimal C_f and its initial setting is of the network capacity. When the initial C_f is not optimal, the allocation becomes infeasible and CASCARA has to increase the C_f to meet the traffic demands.

Finding α and β with grid search. Increase in C_f is a definite increase in the bandwidth cost for the billing cycle. The step size by with C_f is increased (β) is also important: too high and it would wastefully increase the cost, too low and it would mean having to increase C_f again in the future. Once increased, there is no cost saving advantage to reducing C_f. Incrementing C_f later is worse than having started with the

[3]For confidentiality reasons, we cannot not share the capacity fractions.

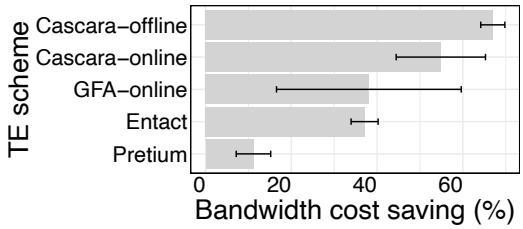


Figure 6: Costs savings from CASCARA and related approaches. Bars show the mean and wiskers show the standard deviation.

optimal C_f since links' augmentation slots are wasted before the increment is made. Thus, preemptively increasing C_f by α during the billing cycle mitigates the issue of wasteful use of link augmentation slots. The hyperparameters, α and β are important to select. We perform a grid search to find the ones best suited for the cloud network. Details of the grid search are in Appendix A.5. The best values of α and β are used to for the following discussion.

4.3 Comparison with previous work

We now discuss the cost savings enabled by CASCARA-online over twelve billing months from June 2018 to June 2019 (Figure 6). As before, we use the production network's traffic demands, topology and peering rates to measure the cost savings that CASCARA-online would provide. We first show that CASCARA-online achieves 55% cost saving, within 10% of the savings from CASCARA-offline which knows the optimal C_f in advance. Then, we evaluate existing approaches that have focused on similar objective functions as CASCARA. We exclude approaches that delay traffic to future time slots [13,21] as these are not viable for the cloud provider we study (§7). The three main systems from related work are:

Pretium for dynamic file transfers in the WAN [20]. Pretium focuses on optimizing percentile costs of internal WAN links for dynamic transfers within the WAN [20]. They proposed to use the average of top 10% utilizations as a proxy for 95^{th} percentile cost of links. We find that Pretium offers modest cost saving of 11% on average compared to CASCARA's 55% savings for egress WAN traffic. Pretium assumes that the 95^{th} percentile of a link's utilization is linearly correlated with the *average of top k utilizations* [20]. We evaluate this assumption using the utilizations of over 50 peering links from the cloud WAN to large ISPs in N. America. Figure 13b shows the Pearson correlation coefficient to measure the extent to which the average of top 10% utilizations can be used as a proxy for 95^{th} percentile utilization of inter-domain links. We find that the correlation coefficient for over 25% of the links is less than 0.5. Since previous work's hypothesis was derived from the data of a *single* WAN link measured a few years ago, the correlation between average of top 10% and 95^{th} percentile utilization may exist for some links but not all. Ever-changing traffic patterns from WANs due to new services like gaming also explain this difference.

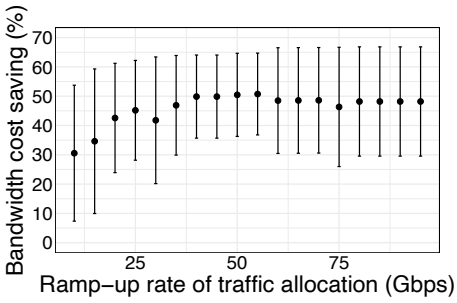


Figure 7: The impact of ramp-up rate on cost saving potential (mean and std. deviation calculated over 12 billing cycles).

Entact for cost minimization in clouds [35]. Entact shares a lot of the goals with CASCARA, including finding cost optimal traffic allocations constrained by client latency. However, Entact chose to optimize *linear bandwidth prices* since percentile pricing is hard to optimize [35]. In a linear pricing scheme, greedy traffic allocation to cheapest links is optimal. However, the greedy algorithm does not fare well in percentile pricing schemes, as show in Figure 6's comparison between CASCARA and Entact. The reason is that allocations in *every* time slot contribute towards the billable bandwidth in linear pricing schemes (*e.g.,* average and sum of allocations) but in percentile pricing, some percent of the allocations are *free*. Greedy allocations fail to take advantage of this phenomenon.

Global Fractional Allocation (GFA) for multihoming [12]. Finally, authors of [12] analyzed cost optimizations in the setting of multi-homed users. GFA comes closest in its approach to CASCARA and this is also reflected in the cost saving comparison in Figure 6. However, CASCARA outperforms GFA by 17% in the average case. There are two main reasons for this: GFA assumes a much smaller scale of the problem where the options for allocations are 3 to 4 upstream ISPs. This makes their naive estimation of cost lower bound ineffective: by using *only* 5% of the timeslots of peer links to meet demands was a viable option, traffic allocation would be *free*. Secondly, when GFA runs into an infeasible allocation, it assigns *all* remaining flow to a single link. This is often impractical at the cloud scale where the demand is too high for one peer link to handle the slack.

