June 2021

Traffic engineering in planet-scale cloud networks

Rachee Singh
University of Massachusetts Amherst

TRAFFIC ENGINEERING IN PLANET-SCALE CLOUD NETWORKS

A Dissertation Presented

by

RACHEE SINGH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2021

College of Information and Computer Sciences

TRAFFIC ENGINEERING IN PLANET-SCALE CLOUD NETWORKS

A Dissertation Presented

by

RACHEE SINGH

Approved as to style and content by:

Prof. Phillipa Gill, Chair

Prof. Deepak Ganesan, Member

Prof. Brian Levine, Member

Dr. Victor Bahl, Member

Prof. James Allan, Chair of the Faculty
College of Information and Computer Sciences

DEDICATION

To my Mitchoo

ACKNOWLEDGMENTS

Before starting graduate school, I read advice about what it takes to be a good PhD student. One common thread in the advice about graduate school was, roughly paraphrased: "If you don't know exactly why you are in graduate school, it is highly likely that you will not finish." So, I started graduate school with the lingering fear that I will not finish because I did not know *exactly* what I wanted from it. In hindsight, I believe that the main reason I could get through is because my advisor, Phillipa, believed in me and supported me through the years. Her support was instrumental in helping me develop any confidence in my own ability to be a researcher. She is the enabling force behind this dissertation.

After two summer internships with Microsoft Research's (MSR) Mobility and Networking (MNR) Group, I got the unique opportunity to work on my PhD thesis while being a full-time researcher at MSR. I had the good fortune to seek feedback and discuss research with an exceptional group of computer scientists. I am very grateful for all the mentorship I have received from Nikolaj Bjørner, Jitu Padhye, Manya Ghobadi, Sharad Agarwal, Alec Wolman, Ranveer Chandra and others. Most of all, I benefitted from regular meetings with my manager and research mentor, Victor Bahl. I used these meetings to try different pitches for my work and improve them with Victor's feedback. Victor gave me cover from corporate churn to work on thesis research. I could not have found the time to finish this work without his support.

At times, the path forward in graduate school either felt murky or futile. In such times, I was lucky to find mentors who helped me navigate the existential crises. Rishab Nithyanand was my first mentor in graduate school. He encouraged me and believed in me even when I was new to research. Justine Sherry is one of the kindest people in the networking community. Her advice on picking research problems and strategies for an academic career helped me when I felt lost. Ratul Mahajan took time to chat with me when I was unsure how to start research at MSR. He helped me understand what it takes to be good researcher at MSR and collaborate with product groups. I thank Vern Paxson for giving me clarity about research progress and career paths in academia. I thank Brian Levine for his mentorship, advice and discussions in the security reading group. Preparing for the informal debate with Brian on the utility of anonymous communication networks was a very fun and informative experience for me. Deepak Ganesan's advice has shaped the ideas and approaches pursued in this dissertation.

I am lucky to have found a group of amazing friends in graduate school. I thank Jill for hunting good coffee and chocolate with me. Mina is the second person in my two-person research support group. Making espresso every day in the lab with Keen was the highlight of many days. Only Vaishnavi would take an afternoon off to hunt for salmonberries outside building 99 with me. I thank Akshay for being a tough paper critic and a coffee snob. Frank is the kindest and the most constructively critical friend I could have hoped for. Shinyoung, thanks for being there when I needed soup and support.

This thesis is dedicated to Mitchoo, my mother. She is the strongest, fiercest and nicest person I know. When life is tough, I think of how Mitchoo would make me some tea and say "*thodi aur koshish*" which translates to "try a bit more". My father is the source

of my ambition. His honesty, perseverance and thirst for knowledge continues to inspire me. This work is a result of the countless sacrifices my parents have made for me. I am lucky to have a loving sister who is probably more concerned about my dissertation submission than I am. I thank my brother for being a source of joy and encouragement. Most importantly, I would not have pursued a PhD had it not been for you, Em. Thank you for being the voice of reason when I am ready to give up close to a deadline. Thank you for believing in me and encouraging me to try new things.

ABSTRACT

TRAFFIC ENGINEERING IN PLANET-SCALE CLOUD NETWORKS

MAY 2021

RACHEE SINGH

B.E (Hons.), BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

MS, UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Prof. Phillipa Gill

Cloud wide-area networks (WANs) play a key role in enabling high performance applications on the Internet. Cloud providers like Amazon, Google and Microsoft, spend over hundred million dollars annually to design, provision and operate their WANs to fulfill the low-latency, high-bandwidth communication demands of their clients. In the last decade, cloud providers have rapidly expanded their datacenter deployments, network equipment and backbone capacity, preparing their infrastructure to meet the growing client demands. This dissertation re-examines the design and operation choices made by cloud providers in this phase of exponential growth along the axes of network performance, reliability and operational expenditure. I use empirical evidence from a large commercial cloud to

develop software-defined traffic engineering systems that remedy the inefficiencies in the operation of cloud networks.

First, I demonstrate how knowledge of optical signal quality can lead to a 75% increase in capacity for 80% of the optical wavelengths in the cloud backbone. I show that optical wavelengths can potentially sustain 175 Gbps or higher capacity but they were being utilized for a conservative 100 Gbps only, leaving 145 Tbps of network capacity on the table. This gain stems from the fact that operators have been conservative in utilizing the fiber out of concerns for network reliability. My analysis shows that by dynamically adapting link capacities, it is possible to have the best of both worlds – gains in network capacity with fewer link failures. I develop a traffic engineering controller for the WAN that dynamically adapts link capacities in response to changing optical signal quality. The rate adaptive wide-area network (RADWAN) controller reclaims terabits of network capacity from the existing cloud WAN infrastructure while preventing 25% of link failures.

Second, I demonstrate cost inefficiencies in the design of cloud optical backbones. Cloud providers traditionally operate point-to-point inter-regional networks —- where optical signals are converted to electrical signals and back at every geographical region. While flexible to accommodate new traffic demands, the conventional point-to-point design does not keep in view the nature of traffic demands and the traffic flow imposed by them. My analysis shows that on average, 60% of traffic traversing through WAN regions is *passing through* — neither originating nor terminating at the region. The pass-through or transit traffic undergoes wasteful optical-to-electrical-to-optical (OEO) conversions at all intermediate regions in point-to-point networks, occupying scarce optical line- and router ports. These ports contribute a majority of the cost of provisioning capacity in cloud

networks with existing fiber deployments. I design and implement Shoofly, a network design tool that minimizes hardware costs of provisioning long-haul capacity by optically bypassing network hops where conversion of signals from optical to electrical domain is unnecessary and uneconomical. Shoofly provisions bypass-enabled topologies that meet 8X the present-day demands using existing network hardware. Even under aggressive stochastic and deterministic link failure scenarios, these topologies save 32% of the cost of long-haul capacity.

Finally, I develop Cascara, a cloud edge traffic engineering controller to minimize the cost of inter-domain bandwidth incurred by the cloud provider. Traffic engineering systems at the cloud edge attempt to 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. I 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, Cascara optimizes inter-domain bandwidth allocations with non-linear pricing schemes. Cascara exploits the abundance of latency-equivalent peer links on the cloud edge to minimize costs without impacting latency significantly. Extensive evaluation on production traffic demands shows that Cascara saves 11–50% in bandwidth costs per cloud point of presence, while bounding the increase in client latency by 3 milliseconds.

CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

PUBLICATIONS RESULTING FROM THIS DISSERTATION

Chapter 1 combines work from two previous publications [76, 75]:

- Rachee Singh, Manya Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. *Run, Walk, Crawl: Towards Dynamic Link Capacities*. In ACM HotNets 2017.

- Rachee Singh, Manya Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. *RADWAN: Rate Adaptive Wide Area Network*. In ACM SIGCOMM 2018.

Chapter 2 revises a previous publication [74]: Rachee Singh, Nikolaj Bjørner, Sharon Shoham, Yawei Yin, John Arnold, Jamie Gaudette. *Cost-effective capacity provisioning in wide-area networks with Shoofly*. In ACM SIGCOMM 2021.

Chapter 3 revises a previous publication [73]: Rachee Singh, Sharad Agarwal, Matt Calder, Paramvir Bahl. *Cost-effective Cloud Edge Traffic Engineering with Cascara*. In USENIX NSDI 2021.

CHAPTER 1

INTRODUCTION

The last decade has seen a large-scale commercialization of cloud computing and the emergence of global cloud providers like Amazon, Google and Microsoft. Today, cloud networks occupy a central position in the Internet ecosystem due to the large volume and variety of popular content they serve to users. The cloud has enabled bandwidth-intensive and latency-sensitive applications (*e.g.,* real-time video, cloud gaming) that require strict performance and reliability guarantees. To achieve high performance and reliability while keeping pace with the exponential growth in demand, cloud networks have made several *ad-hoc* design and operation decisions over the last decade.

In this dissertation, I re-evaluate the design and operation decisions of cloud networks through a cross-layer empirical analysis of the wide-area network (WAN) of a large commercial cloud provider. These design decisions have determined the operating point of the cloud network on several trade-offs between desirable network characteristics. Of these, I quantify three fundamental trade-offs: (1) reliability vs. throughput (2) latency vs. operation cost (3) throughput vs. operation cost (Figure 1.1 summarizes the contributions). I develop centralized traffic-engineering (TE) solutions to strike the right balance on these trade-offs. My thesis is that:

Centralized traffic-engineering is the key to enable efficient wide-area networks.

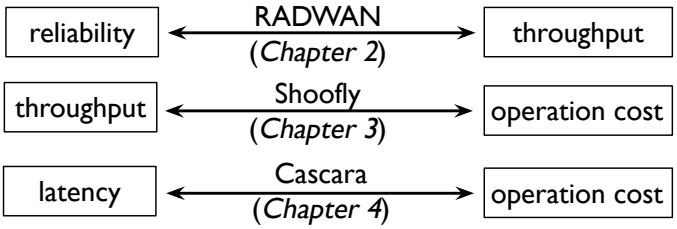


Figure 1.1: Cost and performance trade-offs studied in this dissertation.

I describe the contributions towards my thesis (§1.1) and their adoption in production cloud networks (§1.2).

1.1 Key Contributions

This dissertation develops centralized traffic engineering algorithms to navigate the following trade-offs in the design and operation of cloud WANs:

Reliability vs. throughput. Fiber optic cables connecting datacenters form the backbone of the cloud network. Optical link failures degrade the network capacity and hamper the network's ability to meet client demands. Thus, operators provision cloud backbones to avoid optical failures at any cost. This risk-averse mindset has led to a conservative approach towards utilizing fiber, the most expensive asset of the cloud backbone. As per this approach, each wavelength of light on fiber is *statically configured* to carry 100 Gbps of traffic. The minimum optical signal quality needed to support 100 Gbps of error-free transmission on fiber is 6.5 dB. I analyze the optical signal quality of wavelengths on fiber in a cloud backbone and find that the optical signal quality of over 75% of wavelengths is high enough to sustain up to 175 Gbps of capacity — 75 Gbps higher than their current conservative usage.

To harness capacity proportional to the optical signal quality from fiber links, network operators must set a higher SNR threshold for signals on fiber *e.g.,* 10 dB to support 150 Gbps of capacity. A potential consequence of setting higher thresholds is that dips in SNR are more likely to land below the threshold, causing link failures. Thus, higher SNR thresholds enable higher link capacities but risk lower link availability. To navigate this trade-off between throughput and reliability, I propose to *dynamically* adapt SNR thresholds of signals in response to the changing optical signal quality. This can be achieved by adjusting the modulation format of optical signals using off-the-shelf optical transceivers called *bandwidth variable transceivers* or BVTs.

This intuitive solution to harness capacity from over-engineered fiber links is challenging to implement in practice due to hardware limitations. BVTs in production networks can re-configure link capacities only after one minute of link downtime. I perform experiments with the evaluation board of a state-of-the-art BVT and find that the cause of link downtime during capacity re-configuration is the time taken in power cycling the laser. This limitation of the hardware can not be resolved without engineering and manufacturing effort from transceiver vendors. Therefore, I develop a *software-defined WAN traffic engineering controller*, called the rate adaptive WAN controller (RADWAN), that incorporates the limitation of the hardware to dynamically adapt link capacities.