And finally, there are several realistic factors that need careful consideration: latency from peer links to clients and existence of routes at the peering router to engineer traffic. CASCARA not only performs better in idealized environments by achieving higher cost saving that existing systems, it also takes real-world constraints of a large production WAN into account. We describe these in further details in §5.

4.4 Operational safety checks in CASCARA

We discuss the safety checks built into the CASCARA algorithm to ease the process of operating it in production.

Stable traffic allocation. One concern with algorithms as-

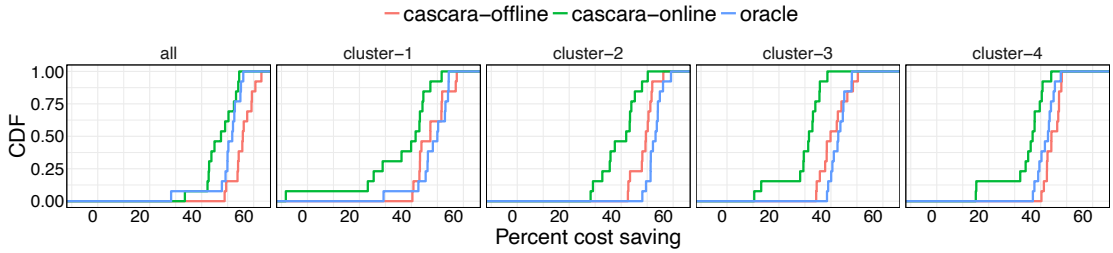


Figure 8: shows distributions of percent cost saving by CASCARA-offline, CASCARA-online and the oracle (Algorithm 1) over 12 billing cycles. CASCARA-online achieves near-optimal saving with different sets of links and corresponding demands as input.

signing traffic flows on peer links is that the allocation must be mindful of the performance impact on the inter-domain paths. CASCARA ensures that allocation to backup BGP paths does not change too rapidly by using a *maximum ramp-up* rate parameter that controls the maximum increase in the allocation to any peer link in the network. This ramp-up rate paces traffic allocation to links and allows CASCARA to incorporate path performance feedback into its decision making. We discuss how CASCARA incorporates performance metrics in its control loop in the next section. Figure 7 shows the cost saving potential of CASCARA as a function of the ramp-up rate. Very slow shifts which use a maximum ramp-up rate of 10 Gbps restrict the cost savings of CASCARA. However, at 30 Gbps ramp-up rate, CASCARA has reached its full saving potential and more rapid shifts of traffic do not offer much improvement in cost savings percentage.

Predictable traffic allocations on edge links. CASCARA's traffic allocation to edge links are more stable than present-day allocations which are driven by user-facing demands. There are two reasons for this. First, CASCARA selects a pre-decided fraction of a link's capacity as the utilization on the link for 95% of billing slots and changes are made to this fraction only when it is essential for meeting demand over C_f. Secondly, even when the allocation to a link has to be augmented, CASCARA ensures that a link, once augmented, is used until its free slots have been exhausted. Predictable allocations on edge links allow network peers to provision capacity appropriately in place of being prepared for arbitrary spikes in traffic demands.

Incremental deployability. CASCARA can be incrementally deployed across edge link groups in the cloud. To show this, we divide the peer links of ISP-1, ISP-2 and ISP-3 into four geographical clusters based on their PoP. These four clusters correspond to links at PoPs in north-central, south-central, East Coast and West Coast regions of North America. We compute the cost savings within each cluster by engineering the demands of the cluster onto its links. Figure 8 shows that CASCARA-online can achieve near-optimal cost (CASCARA-offline) savings across all peer links (cluster *all*) and also within the 4 geographical clusters. We note that in some cases CASCARA-offline achieves higher cost saving than the oracle (Alg. 1) due to the LP gap in the solution of the MILP (§3.4).

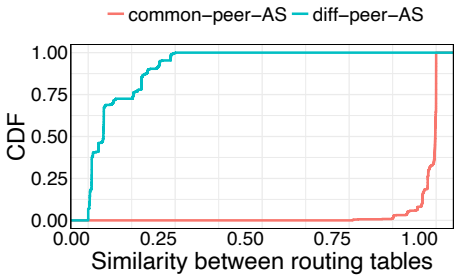


Figure 9: shows the distribution of address space similarity between peer links of ISP-1, ISP-2 and ISP-3 at different PoPs of the cloud. Each ISP announces nearly the same address space at different PoPs but the overlap in address space across ISPs is very small.

5 Performance-aware cost saving

We have demonstrated that there is significant potential of saving inter-domain bandwidth costs in a cloud network (§3) and CASCARA's efficient online algorithm can realize this potential by achieving near-optimal cost saving (§4). In this section we discuss practical aspects of achieving cost savings, namely, feasibility of engineering egress traffic in a WAN and the impact of CASCARA on client latency.

5.1 Availability of client routes at peer links

CASCARA engineers outbound traffic demand to peer links to achieve cost optimality over the billing cycle. However, it must ensure that peer links have the routes required for traffic shifted onto them. Otherwise, traffic to clients could get black-holed at the peering edge router. Using the routes announced by the three ISPs we focus on, we measure the address space overlap between peer links and find that ISPs announce the same address space across different peering locations (*e.g.,* Dallas vs. Seattle) but the overlap of address space across peers (*e.g.,* ISP-1 vs. ISP-2), even at the same PoP is minimal (Figure 9). Thus, CASCARA needs a mechanism to track the existence of relevant routes at peer links.

Tracking prefix route announcements by ISPs at different cloud PoPs in CASCARA leads to an explosion of the problem size since there are over 600,000 prefixes on the Internet. Aggregating clients to their corresponding geographic metropolitan area (metro) and autonomous system (AS) pair

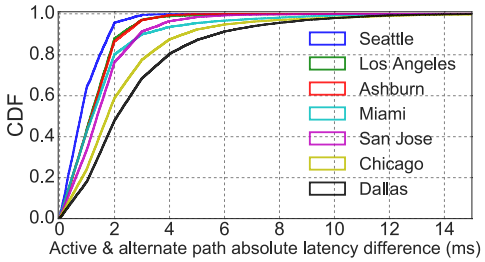


Figure 10: The difference in median latencies between primary and alternate BGP paths calculated from client measurements. The difference between latencies of primary and alternate paths is small.

significantly reduces the scale of the problem. This grouping of client prefixes within the same AS and small geographical locality has been used effectively in previous work [4]. We find that the points of presence where the cloud provider peers with ISP-1, ISP-2 and ISP-3 serve approximately 40,000 (metro, AS) pairs, reducing the scale of the mapping required to capture the existence of relevant BGP routes at peer links. Thus, we construct a bi-partite mapping between clients and peer links *i.e.,* an edge between client c to peer link p implies p has the relevant routes to c. We then constrain the traffic allocation in each timestep by the client to peer link mapping. We compute this allocation efficiently with a linear program (LP) within Alg. 2 that maps clients demands to peer links.

5.2 Bounded impact on client performance

Next, we tackle the challenge of limiting the performance impact of CASCARA's cost optimization. For this, we continuously measure the performance of alternate BGP egress paths to destination prefixes by directing a small amount of traffic over alternate peer links at eight PoPs [26,27,35]. We selected these PoPs as they carry high traffic volume – approximately 47% of all the cloud provider's North American traffic, and have high capacity alternate links.

Links at the same PoP have equivalent client latency. We analyze over 300 million measurements to the cloud PoPs for the month of August 2020, spanning 40,000 client metro and AS pairs, each with thousands of latency measurements towards the cloud on any given day. We first show the existence of latency equivalent peer links at the same PoP. Borrowing from existing methodology [26], we measure the difference in median latency between the BGP best path (*primary*) and the alternate BGP path for all clients that are served by the PoPs over 15 minute time buckets. Figure 10 shows that 80% of the time the difference in the latency is less than 3 ms. This implies that shifting client traffic to links at the same PoP, impacts the client latency by 3 ms or less.

Shifting traffic to peer links at a PoP different than the one where it ingressed introduces two challenges. First, it can inflate latency as the traffic would traverse the cloud backbone to reach the second PoP. The second PoP could be further from the client than the original, also inflating PoP to client latency. Second, traversal of the cloud backbone can congest backbone links but cloud providers often overprovision backbone capacity [6] and manage intra-WAN link utilizations with centralized controllers like SWAN [17] and B4 [19] to mitigate hot spots. Thus, we focus on the latency impact of CASCARA in this work. We find the *primary* PoP and peer ISP which historically has been the preferred egress for a client. This primary link defines the baseline for our experiments – any changes in client latency are measured in comparison with the primary peer and PoP.

Bound the latency impact in egress link selection. To limit the degradation to client latency, we inform CASCARA's allocation (Algorithm 2) of the most recent latency from a peer link to the client. In every timestep, while fulfilling demands to a client, CASCARA enforces that traffic is allocated along the primary and other sets of links. We select the set of links to empirically construct the relationship between latency impact and saving of CASCARA. We consider the set of links for each client to include ISPs with route towards the client – including a transit ISP, at the client's primary PoP. This means that along with the links to its primary ISP, the client's demand could be carried over the transit ISP link at the same PoP. This can increase the set of outbound link options for a client by two links in the best case. Since, links at the same PoP have equivalent latency, this configuration of CASCARA does not cause significant latency degradation (Figure 10).

We use CASCARA to engineer traffic at each PoP and compute the offline cost optimal solutions (Figure 11) for comparison. At some PoPs (PoPs 0, 2 and 13), there are up to five latency-equivalent peer links to most clients. *e.g.,* two interconnections with ISP-1, one with ISP-2 and two with the transit ISP. CASCARA-offline shows the potential to save up to 50% of bandwidth costs at such PoPs. At other PoPs (PoPs 4, 5, 11), there are only 2 latency-equivalent peer links to most clients *e.g.,* one interconnection with ISP-1 and one with the transit ISP. Moreover, high diversity in demands across PoPs due to client population density leads to differing opportunities of cost savings across PoPs. We use CASCARA-online to engineer traffic in an online manner with route and latency constraints. In each five minute timeslot, CASCARA allocated traffic to clients on latency equivalent links at the client's primary PoP. On average, each iteration of CASCARA takes approximately 3 seconds to compute traffic allocations, including the construction of the LP and extraction of traffic allocations on links. We note that our implementation uses Python 2.7 and could be further optimized for running time. However, TE systems typically perform allocations once every 5-10 minutes, thus CASCARA's runtime of 3 seconds is reasonable. Across all PoPs, CASCARA achieves the overall cost saving of 21% while ensuring that client latency remains unaffected. The per-PoP configuration we have evaluated enforces the strictest possible latency bound on CASCARA. CASCARA allows cloud providers to configure the acceptable