RADWAN engineers traffic to links in the WAN while adapting link capacities *only when necessary* to reduce the link downtimes associated with capacity reconfigurations. A capacity reconfiguration is *necessary* in two cases. First, if the demand is so high that the current capacity configuration cannot sustain it. In this case, the increased demands can be absorbed by increasing the capacity of signals with high SNR. Second, if the SNR

of a signal falls below its current threshold, it is necessary to lower the signal's capacity to prevent it from failing altogether.

We develop a hardware testbed that mimics a WAN with four datacenters, connected with thousands of kilometers of fiber to evaluate RADWAN. Our experiments show that RADWAN can achieve 40% higher network throughput by adapting link capacities when the demand is too high to meet with the existing capacity configuration. Additionally, when the signal quality on fiber dips, RADWAN adapts to a lower capacity, preventing 25% of link failures in the cloud network (Chapter 2).

Throughput vs. operation cost. Cloud providers traditionally provision *point-to-point* optical backbone networks where the fiber and optical equipment (*e.g.,* amplifiers, terminals) connects to electrical equipment (*e.g.,* routers and switches) as shown in Figure 1.2. In point-to-point optical backbones, optical signals on fiber are converted to electrical signals and back at every geographical region (*e.g.,* US-east, US-central). Thus, inter-regional traffic undergoes optical-to-electrical-to-optical (OEO) conversion at all intermediate regions on the path towards its destination. For instance, wavelength λ undergoes an OEO conversion at the first router in Figure 1.2. This backbone design is backed with conventional wisdom that point-to-point backbones offer control and flexibility at the network layer, keeping the optical layer uncomplicated and easy to manage. This flexibility allows the network to accommodate emergence of new and unexpected traffic demands.

However, traffic in the cloud WAN has evolved over the years and settled into predictable patterns. These patterns suggest that the point-to-point backbone design is cost-inefficient when scaling network capacity. I analyze inter-regional traffic patterns in the backbone of a large commercial cloud provider and find that 60% of traffic traversing

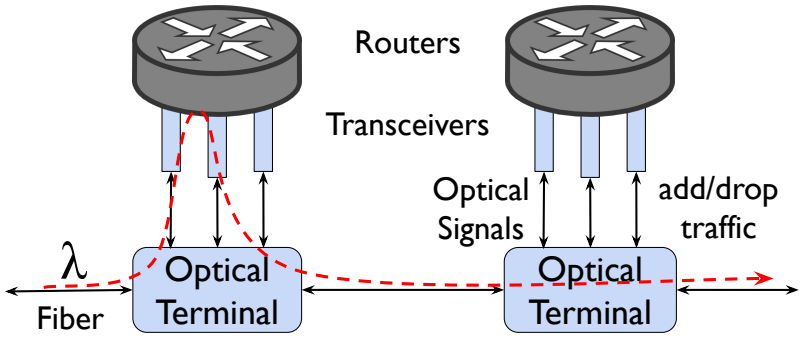


Figure 1.2: Optical (blue) and electrical (grey) equipment in cloud backbones.

through 30% of geographical regions in the WAN is *passing through* – neither originating nor terminating at the region. The pass-through or transit traffic undergoes wasteful OEO conversions at all intermediate regions in point-to-point networks, occupying scarce optical line- and router ports. These ports contribute a majority of the cost of provisioning capacity in cloud networks with existing fiber deployments.

I propose to minimize the hardware cost of provisioning long-haul capacity by removing wasteful OEO conversions in the WAN. We refer to the elimination of an OEO conversion at a region as optically bypassing the region *e.g.,* λ bypasses the second router in Figure 1.2. A key challenge is that optical bypass of geographical regions forces the wavelengths to travel longer distances on fiber before they can be regenerated. Signals traveling longer distances undergo more dispersion and attenuation leading to lower optical signal quality. Lowered signal quality can necessitate lower link capacity. Thus, operators can reduce the cost of provisioning WAN capacity by suffering a reduction in the achievable throughput of links. To quantify this trade-off, I analyze the SNR on fiber and find that the signal quality of 75% of optical channels on fiber is sufficiently high to sustain trans-

mission over longer distances without reduction in their data rates. The remaining signals must downgrade their data rates to traverse longer fiber paths without regeneration.

I develop Shoofly, an optical backbone design tool that formulates the problem of identifying optical bypass opportunities in a cloud network with the goal of minimizing the hardware costs of long-haul capacity. The Shoofly-proposed backbone topology can reduce the hardware cost of long-haul capacity by 40% while continuing to meet up to 8X the present-day demands using existing hardware and fiber deployments. Longer fiber paths can increase the likelihood of link failures. Shoofly integrates resilience to stochastic and deterministic link failures while identifying optical bypass opportunities. This ensures that the bypass-enabled topologies proposed by Shoofly are just as resilient to link failures as the present-day network. (Chapter 3).

Latency vs. operation cost. In the final chapter of this dissertation, I focus on the operation of the cloud edge that interfaces the cloud with the rest of the Internet. At the edge, the cloud network *peers* with thousands of autonomous systems on the Internet. The peering interconnections of the cloud provider have terabits of total capacity and their operation is governed by private legal agreements between the cloud provider and ISP networks. These agreements set the rate at which exchanged traffic, referred to as inter-domain traffic, is billed by the ISPs. Inter-domain traffic costs comprise a significant amount of the operating expenditure of cloud providers.

Traffic engineering systems at the cloud edge attempt to strike a fine balance between minimizing operation costs and maintaining the latency expected by clients. The nature of this trade-off is complex due to non-linear pricing schemes prevalent in the market for inter-domain bandwidth. I quantify this trade-off and uncover several key insights from the

link-utilization between a large cloud provider and Internet service providers. Based on these insights, I develop Cascara, a cloud edge traffic engineering controller to minimize the cost of inter-domain bandwidth.

Bandwidth allocations at the cloud edge impact both the client latency and inter-domain bandwidth costs of the cloud provider. Thus, Cascara's traffic allocations balance both goals — minimizing bandwidth costs and maintaining low client latency. To achieve this, Cascara leverages the cloud provider's rich diversity of *latency-equivalent* BGP peers to offer cheaper options to outbound traffic. Through extensive experiments I demonstrate that Cascara provides near-optimal cost saving in practice and can be deployed safely and incrementally in cloud WANs (Chapter 4).

1.2 Impact on production networks

The solutions proposed in this dissertation were inspired from discussions with hardware manufacturers (*e.g.,* BVT vendor Acacia Inc., major switch vendor Arista Networks) and cloud network operators. Their feedback has informed the design choices in the systems proposed in this dissertation to ease the incorporation of our ideas in large-scale production networks. In fact, the three main contributions of this dissertation have either been adopted or are on a path towards adoption in the production network of a large commercial cloud provider.

RADWAN. The analysis on rate-adaptive links has led the cloud provider to re-think the static thresholds for SNR used in production. While deploying dynamic capacity reconfigurations will take significant engineering effort from the cloud operators, simply changing the thresholds to gain more capacity from the over-engineered fiber links is an important first step to enable part of RADWAN's throughput gains without sacrificing link reliabil-

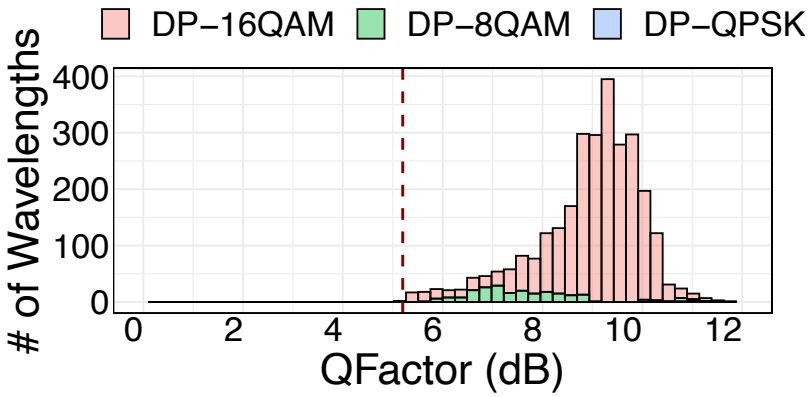


Figure 1.3: Inspired from RADWAN's findings, a large commercial cloud provider now harnesses 200 Gbps, 150 Gbps and 100 Gbps capacity (for DP-16QAM, DP-8QAM, DP-QPSK modulation formats) from fiber links based on the SNR of signals instead of operating all links at 100 Gbps.

ity. The cloud provider has deployed this change to the fiber links. Today, majority of the wavelengths in the cloud WAN operate at 200 Gbps (modulation format of 16-QAM), resulting in the gain of over hundred terabits of backbone capacity as shown in Figure 1.3.

Shoofly. Based on the insights from Shoofly, the cloud provider plans to enable optical bypass in their network in the coming year. Implementing bypasses in the cloud WAN requires modifications to the physical connections between optical terminals and routers. I quantify the logistical burden of provisioning the Shoofly proposed bypasses on cloud operators. My analysis shows that Shoofly's logistical burden is low — 25% of optical bypasses achieve 80% of the cost savings.

Cascara. Since the development and publication of Cascara, the inter-domain bandwidth costs incurred by the cloud provider have been rising. In fact, as the global pandemic hit in March 2020 and work-from-home orders were issued in different parts of the world, the cloud provider saw a sharp increase in edge traffic demands. This led to over 40% increase in inter-domain bandwidth costs. Due to this, cloud network operators plan to

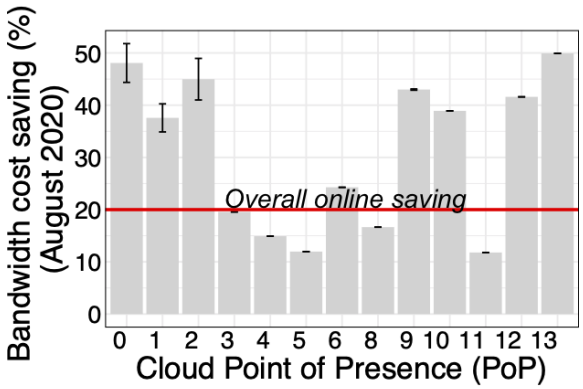


Figure 1.4: shows cost savings with Cascara when engineering traffic on a per-PoP basis while limiting the impact on client latency to 3 ms in the worst case (mean and standard deviation computed over three runs of Cascara).

deploy Cascara incrementally, starting with one point-of-presence (PoP), to engineer non-interactive traffic at the cloud edge. The choice of PoP for deploying Cascara is derived from our experiments (Figure 1.4 from Chapter 4) that show some PoPs have significantly higher cost-saving potential than others.

1.3 Centralized-traffic engineering enables efficient WANs

This dissertation re-evaluates a decade of work on the design and operation of cloud networks using rigorous empirical analysis. The main contributions of this dissertation show that centralized traffic engineering is an important tool for enabling cost and performance efficiency across the layers of the networking stack. I have leveraged centralized TE to provision rate adaptive networks (RADWAN), reduce cloud operation costs (Cascara) and provision long-haul capacity (Shoofly), demonstrating the versatility of this tool. The increasing adoption of software-defined technology in production WANs [87, 72, 46] will continue to ease the deployment of centralized traffic engineering systems for new use cases in the WAN *e.g.,* like build planning, compute placement *etc.*

CHAPTER 2

RATE-ADAPTIVE WIDE-AREA NETWORKS

Optical backbones are million-dollar assets, with fiber comprising their most expensive component. Companies like Google, Microsoft, and Facebook purchase or lease fiber to support wide-area connectivity between distant data center locations but have not been able to fully leverage this investment because of the conservative provisioning of the optical network. We show that wide area fiber links exhibit significantly better signal quality (measured by the signal-to-noise-ratio or SNR) than the minimum required to support transmission at 100 Gbps, leaving money on the table in terms of link capacities.