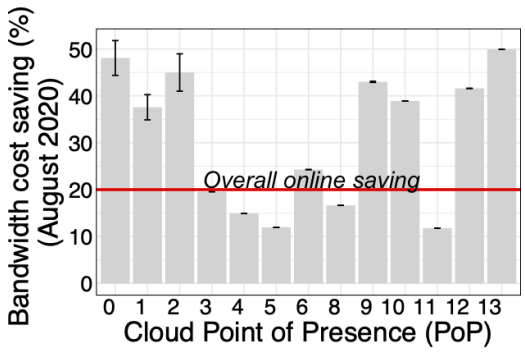


Figure 11: shows cost savings with CASCARA when engineering traffic on a per-PoP basis while limiting the impact on client latency to 3 ms in the worst case (mean and standard deviation computed over three runs of CASCARA).

worst-case latency degradation while saving bandwidth costs.

6 Discussion

In this section, we investigate the source of CASCARA's cost savings. We discuss implications of our findings on peering contracts with ISPs and bandwidth pricing on the Internet.

6.1 Where do the cost savings come from?

Network operators have historically used heuristics to limit their bandwidth costs. These include, load balancing traffic over equivalent links and preferring cheaper peer links in the BGP best path selection by setting localpref appropriately.

Localpref based cost saving is sub-optimal. We illustrate the cost savings from CASCARA with a small example using 2 links and 3 billing slots. There are two egress links from a network (Link 1 and Link 2), each of capacity 5 traffic units and unit peering rate. The traffic demand is assigned to Link 1 and 2 in any time slot (Figure 12). Traffic must not be dropped if there is enough capacity on the outbound links. For simplicity, the links are billed using the median (50^{th} percentile) utilization over three time slots. Since the peering rate of both links is the same, localpref-based cost minimization will simply balance traffic on the two links. Under this scheme, the link utilizations are : $1, 2.5, 1.5$ in time slots 1, 2 and 3 respectively (shown in red in Figure 12) for both links. The median utilization is 1.5 for both, the total cost of the links is 3 units. An alternate traffic assignment to the links is shown in blue in Figure 12, where the utilizations of link 1 and 2 are $\{1, 5, 0\}$ and $\{1, 0, 3\}$ respectively. The median cost in this case is 1 for both links, total cost being 2 units. This scheme saves one third of the traffic cost while meeting the same demand. We note that by extension, sending all traffic to a link that is cheaper would also be sub-optimal.

Free time slots for saving cost. The example shows that in case of median billing, one of the three time slots does not contribute towards the final cost of the link. Each link has one free slot that can absorb peaks in demands to reduce

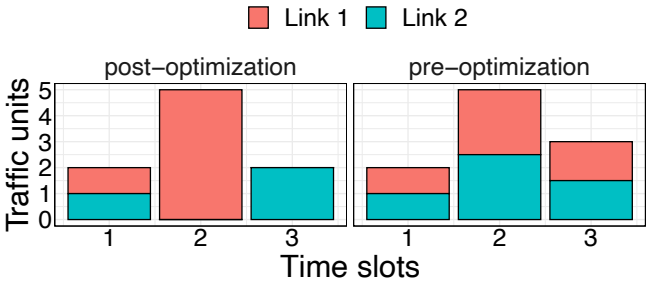


Figure 12: A toy example comparing CASCARA's cost-optimized allocations vs. load-balanced allocations.

costs. Similarly, bandwidth on the Internet is billed using 95^{th} percentile billing, meaning that 5% of 5-minute time slots in a month are *free* for each link. This implies that for roughly 36 hours in a month, traffic allocation on any link does not contribute to the final billed cost. While it may seem that the free slots provide little wriggle room for saving cost, cloud providers have a rich diversity of network peers in several PoPs. These peer links provide free slots in the billing context and enable multiple latency-equivalent ways to reach clients.

6.2 Do the findings generalize?

We believe our results generalize to any large global cloud, content provider, or content delivery network. The first reason is that the cloud provider network is not unique. These networks all share several critical properties in common with each other: **(1)** presence in hundreds of PoPs around the world to deliver traffic very close to users and **(2)** extensive peering and short AS paths [5, 32]. The second reason is that other large networks have shown that given such large deployments and peering, many of the alternate paths to users have similar latency [1, 26]; also allowing these networks to optimize bandwidth costs with stable performance.

Cost optimization is not a one-time effort. Traffic patterns across billing slots change – demands have been rising steadily at 30-40% per year. The surge in demand [7] from the COVID-19 pandemic has made traffic patterns more dynamic. We have evaluated CASCARA using over a year worth of demands, including evaluation in August 2020 to capture the post-pandemic traffic growth. Our findings show small month-to-month variation in saving but overall, the savings are significant and consistent. We note that cost savings compound over time as demand continues to rise exponentially.

6.3 How practical is CASCARA?

While CASCARA benefits from large sets of latency-equivalent peer links, it can be deployed incrementally over peer links (§5), allowing cloud operators to choose to expand CASCARA's purview over time. CASCARA can bound the amount by which allocations to links can change across time slots to prevent sudden changes in traffic (§4.4). Moreover, we foresee CASCARA as a component of a larger software-defined edge [27, 33] that already prioritizes successful traffic

delivery based on the capacity and availability of the downstream path. During demand surges or outages, the high priority components of the TE system may take action to mitigate customer impact, putting CASCARA on hold for some types of traffic for short periods of time. Automated network build-out alerts limit the duration of persistent capacity crunches, enabling cost savings from CASCARA in the long term. Where cost-optimization falls among second-order priorities will vary across cloud providers and their business needs.

6.4 Implications for existing peering contracts

An important concern in optimizing the cost of inter-domain traffic is the long-term impact it may have on peering contracts. For instance, if free peers observe higher traffic volume from the cloud, they may reconsider their peering agreement or lean towards paid exchange of traffic [22]. Due to these factors, we evaluated CASCARA only on links with paid North American peers. We argue that the peering rate captures the value of the interconnection to both networks involved and thus optimizing the outbound allocations for cost, not exceeding the peering port capacity at the edge, is a reasonable strategy. Additionally, peering rates in certain regions of the world are disproportionately high due to monopolistic transit ISPs and complicated socio-political factors, making high bandwidth rates the cost of operating in the market.

6.5 Implications for bandwidth pricing

CASCARA shows that the abundance of latency-equivalent peer links has enabled networks to significantly reduce their expenditure on inter-domain bandwidth. With the findings of CASCARA, we encourage the community to revisit the classic problem of pricing inter-domain traffic effectively. A subject studied since the dawn of the Internet [24], inter-domain bandwidth pricing models and rates determine paths taken by traffic and subsequently the end-user performance. With the emergence of cloud and content providers as the source of disproportionately large volume of Internet traffic, current pricing models may not suffice in ensuring the harmonious existence of networks on the Internet [10, 30]. Today, a handful of networks (cloud and content providers) can take advantage of their rich connectivity to save inter-domain bandwidth costs, potentially taking a portion from the profits of ISPs. Some recent proposals suggest ways to better align the cost of Internet transit and the revenue gained by networks [16, 34].

7 Related Work

In this section, we discuss important pieces of work related to CASCARA and set them in the context of our contributions.

Intra-WAN traffic engineering. Large cloud providers have embraced software-defined, centralized traffic engineering controllers to assign flow within their private WAN to maximize their utilization, guarantee fairness and prevent congestion [17,19,23]. Bandwidth costs in the context of WANs were considered in Pretium [20] (comparison with CASCARA in Section 3.4). Stanojevic *et al.* used Shapley values to quantify the value of individual flows under percentile pricing [28].

Engineering the WAN egress. Recent work has proposed a software-defined edge to manage outbound flows from their networks [27, 33]. The goal of these efforts has been to react to poor client performance by switching to better performing BGP next hops. The allocation decisions made by CASCARA can be implemented using a software defined edge like Espresso or EdgeFabric. The subject of TE in multi-hoped networks has been studied [12, 25] and we compare CASCARA with a representative set of work from this space (§3.4).

Engineering delay tolerant traffic. Previous work has explored the potential of delaying traffic across timeslots to save bandwidth costs at the end of the billing cycle [21]. However, the cloud provider we analyze does not consider delaying client traffic by several minutes as a viable option.

Performance-based routing on the Internet. Google's Espresso [33] implements performance-based routing on the Internet to improve client performance. Recently, other large global networks have shown limited potential in optimizing latency by routing [1, 26]. Our work effectively exploits this realization by optimizing cost while keeping latency stable.

Bandwidth pricing schemes. In the early years of the Internet, economists studied potential mechanisms to price bandwidth [24]. Congestion pricing was proposed to bill based on the use of network resources at times when they are scarce. These pricing schemes incentivize users to reduce consumption of network resources during peak utilization by pricing bandwidth higher when the network is congested.

8 Conclusion

In this work, we quantify the potential of saving inter-domain bandwidth costs in a large commercial cloud provider and find that optimal allocations can save up to 60% of current inter-domain bandwidth costs while meeting all traffic demands as they arrive. Inspired by this, we develop an efficient online TE framework, CASCARA, that achieves cost savings within 10% of the optimal. CASCARA's cost savings are robust to changes in traffic patterns and peering rates. Finally, we show that CASCARA can balance the cost-performance tradeoff by achieving 11-50% cost savings per cloud PoP without degrading client latency significantly.

9 Acknowledgements

We thank our shepherd, Srikanth Sundaresan and the anonymous reviewers for helping improve this work. We are grateful to Anuj Kalia, Akshay Narayan, Arvind Krishnamurthy, Dan Crankshaw, Emaad Manzoor and Phillipa Gill for their feedback. We thank Ingrid Erkman, Brett Lewan and the Azure WAN team for their feedback on CASCARA's design.