In other words, there is potential to operate fiber links at higher capacity, thereby increasing the throughput of existing optical networks. We analyze historical SNR from 8,000 optical channels in a backbone network and find that the capacity of 64% of the links can be augmented by 75 Gbps or more, leading to a capacity gain of over 134 Tbps in the network. However, we argue that simply raising link capacities to a higher value (*e.g.,* 150 Gbps or 200 Gbps) increases the rate of link failures because the signal quality fluctuates and operating near the SNR threshold makes links susceptible to failure.

Moreover, enforcing a static link capacity forces operators to treat link failures as *binary* events: when the SNR of a link falls below a static threshold, the link is treated as "down." We show this is wasteful, as at least 25% of current failures can be mitigated by reducing the rate of transmission from 100 Gbps to 50 Gbps.

At the core of these issues is a fundamental orthodoxy in the operation of wired networks: a fiber link is either up with a fixed capacity or it is down, largely oblivious to changes in the quality of the underlying optical signal. In this school of thought, operators are forced to account large margins between the actual SNR and the operating capacity if they want to avoid frequent link failures. In contrast, wireless networks employ a variety of schemes to adapt the transmission rate in response to changing signal quality [67, 8, 85]. However, adapting transmission rates to the wireless channel quality is difficult, as the quality can vary at time-scales shorter than a single packet transmission time [85]. In addition, obtaining accurate measurements of received signal strength indication (RSSI) of wireless media (a proxy for true SNR) is difficult in practice [44] because of issues like miscalibration and packet corruption.

We argue that optical links are well positioned to be rate-adaptive. First, signal quality varies at a much coarser time granularity in fiber than in wireless media (hours as opposed to milliseconds). This stability can be leveraged in wide area networks to amortize the cost of infrequently shifting between multiple discrete modulation schemes as signal quality changes. Second, unlike wireless signals, optical signal quality is easily inferred from the bit-error rate (BER) reported after forward error correction (FEC). Leveraging these benefits, we present RADWAN (Rate Adaptive WAN), a system that adapts channel bitrates in WANs to improve the overall throughout and availability of the network.

RADWAN consists of a centralized rate-adaptive WAN controller that gathers SNR from all fiber channels in the network to adjust the modulation format of the channels to achieve higher or lower data rates. In traditional wide area settings, the QPSK modulation format supports data rates of 100 Gbps for distances up to 3,000 km, 8QAM allows

150 Gbps for distances up to 2,100 km, and 16QAM allows 200 Gbps for distances up to 800 km (see §4.5 for a discussion on distance). By switching links to a lower modulation format (e.g., BPSK with data rates of 50 Gbps), RADWAN allows critical WAN links to function at lower data rates instead of failing altogether. We refer to these variable capacity links in RADWAN as *dynamic capacity links*. By building on top of existing software-based WAN controllers [46], RADWAN allows traffic engineering schemes to exploit dynamic capacity to improve network throughput. We make two key contributions to make rate adaptive WANs practical:

Optimal WAN traffic engineering. A major challenge associated with dynamically adapting link capacities in WANs is the latency incurred by network hardware when changing a link's modulation format. To reconcile the latency of capacity changes with the benefits of adapting link capacities in WANs, the RADWAN controller re-formulates the centralized traffic engineering optimization problem to avoid unnecessary capacity reconfiguration (§2.3). We evaluate our controller by comparing the throughput gains of employing RADWAN at scale to those of a state-of-the-art controller. Our results show that in a real-world network topology and with conservative traffic churn settings, RADWAN improves the overall network throughput by 40% while also improving the link availability (§2.5). We estimate that RADWAN lowers the dollar per gigabit per second cost of traffic by 32% (§4.5).

Avoiding high latency of modulation reconfiguration. We build a testbed emulating a WAN connecting four data centers via 1,540 km of fiber. Using this testbed, we confirm the viability of modulation reconfiguration to achieve greater network throughput. We benchmark the behavior of the RADWAN controller as it reacts to SNR degradation

by switching to a lower modulation format. During the modulation change, the line-rate traffic on the affected link is migrated to a backup path until the modulation change is complete (§2.4). Our experiments show that reconfiguring modulation formats on commodity hardware incurs a latency of 68 seconds, on average. We develop a prototype that demonstrates the feasibility of decreasing this reconfiguration time by a factor of 1,000 (§2.6.1).

RADWAN opens the door to revisiting several classical networking problems in light of dynamic capacity links. For instance, are there graph abstractions that can capture networks with dynamic capacity links? How do classical networking algorithms (such as the maximum-flow problem [33]) change in the presence of variable link capacities? Are there smart capacity planning, failure-recovery, load-balancing, or on-demand bandwidth allocation algorithms that can benefit from rate adaptive links? RADWAN prepares the ground for thinking about these problems.

2.1 Quantifying the opportunity

We investigate the signal quality of 8,000 optical channels in a large optical backbone network. Our dataset consists of the average, minimum, and maximum SNR per channel, aggregated over 15 minute intervals for $\approx$ three years (Feb. 2015 - Dec. 2017). We characterize the SNR of these channels and quantify its variations. In wireless networks, signal quality may vary in short time intervals and estimating SNR is complicated by signal interference [85], but signals in fiber optical media do not face these challenges.

Impact on capacity. We note that in our data, the average SNR is much higher than the required threshold for operating links at their current rate of 100 Gbps. Figure 2.1

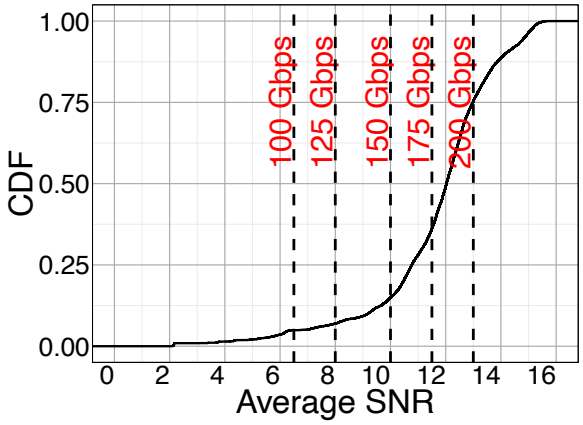


Figure 2.1: Distribution of the average SNR of over 8,000 channels in a backbone network for three years. Note that the SNR of optical channels is much higher than the required SNR for a 100 Gbps bit rate (6.5 dB).

shows the distribution of the average SNR of all 8,000 channels, with vertical dashed lines marking the SNR threshold for various rates. The figure shows that 95% of the channels have an average SNR above the required threshold for 100 Gbps. Even better, 64% of the channels have an SNR that can support data rates of 175 Gbps or higher but are currently used for a conservative 100 Gbps only. This represents a significant opportunity to improve the throughput of optical links by operating closer to the actual SNR of the signal.

But what about stability? While the average SNR may be well above what is needed to drive links at 100 Gbps, simply increasing the links' rate to a higher value, say 150 Gbps, will not work in practice because SNR fluctuates, as illustrated in Figure 2.2. The figure shows the SNR of 40 channels on one fiber cable observed over 2.5 years. We note that the SNR of these channels is largely stable, but there are occasional dips caused by impairments in the fiber or other optical hardware. The frequency and duration of these dips vary for different fibers in the network. The dashed horizontal lines in Figure 2.2

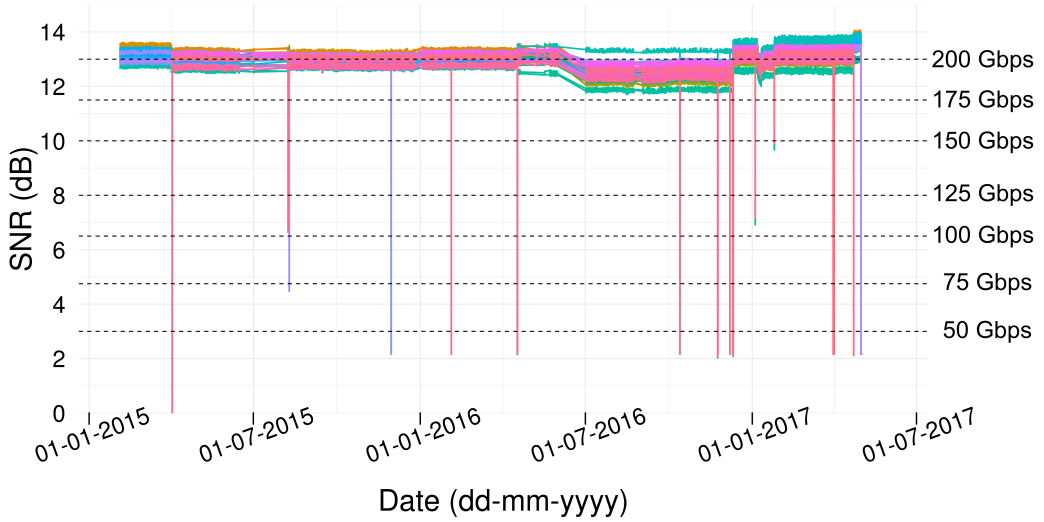


Figure 2.2: SNR variations in 40 optical channels (i.e., IP links) on a wide area fiber cable. Dotted lines represent the feasible link capacity for a particular SNR.

show the required threshold for various data rates. Higher SNR means we can have higher data rates.

We further consider the variability of SNR across all links for different time-scales. For each time interval of size 15 minutes, 10 hours, 1 day and 1 week, we calculate the variability of SNR (the difference between maximum SNR and minimum SNR) for all optical channels in the backbone network. Figure 2.3 shows the distribution of SNR variation in time intervals of different sizes. We confirm that SNR remains stable over several hours at a time. A small fraction ($\leq 5\%$) of links show a variation of over 1 dB in the 10 hour interval. Moreover, although our SNR measurements are aggregated over 15 minute intervals, we argue that our conclusions are sound, as Figure 2.3 shows negligible variation in SNR for 15 minute time intervals. This contrasts with wireless media where significant SNR changes can happen within a few milliseconds.

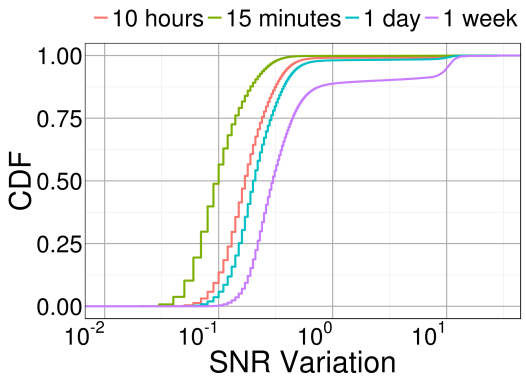


Figure 2.3: Variations in the channel SNR in intervals of different durations. Observe that most links do not observe significant variation in SNR for several hours.

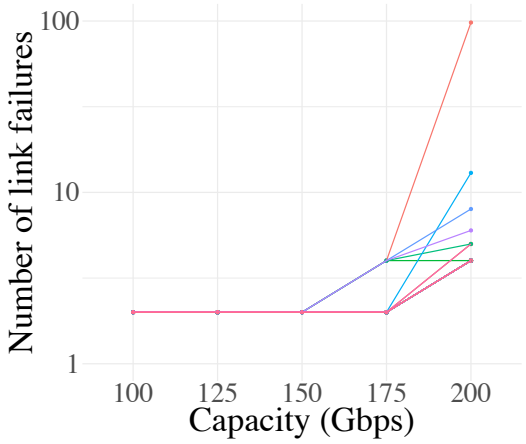


Figure 2.4: Number of link failures for 40 links (one color per link) for a given capacity. For this particular fiber, while increasing capacity up to 175 Gbps does not increase link failure events, achieving 200 Gbps capacity comes at the cost of increased link failures.

Why do we need variable bandwidth links? Based on our observation of the mostly stable but over-provisioned SNR of links, one might be tempted to operate links closer to the actual SNR by simply making a one-time decision to increase the transmission rate of all links. However, we find that the frequency of link failures increases if we cannot dynamically adapt to SNR changes. This is because infrequent but sizable variations in SNR occur in fiber links. While the SNR of a small fraction of links changes significantly in a few hours, 10% of all links undergo 2 dB of change in SNR within a week (Figure 2.3). To illustrate this, we select a fiber where the SNR of each link (i.e., optical channel) is high enough to make all capacity denominations feasible over three years. We then analyze the number of failures the links would undergo if they were modulated with higher but static capacities. Figure 2.4 shows that links on this fiber do not see a significant increase in the number of failures as the capacity is increased up to 175 Gbps, but some would have up to 100 failures if driven at 200 Gbps. We find this behavior repeated in other fibers, but depending on the number of links, fiber length, technology, and age of equipment, the point at which the failures start to increase differs for each fiber and for each channel on the fiber. Hence, it is impossible to select a one-size-fits-all static capacity that is higher than 100 Gbps.

Next, we characterize the duration of SNR dips to evaluate the magnitude of disruption they could cause if we choose a higher modulation (hence higher bandwidth). Figure 2.5 plots the duration of link failures for the various modulated bandwidths (based on the link's average SNR). We observe that such SNR dips last for several hours which means we cannot simply select a static modulation and dismiss the SNR dip events. The good

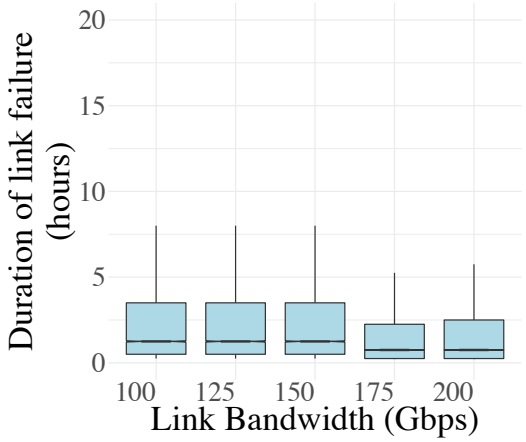


Figure 2.5: Duration of failures if WAN links operate at a given capacity.

news is that by enabling variable bandwidth links, we can react to SNR dips by changing the bandwidth to match the SNR.

Impact on availability. Today, when the SNR of a link's optical signal drops below its pre-determined threshold, the link is declared down. However, not all failures are complete loss-of-light. SNR drops may be caused by planned maintenance work (e.g., a line card replacement) or unplanned events (e.g., fiber cut, hardware failure, human error). While some of these impairments make the link unusable (e.g. fiber cuts), others may simply lower the signal quality (e.g. degradation of an amplifier) without completely shutting off the signal. Links undergoing failures due to lowered signal quality can still be used to send traffic at a reduced rate, highlighting another opportunity to improve link availability.

To define the opportunity area, we record the lowest SNR of failure events (when the SNR falls below the 100 Gbps threshold which is 6.5 dB). Figure 2.6 shows the distribution of lowest SNR values at link failures. We observe that in 25% of the failures, the lowest SNR is above 3 dB, enough to drive a link at 50 Gbps capacity. Therefore, 25% of the link

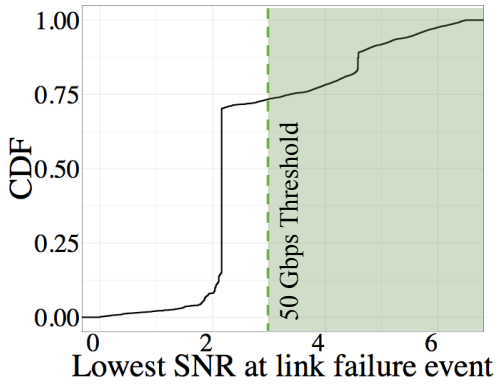


Figure 2.6: Distribution of the lowest SNR values when a link failure event happens. The lowest SNR is above 3.0dB (sufficient to drive a link at 50 Gbps) 25% of the time.

failures could have been avoided by driving the impacted links at 50 Gbps, indicating the improvement in availability offered by dynamic capacity links.

2.2 Dynamic Capacity Links

Our characterization of the SNR values of optical links suggests they are currently operating well below their potential transmission rates. However, operating links at constant transmission rates closer to the observed SNR increases the likelihood of link failures. To balance this trade-off, we propose a dynamic adjustment of physical link capacities in *centrally controlled wide area networks* by changing the *modulation format* of optical signals. These choices are motivated by the latest hardware and software developments in the industry:

Adapting bit-rates by changing the modulation. Recent advances in the development of bandwidth variable transceivers (BVTs) provide a promising first step towards increasing the network bandwidth and improving availability by decreasing transmission rates in

the face of low SNR (vs. incurring a link failure). State-of-the-art BVTs are capable of modulating signals on the fiber with three different formats: 16QAM, 8QAM and QPSK. All other factors being constant, signals in 16QAM format can carry traffic at 200 Gbps, 8QAM can carry 150 Gbps and QPSK can carry 100 Gbps. However, these transceivers were designed with the assumption that operators would make a one-time choice of modulation format.

This is reflected in the latency incurred in changing the modulation of ports on modern Arista 7504 switches. In our experiments, we find that on average, changing the modulation of a port incurs a latency of over one minute. During this time, the link undergoing the modulation change is down and cannot carry traffic. This is because of the assumption by the manufacturers that the modulation change is a one-time event. To benchmark the reconfiguration latency, we experiment with a transceiver evaluation board and investigate ways of reducing capacity reconfiguration time (Section 2.6.1). We note that it will take significant engineering efforts to make hitless capacity change production ready for use.

Software Driven WANs. Effective utilization of network infrastructure in modern WANs is enabled by software-driven centralized traffic engineering (TE) [49, 46, 61] that maximizes the network throughout for changing demand matrices. Therefore, we consider the implementation of dynamic capacity links in such networks. We note that throughput maximization is one possible goal of traffic engineering. Previous work has formulated TE to achieve optimal social welfare [50], meet deadline-sensitive transfers [56], and provide improved guarantees for high priority traffic [46].

TE controllers are consumers of network link capacities, as they make decisions about routing flows along the best paths with available capacity. Had link capacity reconfigura-

tion been a hitless phenomenon, existing TE controllers could largely function unmodified with dynamic capacity links. However, capacity reconfiguration is expensive, as it causes a link outage lasting for over a minute. We discuss the impact of this additional constraint on TE controllers in the next section.

2.3 Traffic Engineering with Dynamic Capacity Links

In a network with dynamic capacity links, the state of the network in each run of the TE optimization algorithm is dependent on the links' underlying SNR. Therefore, TE controllers must be modified to gather the SNR of all links in the network and to treat the link capacities as variables. Our proposed RADWAN centralized TE controller can leverage dynamic capacity links to achieve higher network throughput and availability. RADWAN handles a spike in the demand matrix by upgrading the capacities of one or more links. However, state-of-the-art bandwidth variable transceivers (BVTs) require over a minute to change the capacity of a link (§ 2.6.1), rendering the link unusable for that period. In response to this link flap, existing traffic flows must be migrated away from the link undergoing capacity reconfiguration, but such flow migrations can cause transient congestion in the network and must be done minimally.

Therefore, we argue that in a network composed of dynamic capacity links, the objective of traffic engineering changes from simply maximizing the network throughput to maximizing throughput *while minimizing churn caused by link capacity reconfigurations*. In Section 2.3.1 we discuss how network churn can be quantified to achieve low disruption while meeting traffic demands via link capacity reconfiguration.

2.3.1 Quantifying network churn

Current hardware does not support hitless capacity changes; therefore, we propose dealing with churn induced by link capacity changes in software. As a first step, we introduce a definition of churn induced by a link capacity change in terms of the rate of traffic on the link. The capacity change (either an increase to meet demands or a decrease due to lowered signal quality) of link l carrying f_l units of traffic will displace f_l units. The displacement of large flows is more likely to cause transient congestion as opposed to smaller flows. Therefore, we define churn induced by the capacity change of link l as:

$$churn(l) = f_l \tag{2.1}$$

The overall churn induced by capacity changes in a network, C, is a summation of the churn from each link undergoing capacity change:

$$C = \sum_{\text{links}} churn(l) \tag{2.2}$$

We note that this is only one of many possible ways to define the churn caused by link flaps in the network. We encourage practitioners to consider other definitions to reduce the churn of preferred traffic classes (e.g., interactive traffic over background traffic).

2.3.2 Computing flow allocations

When computing allocations of flows along different paths in a network composed of dynamic capacity links, the goal of RADWAN is to maximize the network utilization (as was the case with earlier work [46, 49, 61]) while keeping churn due to capacity reconfigurations minimal. In this section, we formulate this goal as a constrained optimization

problem using the definition of churn from Section 2.3.1. RADWAN periodically evaluates the optimization goal to assign traffic flows along network paths. In each round of its operation, RADWAN has access to attributes of the network state which serve as input to the optimization problem. We now describe various elements of the RADWAN controller.

Inputs. Traditional TE controllers take *network topology* and *traffic demand matrix* as input to compute allocations of flows along label-switched network paths. In addition to these, our controller requires SNR measurements for all physical links in the network. Using this information, the controller derives the *potential* capacity of each link, over its existing capacity.[1] Implicitly, the controller is also aware of the existing flow on all links in the network, assigned in the previous round of controller operation.

Allocation Objective. Algorithm 1 describes the optimization goal of RADWAN. At its core, the optimization is a modified multi-commodity flow that maximizes overall throughput of the network while augmenting link capacities minimally. The optimization variables $b_{i,j}$ specify the allocation of flow i along path j in the network. Allocation of flow along link l in the network is constrained by the sum of the link capacity (c_l) and the potential increase in capacity (p_l) depending on the link's SNR. ϵ is a small positive constant denoting the relative importance of the two aspects of the objective function: maximizing throughput and minimizing churn. Finally, in a given round, the network churn caused by the capacity change of link l is 0 if the optimal flow assigned to the link is less than or equal to the link's capacity (c_l). However, if the link has more flow assigned to it than its current capacity, it induces network churn equal to the amount of traffic on

[1]Even if there is potential to increase a link's capacity by, say, 50 Gbps, the controller must do an upgrade only if this extra capacity is needed to meet traffic demands.

Algorithm 1: Traffic Engineering Optimization

Inputs:

d_i: flow demands for source destination pair i

c_l: capacity of each link l

p_l: potential capacity increase of each link l

$I_{j,l}$: 1 if tunnel j uses link l and 0 otherwise

f_l: existing flow on link l ($f_l \leq c_l$)

T_i: set of tunnels set up for flow i

Outputs:

$b_i = \sum_j b_{i,j}$: b_i is allocation to flow i

$b_{i,j}$ is allocation to flow i along tunnel j

Maximize: $\sum_i b_i$ - $\epsilon(\sum_l \text{churn}(l))$

subject to:

$\forall i, 0 \leq b_i \leq d_i$

$\forall i, j, b_{i,j} \geq 0$

$\forall l, \sum_{i,j} I(j,l) b_{i,j} \leq c_l + p_l$

$\forall i, \sum_{j \in T_i} b_{i,j} \geq b_i$

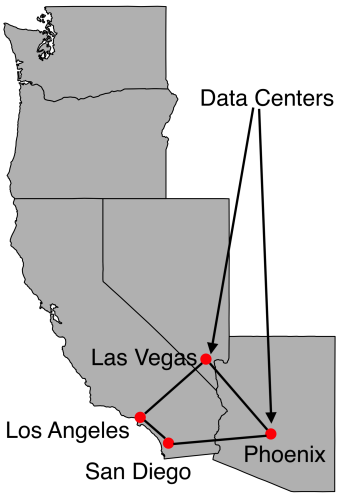
$$\text{churn}(l) = \begin{cases} 0, & \sum_{i,j} b_{i,j} I(j,l) \leq c_l \\ f_l, & \text{otherwise} \end{cases}$$

it (f_l), as assigned in the previous round of flow allocation. The nature of network churn makes the objective function of the optimization piece-wise linear.