References

[1] Todd Arnold, Matt Calder, Italo Cunha, Arpit Gupta, Harsha V. Madhyastha, Michael Schapira, and Ethan Katz-Bassett. Beating bgp is harder than we thought. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, HotNets 2019, page 9?16, New York, NY, USA, 2019. Association for Computing Machinery.

[2] Dimitri P Bertsekas and Robert G Gallager. *Data networks*, volume 2.

[3] CAIDA. CAIDA AS Classification. http://data.caida.org/datasets/as-classification/, (Accessed on 2020-01-15).

[4] Matt Calder, Ryan Gao, Manuel Schröder, Ryan Stewart, Jitendra Padhye, Ratul Mahajan, Ganesh Ananthanarayanan, and Ethan Katz-Bassett. Odin: Microsoft's scalable fault-tolerant CDN measurement system. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 501–517, Renton, WA, April 2018. USENIX Association.

[5] Yi-Ching Chiu, Brandon Schlinker, Abhishek Balaji Radhakrishnan, Ethan Katz-Bassett, and Ramesh Govindan. Are we one hop away from a better internet? In *Proceedings of the 2015 Internet Measurement Conference*, pages 523–529, 2015.

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

[7] Cloudflare. Cloudflare During the Coronavirus Emergency. https://blog.cloudflare.com/cloudflare-during-the-coronavirus-emergency, (Accessed on 2020-03-12).

[8] CVX Research. Software for Disciplined Convex Programming. http://cvxr.com/, (Accessed on 2019-10-02).

[9] Dr. Peering. The Art of Peering: The Peering Playbook. http://drpeering.net/white-papers/Art-Of-Peering-The-Peering-Playbook.html, (Accessed on 2020-08-01).

[10] Free Press. Net Neutrality. https://www.freepress.net/issues/free-open-internet/net-neutrality, (Accessed on 2020-01-19).

[11] GNU. GNU Linear Programming Kit. https://www.gnu.org/software/glpk/, (Accessed on 2019-10-02).

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

[13] L. Golubchik, S. Khuller, K. Mukherjee, and Y. Yao. To send or not to send: Reducing the cost of data transmission. In *2013 Proceedings IEEE INFOCOM*, pages 2472–2478, 2013.

[14] Igor Griva, S Nash, and Ariela Sofer. *Linear and Nonlinear Optimization: Second Edition*. 01 2009.

[15] Gurobi. GUROBI Optimization. https://www.gurobi.com/, (Accessed on 2019-10-02).

[16] Yotam Harchol, Dirk Bergemann, Nick Feamster, Eric Friedman, Arvind Krishnamurthy, Aurojit Panda, Sylvia Ratnasamy, Michael Schapira, and Scott Shenker. A public option for the core. SIGCOMM '20, page 377–389, New York, NY, USA, 2020. Association for Computing Machinery.

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

[18] IBM. CPLEX Optimizer. https://www.ibm.com/analytics/cplex-optimizer, (Accessed on 2019-10-02).

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

[20] Virajith Jalaparti, Ivan Bliznets, Srikanth Kandula, Brendan Lucier, and Ishai Menache. Dynamic Pricing and Traffic Engineering for Timely Inter-Datacenter Transfers. In *SIGCOMM'16*, 2016.

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

[22] Steven Levy. *In the plex: How Google thinks, works, and shapes our lives*. Simon and Schuster, 2011.

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

[24] Jeffrey K MacKie-Mason and Hal R Varian. Pricing congestible network resources. *IEEE journal on Selected Areas in Communications*, 13(7):1141–1149, 1995.

[25] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, and S. Uhlig. Interdomain traffic engineering with bgp. *Comm. Mag.*, 41(5):122?128, May 2003.

[26] Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. Internet performance from facebook?s edge. In *Proceedings of the Internet Measurement Conference*, IMC ?19, page 179?194, New York, NY, USA, 2019. Association for Computing Machinery.

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

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

[29] TeleGeography. The State of the Network. https://www2.telegeography.com/hubfs/assets/Ebooks/state-of-the-network-2019.pdf, (Accessed on 2020-01-19).

[30] TIME. Netflix's Disputes With Verizon, Comcast Under Investigation. https://time.com/2871498/fcc-investigates-netflix-verizon-comcast/, (Accessed on 2020-01-19).

[31] Brian Trammell and Benoit Claise. Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information. *RFC7011*, 2013.

[32] Florian Wohlfart, Nikolaos Chatzis, Caglar Dabanoglu, Georg Carle, and Walter Willinger. Leveraging interconnections for performance: the serving infrastructure of a large cdn. In *SIGCOMM*, pages 206–220, 2018.

[33] KK 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 *SIGCOMM'17*, 2017.

[34] Doron Zarchy, Amogh Dhamdhere, Constantine Dovrolis, and Michael Schapira. Nash-peering: A new techno-economic framework for internet interconnections. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 403–408. IEEE, 2018.

[35] Zheng Zhang, Ming Zhang, Albert Greenberg, Y. Charlie Hu, Ratul Mahajan, and Blaine Christian. Optimizing cost and performance in online service provider networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, page 3, USA, 2010. USENIX Association.