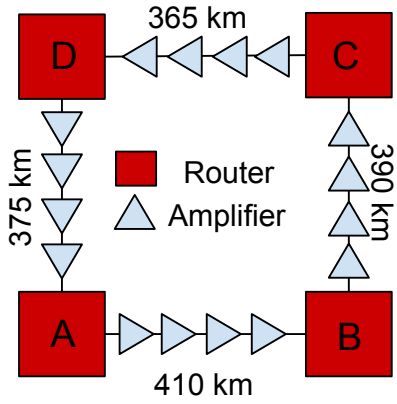
Approximation to Linear Program. To efficiently solve the optimization objective described in Algorithm 1, we approximate the definition of churn as:

$$\text{churn}(l) = \max(0, (\sum_{i,j} b_{i,j} I(j,l) - f_l)) \tag{2.3}$$

This monotonically increasing value of churn, depending on the flow assignments $b_{i,j}$, is different from the actual churn value which is essentially a step function; however, this reasonable approximation allows us to convert Algorithm 1 to an efficiently solvable linear program.



(a)



(b)

(c)

Figure 2.7: (a) Geographic scale of the testbed built to demonstrate the operation of RADWAN. Our testbed emulates a WAN connecting four major cities on the west coast of the United States. (b) Logical view of the testbed where four routers (logically split from a modular chassis switch) emulate four data centers. These routers are connected via hundreds of kilometers of optical fiber and regularly spaced amplifiers. (c) Photograph of the electrical and optical equipment in our testbed.

Managing Churn. For the duration of a link flap, no traffic can be routed along this link. As the impacted links will be offline for just one minute, the affected traffic (churn) has to be managed efficiently to ensure low disruption. We thus compute a single intermediate flow allocation, where the churn is distributed along routes without link flaps. We show in Section 2.5 that a single intermediate step suffices, as the number of link flaps per reconfiguration is low in practice (see Figure 2.14). Methods for networks with highly unstable SNR are described in Section 2.6.1. Once hitless capacity reconfiguration is production ready, the intermediate flow allocation step described in this paragraph can be omitted.

Importance of ϵ. The ϵ parameter defines the balance between RADWAN controller's tendency to maximize network utilization and minimize network churn due to capacity reconfigurations. We encourage operators to use a value of ϵ that captures their willingness towards capacity reconfigurations. We note that future optical equipment that offers reduced capacity reconfiguration time will make capacity changes more attractive and operators can use smaller ϵ values in the optimization to reflect increased willingness for capacity reconfiguration.

2.3.3 Controller Implementation

We implement RADWAN, the traffic engineering controller based on the goals outlined in the previous subsection. The controller implements Algorithm 1 using the popular optimization library CVXPY [23] in Python 2.7.

RADWAN computes flow allocations for the input demand matrix in each round of its operation. Before solving the optimization, RADWAN uses the link-level SNR information to determine: (i) links for which the total capacity must now be reduced since the

new SNR is too low to support the existing capacity. These *capacity downgrades* must be performed even though they will cause the impacted links to be down for roughly a minute; (ii) the potential capacity of other links, above their current capacity, depending on the SNR of the link. For instance, a link could be operating at 100 Gbps, but if it has an SNR of 10.2 dB, it has a potential capacity increase of 50 Gbps, as its capacity can be augmented to 150 Gbps.

In what follows, we present the results of an extensive testbed evaluation of RADWAN (§2.4), benchmarking the effect of optically changing links' capacities on the IP layer. We then simulate RADWAN and compare it with our implementation of the SWAN controller as described in [46]. We perform a data-driven evaluation of the behavior and performance of these two controllers in Section 2.5 and show the gains of capacity variable links on the overall network throughput.

2.4 Testbed Evaluation

In this section, we build a testbed consisting of 1,540 km of fiber and 16 optical amplifiers to evaluate the feasibility of deploying RADWAN in a moderate sized WAN. Our goal is to highlight the impact of modulation changes on realistic traffic flows. We also provide insights to both researchers and practitioners into the state-of-the-art hardware components required to realize a rate-adaptive wide area network.

2.4.1 Testbed Implementation Details

We build a moderate sized testbed which emulates a WAN interconnecting four data centers, as shown in Figure 2.7(a), to evaluate RADWAN. Each data center consists of a

router connected to its neighbors through hundreds of kilometers of optical fiber. To prevent signal deterioration, we connect Erbium Doped Fiber Amplifiers (EDFAs) at approximately every 65-120 kilometers of fiber length. For simplicity, Figure 2.7(b) represents the logical view of the WAN.

Note that we had access to only one Arista 7504 modular chassis; therefore, we used Virtual Routing and Forwarding (VRF) [25] to logically split the same physical switch into four routers (named A, B, C and D in Figure 2.7(b)). Each VRF has a separate routing table and routing protocol instances. By configuring relevant physical interfaces to be in separate VRFs and connecting the interfaces via optical components (fiber, amplifiers), we achieve a logical topology whereby traffic between ports on the switch is sent out on the wire. We verify bi-direction connectivity between each pair of nodes A, B, C and D. The Arista 7504 has integrated bandwidth variable transceivers manufactured by Acacia Inc. (the BVT module, AC 400, is described in detail in Section 2.6.1). These allow us to configure three modulation formats (QPSK, 8QAM and 16QAM) on the switch ports. The complete testbed, including optical and electrical equipment is shown in Figure 2.7(c).

We implement the part of the RADWAN controller responsible for configuring the switch using Arista's PyEAPI [3] framework. With this, we can programmatically configure the modulation formats of different ports, program routes and query status of our commands.

To generate line rate traffic flows in the topology, we use a Spirent traffic generator [77]. With the help of the Spirent device, we program 400 Gbps of TCP traffic flows to test the dynamic capacity links of the testbed.

2.4.2 Benchmarking the WAN testbed

Reacting to SNR degradation. Optical signals in fiber can become attenuated because of ill-functioning amplifiers, disturbances caused during maintenance windows or even ambient temperature conditions. RADWAN reacts to signal attenuation by switching to a lower order modulation format that can be supported by the degraded SNR. In the laboratory setting, we use a Variable Optical Attenuator (VOA) device to add configurable amounts of noise (measured in dB) so that we can demonstrate signal attenuation. We connect the VOA between routers A and B in the test topology. On the underlying switch, this connection is implemented by connecting *Ethernet*$4/1/1$ to *Ethernet*$3/1/1$ with 410km of optical fiber. The Ethernet ports are in separate VRFs (not directly connected), so we set up static routing such that traffic sent from one to the other is sent over the fiber connection. Every five seconds, we increase the noise from the VOA by 1 dBm.

We measure the SNR of the signal on each end of the connection and observe that the SNR of the received signal on *Ethernet*$3/1/1$ steadily deteriorates as the level of noise increases (Figure 2.8). Once the added noise reaches 16 dBm, the transceiver can no longer recover from the increased errors,[2] and the port goes down. At this point, the controller reduces the modulation format of the port from 16QAM to 8QAM. The modulation change takes approximately 70 seconds to complete. We then resume incrementing the noise level using the VOA. When the noise level reaches 18 dBm, the transceiver can no longer recover from the errors to support 8QAM format, and the port goes down again. Our controller reacts by reducing the modulation format yet again, this time from 8QAM

[2]Acacia BVTs have 15% soft decision FEC enabled by default.

to QPSK. After roughly 70 seconds of down time, the ports come back up with QPSK modulation format. The addition of noise of 23dBm or more renders the link unusable, even in the lowest supported modulation format. At this point, the link has failed, and the failure is irrecoverable with the current set of hardware.

Modulation Change Latency. In the above benchmarking experiment, we changed the modulation format of a link in the testbed in response to SNR degradation. We observe that each change in modulation format changes the status of the ports involved to *down*, making them unavailable for sending and receiving traffic. In Figure 2.8, we observe that modulation change operations take approximately 70 seconds. This aspect of the latency of modulation change guides the design of the RADWAN controller (Section 2.3).

2.4.3 Evaluating Modulation Change

In this section, we demonstrate the capability of RADWAN to react to SNR degradation by reducing the modulation format of ports, allowing links with reduced signal quality to function at lower rates. We provide an end-to-end evaluation of RADWAN as it attempts to meet changing demand matrices by upgrading the capacities of links in the WAN. Additionally, we show that RADWAN migrates flows from a link undergoing capacity up-/downgrade (due to improved/poor SNR) to alternate paths until the modulation change is complete.

In each of the following experiments, we show the transmission rate (Tx Rate) of the traffic we attempt to send between nodes in the topology. An overwhelmed node responds to high traffic volume by dropping a portion of the flows. We capture the net traffic received by the sink node of a flow as the receive rate (Rx rate). In the ideal case,

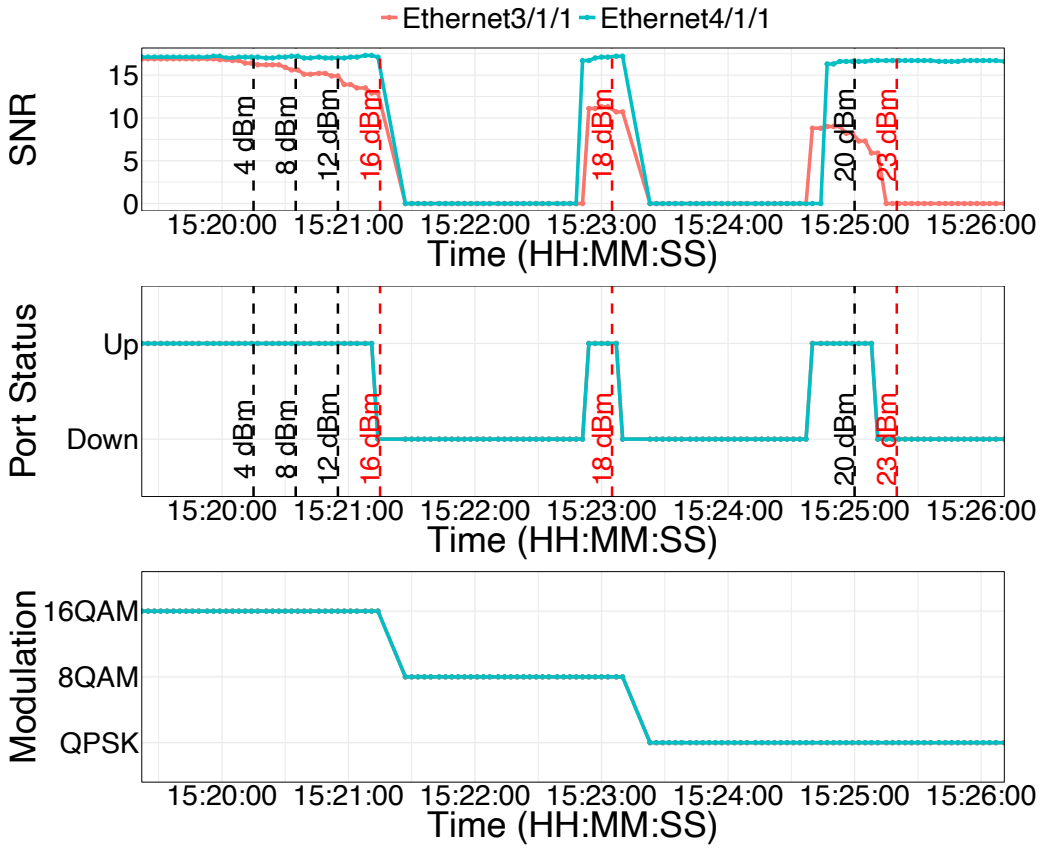
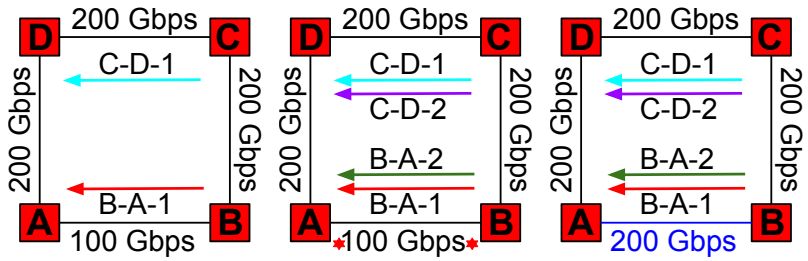


Figure 2.8: Impact of attenuation on link SNR, port status and modulation format as the amount of signal attenuation increases (shown by dotted vertical lines).

the Tx and Rx rates should match, implying that all the traffic sent by the source is reaching the sink node.

Link capacity upgrade. Figure 2.10() shows the starting state of a network with two flows of 100 Gbps, one from Node B to Node A (flow $B-A-1$) and the other from Node C to Node D (flow $C-D-1$). With the introduction of two additional 100 Gbps flows ($B-A-2$) and ($C-D-2$), as shown in Figure 2.10(a), the network becomes congested, because link $A-B$ can only carry 100 Gbps of traffic. As seen in the Rx rate in Fig-

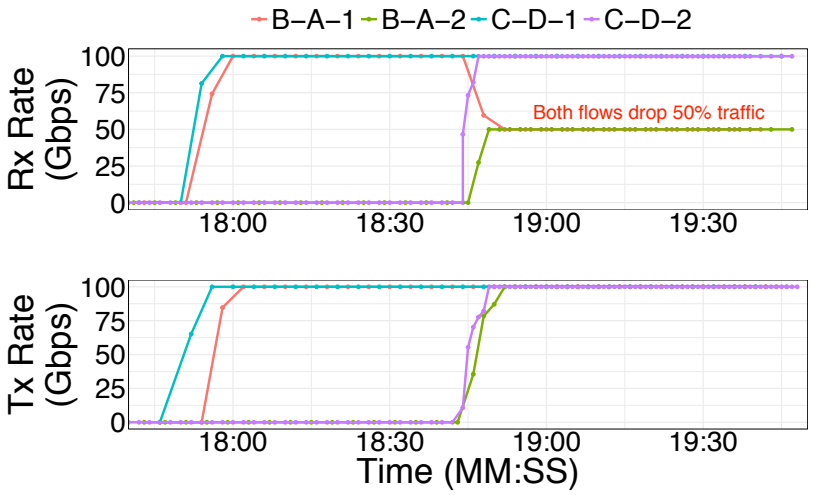


(a) (b) (c)

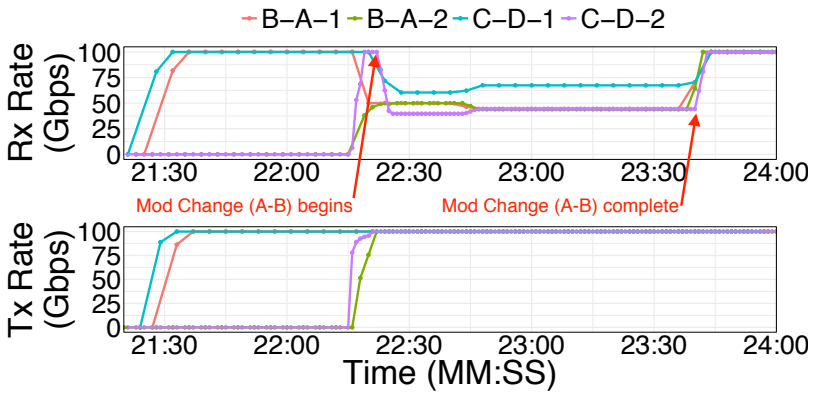
Figure 2.9: () shows the network and link capacities. At the start, all links except link $A-B$ are in 16QAM modulation format, capable of carrying 200 Gbps. $A-B$ being in QPSK format can carry 100 Gbps. In the beginning, there are two flows in the network, each 100 Gbps from $B \rightarrow A$ and $C \rightarrow D$. With an additional demand of 100 Gbps ($B-A-2$) and ($C-D-2$) described in (a), the link $A-B$ gets congested, leading to 50% traffic drops in flows $B-A-1$, $B-A-2$ in the absence of RADWAN, as seen in the Rx Rate in (a). However, in RADWAN deployment, the controller reacts to the increased demand by increasing the capacity of $A-B$ link to 200 Gbps (seen in (b) by changing the modulation format to 16QAM.

ure 2.10(a), both $B-A-1$ and $B-A-2$ share the $A-B$ link fairly and drop 50% of their traffic. However, RADWAN can salvage this congestion by increasing the capacity of the $A-B$ link (as seen in Figure 2.10(b)). To do this, the RADWAN controller reacts to the increased demand by changing the modulation of the $A-B$ link, causing it to be down for roughly one minute. This temporarily congests the $C-D$ link (the Rx rate of all flows drops in Figure 2.10(b)), because the $B-A$ flows are rerouted. However, once the modulation change is complete, all flows can be transmitted successfully with no packet drops. We note that without augmenting the capacity of link $A-B$, the network could not satisfy 400 Gbps of demand but dynamic capacity links with RADWAN enable us to meet the increased demand.

Link capacity downgrade. Figure 2.11(a) shows the starting state of our testbed when the network is carrying three flows of 100 Gbps, two from Node C to D ($C-D-1$,



(a) Network without BVTs.



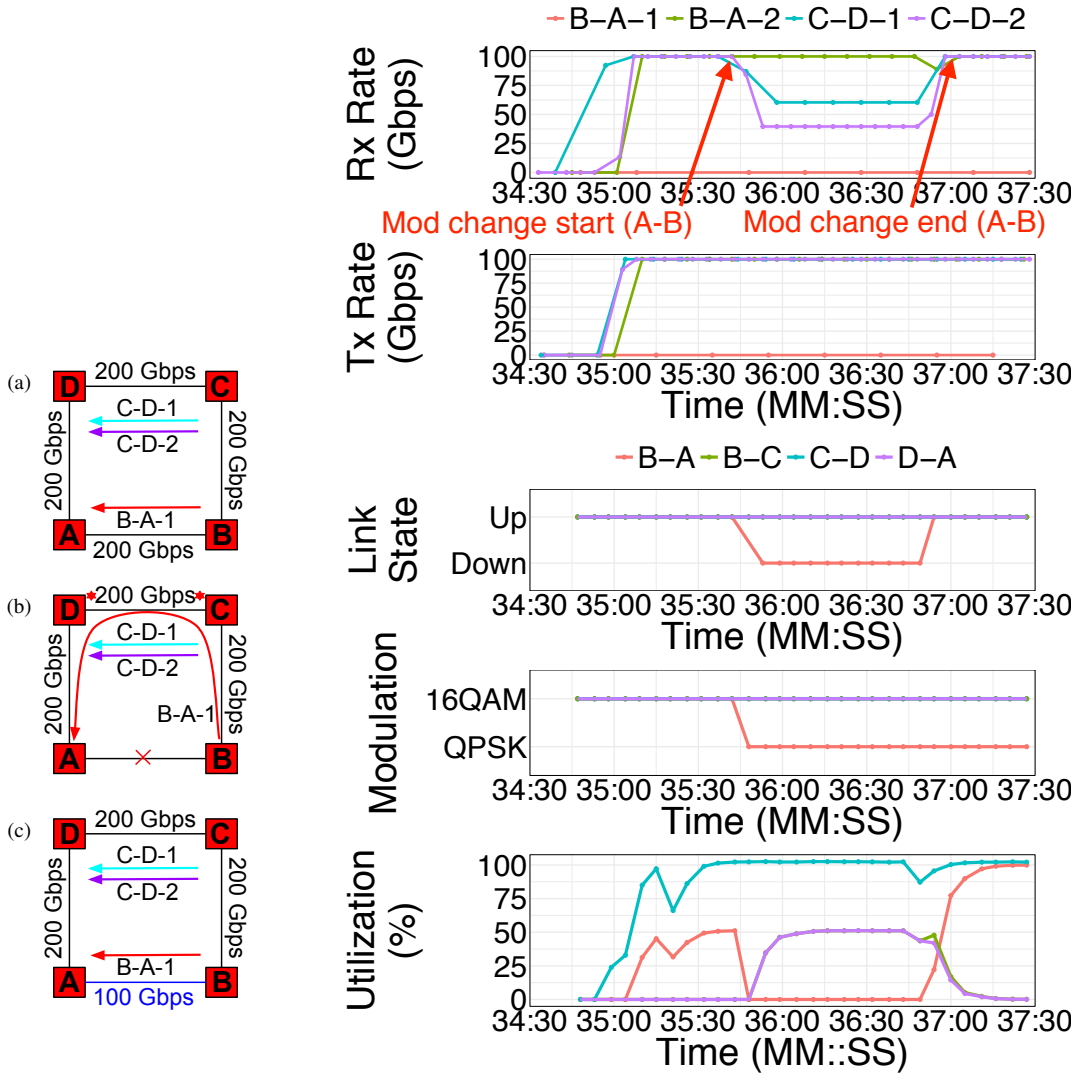
(b) RADWAN with BVTs.

Figure 2.10: The change in modulation format to 16-QAM causes temporary disruption due to rerouting of $B-A$ flows along the $C-D$ link, once the modulation change is complete, the network can carry the flows of 400Gbps without any drops, as seen in the Rx Rate of (b).

$C-D-1$) and one from Node B to A ($B-A-2$). All links in the network can carry 200 Gbps of traffic. Observe that the Rx rate in Figure 2.11(d) matches the Tx rate, implying there is no packet loss. Now, we attenuate the signal between Node A and B using a VOA device, such that the switch ports can no longer sustain transmission at 200 Gbps. Therefore, the link goes down (Figure 2.11(b)), causing ($B-A-2$) to be routed over the longer path $B\rightarrow C\rightarrow D\rightarrow A$ which is configured as the backup route. This transition of the $B-A-2$ flow along the longer path is visible in the utilization of links in the network (Figure 2.11(d)). Links $B-C$ and $D-A$ are now carrying 100 Gbps of the $B-A-2$ flow (and, thus, are 50% utilized). Note that this leads to congestion on link $C-D$ which can only carry 200 Gbps of traffic; accordingly, it drops 100 Gbps of traffic from the $C-D$ flows. The RADWAN controller can mitigate this congestion by reducing the modulation format of the $A-B$ link to QPSK from 16QAM. It takes roughly one minute for the modulation change to take effect, as observed in the *down* status of link $A-B$ in Figure 2.11(d). Once the modulation change is complete, link $A-B$ is back up and carries the $B-A-2$ flow without any congestion in the network (Tx/Rx rates match again). The new network state is shown in Figure 2.11(c). Therefore, our experiments show that RADWAN can react to traffic demands and signal quality by adapting the capacity of links in the WAN.

2.5 Large Scale Evaluation

In Section 2.1, we used three years of SNR measurements to demonstrate that an overall *capacity gain* of 67% is possible by augmenting the capacity of links from 100 Gbps to 125, 150, 175, or 200 Gbps, depending on their average SNR. This is the upper bound of the *throughput gain* achievable with RADWAN. The actual network throughput depends



(d) Reaction of RADWAN to poor SNR on $B-A$

Figure 2.11: (a) describes the starting network state and link capacities. At the start, all links are in 16QAM modulation format, capable of carrying 200 Gbps. There are three flows in the network, each 100 Gbps, one from $B \rightarrow A$ and two from $C \rightarrow D$. Due to signal attenuation, the link $A-B$ fails as seen in (b), causing the $B-A-2$ flow to be routed over the longer path $B \rightarrow C \rightarrow D \rightarrow A$. Observe that the utilization of links $B-C$ and $D-A$ increases in (d). This causes link $C-D$ to become congested and it drops parts of the $C-D-1$ and $C-D-2$ flows (Tx rate falls below the Rx rate in (d)). RADWAN reacts to this situation by reducing the modulation format of link $A-B$ to QPSK as is allowed by the lowered SNR of the link (see (c)). Once the modulation change is complete, all flows are routed along direct paths without any packet loss, as confirmed by the Tx/Rx rates and link utlizations in (d).