A Appendix

A.1 Speeding the MIP solution

In this section we briefly describe the intuitive but ineffective methods we employed to speed the execution of the MILP. The methods did not yield a reduction in running time but we document these for completeness.

Solving the MILP in smaller time slices. Since link utilizations at the edge exhibit strong daily and weekly seasonality, we hypothesized that solving the cost optimization in smaller chunks of time, say, one week at a time, and then stitching together the resulting solutions would find the entire month's optimal allocations. While the smaller problems of weekly allocations could be solved in approximately ten minutes, when stitched together, the overall solution is very far from optimal. In fact, the allocations obtained via this process did not show any significant reduction in inter-domain bandwidth cost over the present-day traffic allocations. On investigating the reason why this approach does not work, we found that while there are regular trends in the traffic demand, bursts of traffic are not spread uniformly across all weeks of a month. Accommodating these bursts with a local, week-long view leads to overall poor cost saving from the stitched allocations.

Automated parameter tuning. The commercial solver we used (Gurobi), provides a tool for automatically tuning the parameters of the solver for a given optimization model. We attempted to use this tool to find parameters which gave the best performance in terms of running time and closeness to the optimal. However, our model was too large for the auto-tune to deliver any results. Thus, we selected the appropriate parameter values manually by several runs of the optimization. These parameters are documented in the code repository we have released.

A.2 CASCARA's link augmentation order

We show that changing the order of links that CASCARA augments during the billing cycle does not make an unfeasible

allocation, feasible. We take the example of an unfeasible ordering, $O_{unfeasible}$ where the demand in timeslot k cannot be met even after augmenting the capacity of all links. Consider the following change in the position of link l_i in the ordering: if l_i is picked for augmenting in timestamp k in place of timestamp j where $j \leq k$. If this were possible, then CAPACITY(l_i) would be available for use in timestamp k. However, this is not possible since l_i had to be the *smallest* capacity link that met the excess demand of timeslot j, any other link that takes its place has to have a higher capacity. This means that by using another link in place of l_i, we would reduce the available capacity in timeslot k. Thus, a change in ordering of links for augmentation would not make a problem instance feasible. □

A.3 Traffic allocation with CASCARA

In this section we discuss details of the CASCARA allocation algorithm which were omitted in Section 4 for brevity. The complete algorithm, Algorithm 3, expands on Algorithm 2. L is the set of links in the network in the increasing order of thei peering rate. The algorithm shows how CASCARA allocates flow to links in every timestamp of the billing period. The solution to this algorithm are link allocations in all time steps. CASCARA maintains a priority queue of links and the priority of a link is decided based on two factors:

- Initial priority: all links have their initial priority set to the number of free time slots they have in the current billing cycle. We update the priority after augmenting the link. In any subsequent billing timeslots, if the demand is higher than C_f, links with lower priority *i.e.,* ones which were used in the previous slots are re-used again. This ensures that the link augmentation is not spread across many links.
- Link capacity: We prefer to augment lower capacity links to save the higher capacity links for the remaining billing cycle. If the demand is too high, high capacity links are more likely to absorb it with augmentation.

We also kep track of the remaining free slots for each link. When all links have exhausted their free slots, allocation in that timestep fails and we have to increment C_f. Let O be the order in which links got augmented. An example ordering of augmented links, O is like so:

$$O = \{[l_1, l_2, l_3], [l_1, l_2,], ..[l_k, l_{k+1}, l_{k+2,..}]\}$$

In timestamp 1, CASCARA augmented allocations to links l_1, l_2 and l_3. The starting priority of links is the same, so the priority queue returns links in ascending order of their capacity. In the next timeslot, CASCARA attempts to meet the demand by augmenting the same set of links to keep allocations stable.

CASCARA initializes C_f to the minimum value that produced a feasible traffic allocation for the previous month. If C_f is too low for the current month's demands, despite augmenting allocation to links, the traffic demand would not be met and C_f will be incremented by β. The augmented link ordering of an infeasible allocation would be like so:

$$O_{unfeasible} = \{[l_1, l_2..], ..., [l_k, l_{k+1}, .., l_m]\}$$

where $\sum_k^m$ CAPACITY(l_i) $\leq demand - C_f$.

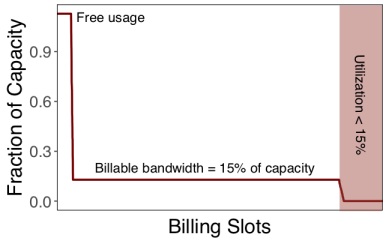
Additionally, CASCARA has a provision to proactively increment C_f by α (not shown in the algorithm). The goal is to proactively perform an inevitable increase in C_f to avoid wasting free slots of links. To do this, CASCARA checks if the number of links with free slots remaining is proportional to the amount of time left in the billing cycle. If the number of burstable links are too few,, C_f is incremented proactively.