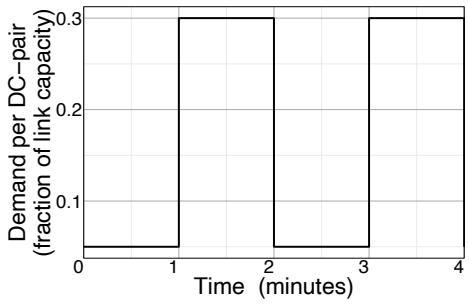


Figure 2.12: Simulated traffic demand pattern between each pair of nodes in the network similar to prior work [46].

not only on the network state (topology, link capacities, tunnels *etc.*) but also on the traffic demand and acceptable churn (defined in §2.3). In this section, we simulate the operation of RADWAN in a large backbone network with periodically varying traffic demands to compute the network throughput achieved. We compare the throughput and availability of the network under RADWAN and a state-of-the-art SWAN controller.

Both controllers are aware of the underlying signal quality of links. But unlike SWAN, RADWAN uses the SNR to update link capacities, choosing amongst discrete choices of 50, 100, 125, 150, 175 and 200 Gbps. As outlined in the previous section, RADWAN only upgrades the capacity of a link to meet increased traffic demand that cannot be met otherwise. Capacity downgrades are done to prevent link failures such that the lower quality link can continue to function at a reduced rate.

2.5.1 Simulation Setup

We consider the network topology of a large commercial WAN and gather SNR measurements from the optical fiber connecting the nodes in the topology for four randomly chosen days in 2016 and 2017. Both RADWAN and SWAN compute flow allocations

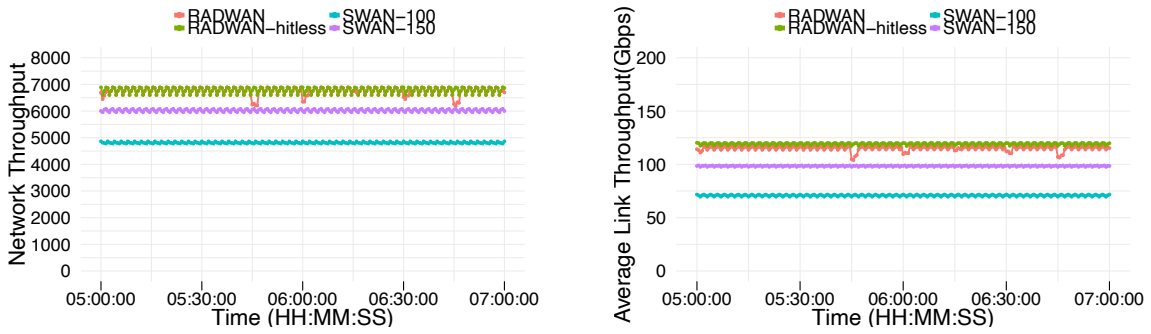


Figure 2.13: (a) Optimal network flow achieved by different traffic engineering schemes. For better visibility, (a) zooms into two hours of the simulation period. RADWAN achieves 40% higher network throughput than the state-of-the-art mechanism, SWAN-100 (RADWAN and RADWAN-hitless are overlapping curves on top of the graph). We also compare SWAN's performance with fixed capacity links operating with a static 150 Gbps modulation format. While SWAN-150 provides an improvement over SWAN-100, RADWAN achieves 12% higher throughput than SWAN-150. (b) Average per link throughput. We observe RADWAN achieves 68% higher per link throughput than SWAN-100.

along various network paths to meet an elastic demand between each pair of nodes in the network.

Since our WAN currently operates links at 100 Gbps, we consider the performance of SWAN in a fixed capacity network where each link operates at 100 Gbps if the SNR is above the threshold of 100 Gbps modulation; otherwise, the link is down. We refer to this scheme as SWAN-100 in the analysis. However, operators can be more aggressive by operating links at a fixed but higher capacity of 150 Gbps. We refer to SWAN operating in such a network as SWAN-150. SWAN-150 is used to compare the benefit of using rate adaptive schemes like RADWAN over a network with higher but fixed link capacities. While current hardware limitations prevent hitless capacity changes, we simulate the performance of RADWAN under both hitless (RADWAN-HITLESS) and non-hitless (RADWAN) link capacity change behavior.

The traffic demand between each node pair varies periodically every two minutes (demand pattern shown in Figure 2.13(a)). Our choice of network demands is similar to previous work [46], since rapid changes in demand matrices stress test the TE controllers. We also offset the traffic demand between each pair of nodes by using a randomized value to ensure that at any given point in time, there is sufficient variety of demands in the network.

Simulation Parameters. Unless otherwise stated, the control loop of both controllers is executed every 30 seconds as stated in [46]. In addition, we assume the demand between each pair of nodes can be split across $k = 2$ shortest paths between the nodes. For RADWAN, we set the churn trade-off parameter ϵ (defined in §2.3) to a conservative value of 0.001. We perform several runs of this experiment, with each run lasting for one day. We find that across four randomly chosen days, our results are similar. Hence, for the sake of brevity, the figures show results from one experimental run.

2.5.2 Evaluation Metrics

We focus on the following three key aspects of cost-efficient network design to evaluate RADWAN.

Network Throughput. First, we compute the optimal network flow that RADWAN can achieve in each run of the controller and compare it with the optimal flow that SWAN achieves for the same network conditions. This provides the network throughput enabled by both controllers for each run of their control loops for the duration of a day. Figure 2.13(b) shows the network flow for both RADWAN and SWAN for two hours of a day (zooming into two consecutive hours, picked randomly for the sake of better visibility in the figure). We observe that RADWAN manages to push 40% more traffic than SWAN-

100 in the same network. The same observation holds consistently with other hours and days we simulated.

Link Throughput. Next, we compare RADWAN and SWAN's per link throughput. For each run of the TE control loop, we compute the total traffic carried by each link and average it over all links in the network. Figure 2.13(c) shows the distribution of average link throughput over time (zoomed over two hours for better visibility). We find that, on average, RADWAN increases the utilization of network links by 68% compared to SWAN-100, getting more utility from each link in the network.

Link Availability. We compute the downtime of links in the WAN as the fraction of total simulation time for which a link is unavailable to carry traffic. Since the WAN we analyze is production grade, it was highly available during the 4 randomly chosen days in this simulation. Therefore, even under the existing SWAN-100 scheme the average link downtime is very small. However, we find that RADWAN reduces the average link downtime by a factor of 18 when compared to SWAN-100 operating in the same network. This is because RADWAN adapts links to lower capacities, when possible, instead of failing them when the signal quality degrades. Even though RADWAN's capacity reconfigurations are not hitless, we note that the link availability under RADWAN does not suffer significantly as very few links undergo rapid changes in capacity. This is confirmed by Figure 2.14 which shows the distribution of the number of capacity reconfigurations observed during the simulation period.

As expected, in the absence of catastrophic optical events (SNR < 3) during the simulation period, RADWAN-HITLESS allows links to be available all the time by instantly

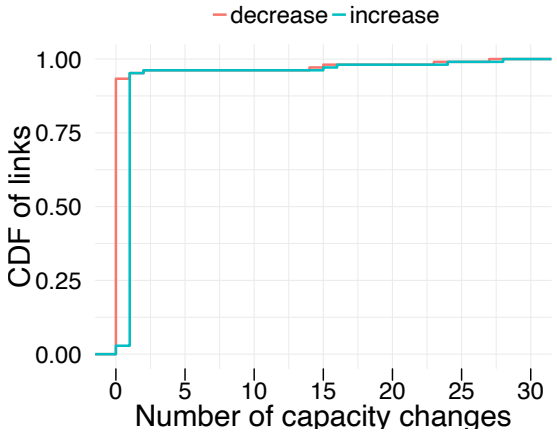


Figure 2.14: Distribution of the number of capacity reconfigurations occurring per link in the network. We note that only 6% of the links change their capacities more than once in the simulation period.

adapting the link capacity to the lower or higher SNR. We also find that SWAN-150 achieves the same availability as SWAN-100 in our simulation.

2.6 Discussion

In this section, we consider future directions of rate adaptive networks and suggest means of achieving hitless capacity change. We then discuss the impact of underlying fiber length on dynamic capacity links and the cost of operating them.

2.6.1 Hitless Capacity Change

BVTs and dependency graphs. Dependency graphs [65, 54] are a seminal technique used for consistent network updates [30]. To perform consistent updates, an old and a new network state is specified such that a routing change is performed only when safe to do so. However, to change the capacity of a link e, carrying flow f before and after the capacity

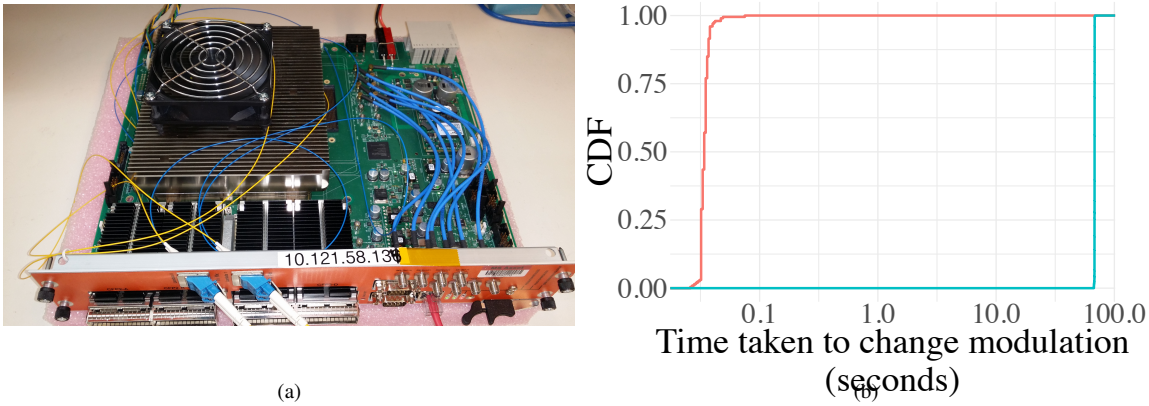
reconfiguration, dependency graphs perform poorly since no alternative path is specified for f.

RADWAN manages link flaps by computing an intermediate routing state for flows during reconfiguration. As such, RADWAN specifies a two-step dependency graph: in order for a scheduled link flap to be activated, the affected traffic is rerouted beforehand.[3] Because of the benevolent nature of SNR in our dataset, coupled with the churn minimization of Algorithm 1, RADWAN jointly activates all link flaps. In more volatile SNR scenarios, RADWAN can be set to activate link flaps over multiple dependent iterations. We conjecture that such intermediate consistency methods can eventually be phased out once hitless capacity changes become production ready, as discussed in the next section.

Towards hitless capacity change. BVTs are not yet optimized to handle the latency of a modulation change. State-of-the-art BVTs can only change the link modulation after bringing the module to a lower power state. This translates to a link flap for higher layer protocols. The duration of such link failures is a challenge in the deployment of dynamic capacity links in production networks. To quantify this, we obtain an evaluation board of the Acacia AC400 bandwidth variable transceiver [1]. This is the same module which is integrated in the switch linecard used as part of our testbed in Section 2.4. Since the evaluation board exposes an API to program the transceiver, we use it to understand the modulation change procedure. We change the link's modulation 200 times from QPSK to 16QAM and analyze the time taken.

---

[3] OWAN [52] also deals with consistent cross-layer reconfiguration in WANs, but it is designed for Reconfigurable Optical Add-Drop Multiplexers, where wavelengths are exclusively *either* activated or deactivated: link flaps due to BVTs are not considered.