Algorithm 3: Online Traffic Allocation (long version)

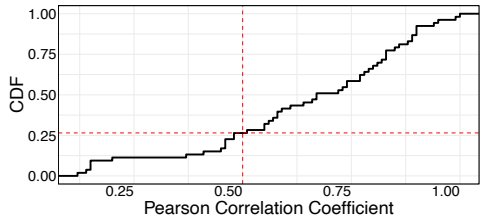
```
Result: Allocation of demand d in every timestamp t
Input: L, n, k, f, CAPACITY, C, α, β
Initialization:
freeslots = (k/100) * n
prio = freeslots        ▷ Initial priority of all links
linkq = PRIORITYQUEUE()
for link ∈ L do
    linkq.insert(link, CAPACITY(link), freeslots, prio)

Function allocate_timestep(d, f):
    link_alloc = {}
    augmented_links = []
    C_f = f * C        ▷ Fraction f of total capacity C
    if d ≤ C_f then
        link_alloc = bin_pack(L, C_f)
    else
        d = d − C_f
        while d ≥ 0 do
            b_link = linkq.pop()
            if !b_link then
                return {}
            augmented_links.add(b_link)
            if b_link ∈ L_1 then
                d = d − (1 - f) * CAPACITY(b_link)
            else
                d = d − CAPACITY(b_link)
            link_alloc [b_link]= CAPACITY(b_link)

        for link ∈ augmented_links do
            link.prio = link.prio − 1
            link.free_slots = link.free_slots − 1
    return link_alloc
End Function
while not allocate_timestep(d, f) do
    f = f + δ
```

A.3.1 Link utilization below the billable bandwidth

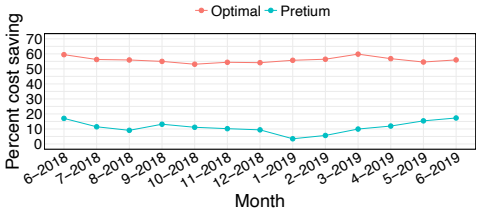
CASCARA chooses the *target* billable bandwidth (C_f) for a month. Given the billable bandwidth, it can be packed on to links by greedily assigning traffic ot cheaper links. [4] Given the minimum feasible C_f, this strategy is optimal. In fact, utilizing any link below its 95^{th} percentile utilization is uneconomical – the link gets charged at the 95^{th} percentile anyway. Figure 13a shows that while the utilization in some billing slots was below the 95^{th} percentile (shaded red), yet, the link was billed for 15% of its capacity, making the period of utilization below 15%, wasteful.



(a) Link utilizations.



(b) Pearson's correlation.



(c) Pretium vs. Optimal saving

Figure 13: (a) Utilization of a link as fraction of its capacity, sorted from high to low across billing slots in a month. (b) 95^{th} %-ile and avg. of top 10% correlation.(c) Cost saving by CASCARA vs. Pretium [20] on a month-by-month basis.

A.4 Details on the implementation of previous systems

We implement the optimization formulation from previous work using top 10% of utilizations in a month as the bandwidth cost of a link. We use CVXPY's implementation of sum of largest decision variables for this purpose. Since this formulation is a linear program, GUROBI solves it in less than a minute. While fast to compute, the allocations from this formulation are ineffective in saving the 95^{th} percentile cost. Figure 13c compares the cost savings per-month between our solutions from Algorithm 1 and previous work. We note that the cost saving from the sum of top-k formulation are modest, 11% on average for all instances. We believe this is because of two assumptions made by previous work:

Assumption 1: 95^{th} percentile of a link's utilization is linearly correlated with the *average of top k utilizations* [20]. We evaluate this assumption using the utilizations of over 50 peering links in a cloud WAN. These links connect the cloud WAN to large ISPs in N. America. We compute the Pearson correlation coefficient to measure the extent to which the average of top 10% utilizations can be used as a proxy for 95^{th} percentile utilization of inter-domain links. We find that the correlation coefficient for over 25% of the links is less than 0.5. Since previous work's hypothesis was derived from the data of a *single* WAN link measured a few years ago, the correlation between average of top 10% and 95^{th} percentile utilization may exist for some links but not all. Ever-changing traffic patterns from WANs due to the advent of new services like gaming also explain this difference.

Assumption 2: The correlation between average of top-k and 95^{th} percentile of a link's utilization holds even after a new traffic allocation scheme replaces the current one. There is no guarantee that assumptions about allocation distributions hold in a newly proposed traffic engineering scheme. In fact, traffic engineering schemes *change* the allocation of flow along network links, modifying how links are utilized.

[4] We have discussed the additional routing and client latency constraints on CASCARA in §5.

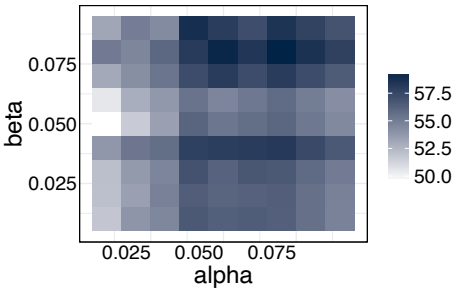


Figure 14: Average monthly bandwidth cost saving with CASCARA as a function of the parameters α and β. We choose the best values of α and β for the evaluation in §5.

A.5 Selecting CASCARA's hyperparameters

We sweep through potential values of α and β to find the ones that fit CASCARA the best. The range of values for α and β is $[0, 1]$ since they represent increments to fraction of total network capacity. We sweep the space in steps of 0.01 to find the parameters that lead to the highest cost savings in the average case across all billing cycles (Figure 14).