(a) (b)

Figure 2.15: (a) AC400 BVT evaluation board to analyze modulation change latency. (b) CDF of the time taken to change modulation (capacity) of a fiber link using the BVT. Link capacity changes take 68 seconds, on average, but we demonstrate ways to change the modulation efficiently, so that it takes only 35 milliseconds.

Figure 2.15a shows the AC400 bandwidth variable transceiver module. We observe that the average downtime of the link undergoing capacity change is 68 seconds, similar to the observation made in Section 2.4. We investigate the cause of latency in capacity reconfiguration and find that the majority of this time is associated with turning the laser back on after reprogramming the transceiver module. We plot the distribution of time taken to change modulation without turning off the laser (Figure 2.15b) and find that it only takes approximately 35 ms on average. This suggests an opportunity to strive towards hitless capacity changes in the fiber.

2.6.2 Cost and Distance

One of the key benefits of deploying bandwidth variable links is their cost savings. While the exact cost of individual transceivers is highly dependent on bulk discounts of-

fered by device manufacturers, conversations with industrial partners suggest that the cost of BVTs is on par with the cost of 100 Gbps static transceivers. Due to comparable costs of the two transceivers, operators are increasingly adopting BVTs even though their modulation format is programmed only a handful of times.

RADWAN allows operators to take advantage of BVTs by enabling higher data rates and consequently reducing the dollars per gigabit ($/Gb) value of traffic in the network. Using the distribution of potential link capacities (Figure 2.1) enabled by Acacia BVTs and the $/Gb cost of sending traffic, we estimate that RADWAN provides an overall cost saving of at least 32% over the state-of-the-art.

A caveat of using higher order modulations is that they limit the distance light can travel in fiber. This is because higher number of symbols (as in 8 QAM and 16QAM) in the modulation format reduces the minimum distance between adjacent symbols, making the transmission more prone to distortion as the signal traverses longer distances [29].

As mentioned in §**??**, QPSK modulation format supports data rates of 100 Gbps for distances upto 3,000 km, 8QAM allows 150 Gbps for distances up to 2,100 km, and 16QAM allows 200 Gbps for distances up to 800 km. We analyzed the fiber distances in our WAN and found that the majority of our fiber paths are less than 800 km (thus capable of supporting 16QAM) and only a small percentage of paths are longer than 2,100 km. While our current proposal did not take fiber length into account, we believe it can be extended to incorporate distance as a constraint.

2.7 Related work

Our work builds on several lines of related research as categorized below.

Optical and IP layer orchestration. Singh *et al.* [76] recently analyzed the SNR of links in a large North American backbone over a period of 2.5 years and proposed adapting link capacities to the SNR optical channels. We extend their study period to three years, and, at the same time, broaden their initial measurement and testbed quantifications. We also propose a centralized TE controller system RADWAN and evaluate the interaction between dynamic capacity links and IP layer flows with simulations at scale and in a realistic testbed. The study by Jin *et al.* [52] on cross-layer optimization between IP and optical layers wavelengths is similar in spirit to our motivation of bridging the gap between optical and IP layers. In their work, Jin *et al.* show the reconfiguration of wavelengths provides latency gains for deadline-driven bulk transfers, also providing a competitive analysis of scheduling single-hop transfers in [51]. But their work keeps the capacity of each wavelength static. In contrast, our work focuses on the reconfiguration of the *capacity* of wavelengths, without the migration of wavelengths across links. In addition, we provide measurements from an operational backbone and argue for changing link capacities with a focus on throughput and reliability. An interesting future direction would be to study the throughput and latency gains of a combination of the two proposals: a fully programmable WAN topology where both capacities and placement of wavelengths on fiber is informed by the centralized TE.

WAN measurements. Govindan *et al.* [41] study 100 failure events across two WANs and data center networks, offering insights into the challenges of maintaining high levels of availability for content providers. Although they do not isolate optical layer failures, they report on root causes of failures, including optical transmitters. We complement their work by focusing on optical layer failures. Ghobadi *et al.* study Q-factor data from

Microsoft's optical backbone [34, 35] and provide insights into the data. Our work complements their analysis on several fronts. First, we make a deeper dive into the impact of temporal changes of SNR on link capacities in terms of capacity gain, availability gains, and realistic throughput gains. Second, we propose and build the system infrastructure required to achieve capacity variable links and benchmark the throughput gains using realistic IP level data. Third, we build a comprehensive testbed and evaluate the impact of capacity reconfiguration, as well as amplifiers, on the path. Our work closes the loop for enabling capacity variable links. Similarly, Filer *et al.* [26] studied the deployed optical infrastructure of Microsoft's backbone; they discuss the benefits of optical elasticity, express a long-term goal of unifying the optical control plane with routers under a single Software Defined Network controller and recognize YANG [10] and SNMP as potential starting points for a standard data model and control interface between the optical layer and the WAN traffic controller. In this work, we explore how programmability in the optical layer yield throughput gains, and we present a cross-layer WAN traffic controller for dynamic capacity links. Marian *et al.* [66] focused on IP and TCP layer measurements, such as packet loss and packet inter-arrival times, on fiber optics spans. In contrast, we capture failures in the optical layer using failure tickets.

Hardware feasibility studies. Yoshida *et al.* [88, 89] studied the use of 12.5 GHz spectrum slices to allocate bandwidth variable connections to improve the spectrum usage. Although their works did not consider real-time adjustment of the capacity, it provided the foundation for the feasibility of building the necessary hardware with variable bandwidth capabilities–the enabler of our work. We use real-world measurements and build a system that fills the gap between optical and IP layers. Fischer *et al.* [28] and Teipen *et al.* [81]

efforts to commercialize higher-speed optical transmission have demonstrated the need for advanced modulation formats, several of which require similar transceiver hardware architecture. Their work showed that adaptive transceivers can be built to support a number of possible operational configurations, but they did not employ a real-time reconfiguration mechanism. In contrast, we discuss the advantages of reconfigurable capacities in real-time based on live SNR measurements.

2.8 Conclusion

In this work, we quantify the throughput and reliability benefits of rate adaptive wide area networks. Our analysis of the SNR of over 8,000 links in an optical backbone for a period of three years shows that the capacity of 64% of the IP links can be increased by 75 Gbps, yielding an overall throughput gain of 134 Tbps. Furthermore, 25% of link failures can be avoided by reducing the transmission rate to 50 Gbps from 100 Gbps. To leverage these benefits, we present RADWAN, a traffic engineering system that dynamically adapts link rates to enhance network throughput and availability. We evaluate RADWAN in a testbed with 1,540 km optical fiber and also simulate throughput and availability gains at scale. By simulating the traffic demand and failures of four random days, we show RADWAN can achieve 40% higher throughput than SWAN. We also address the challenge of the hardware delay in modifying a link's capacity. We analyze the cause of this delay in current optical transceivers and propose a potential solution to reduce this delay from over a minute to a few milliseconds.

CHAPTER 3

COST-EFFECTIVE CAPACITY PROVISIONING IN WIDE-AREA NETWORKS

Traffic demands on cloud wide area networks (WANs) are growing rapidly, driven by new workloads like real-time video and cloud gaming. Cloud providers respond to increase in traffic demands by provisioning additional WAN capacity. However, long-haul network capacity is expensive – the median annual cost of 100 Gbps of long-haul capacity is over $100,000 in N. America as per TeleGeography (Figure 3.1) [80]. Fiber, routers and equipment in the optical line system (OLS) [48] are the key contributors to the cost of capacity in the cloud WAN (Figure 3.2). Large cloud providers have existing fiber deployments, acquired through purchase or long-term leases. Thus, the marginal cost of provisioning capacity in the cloud is dominated by the additional hardware resources used in the process: *router and optical ports*.

Cloud providers operate *point-to-point* inter-regional networks – where optical signals are converted to electrical signals and back at every geographical region [27]. Thus, inter-regional traffic undergoes optical-to-electrical-to-optical (OEO) conversion at all intermediate regions on the path towards its destination. For example, in Figure 3.3, wavelength $\lambda 1$ originates at US-East, terminates at US-West but undergoes an OEO conversion at US-central. Once the signals are converted from the optical to electrical domain, centralized traffic engineering systems take control of routing them [46, 49].

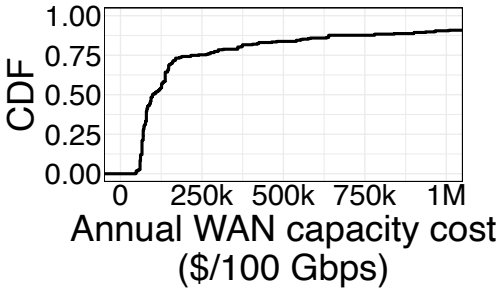


Figure 3.1: Capacity cost.

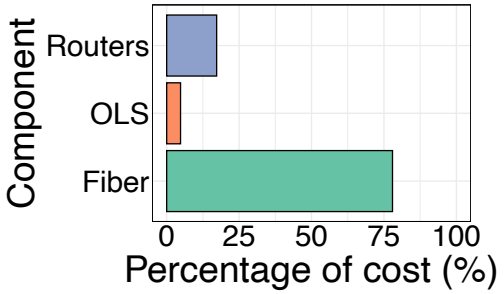


Figure 3.2: Cost breakdown.

This design provides control and flexibility at the network layer, keeping the optical layer uncomplicated and easy to manage. However, the conventional design does not keep in view the nature of traffic demands and the corresponding traffic flow imposed by them. We analyze inter-regional traffic patterns in the backbone of a large commercial cloud provider and find that 60% of traffic traversing through 30% of geographical regions in the WAN is *passing through* – neither originating nor terminating at the region. The pass-through or transit traffic undergoes wasteful OEO conversions at all intermediate regions in point-to-point networks, occupying scarce optical line- and router ports. These ports contribute a majority of the cost of provisioning capacity in cloud networks with existing fiber deployments (Figure 3.2).

In this work, we propose to minimize the hardware cost of provisioning long-haul capacity by removing wasteful OEO conversions in the WAN. We refer to the elimination of an OEO conversion at a region as *optically bypassing* the region *e.g.,* wavelength $\lambda 2$ optically bypasses US-Central in Figure 3.3. Our analysis shows that 60% of the transit traffic through a region is exchanged between only two neighbors of the region – highlighting the potential of reaping most cost savings with very *few* optical bypasses.

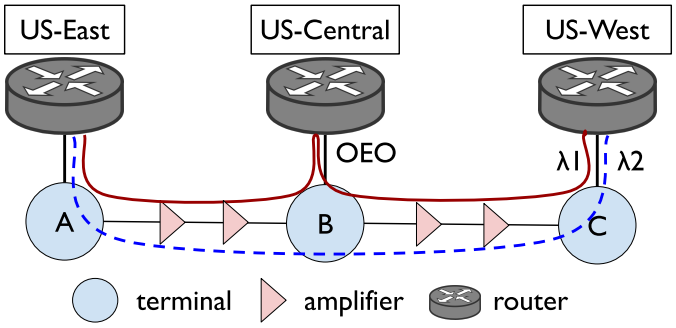


Figure 3.3: Optical terminals consist of wavelength selective switches (WSS), multiplexers/demultiplexers. They connect light channels to router ports. Transponders plugged into router ports convert the optical signals to electrical signals.

While optically bypassing regions in cloud backbone networks can offer significant reduction in capacity cost, it also introduces new operational challenges. First, wavelengths of light in point-to-point regional networks undergo regeneration at every hop, correcting errors caused by signal attenuation and dispersion during transmission on the fiber. By optically bypassing regions, signals are forced to travel longer fiber paths before regeneration *e.g.,* $\lambda 2$ in Figure 3.3 travels from US-East to US-West without regeneration whereas $\lambda 1$ undergoes regeneration at US-Central. Longer distances can necessitate lower data rates for error-free transmission *e.g.,* $\lambda 2$ faces more attenuation than $\lambda 1$ since it travels a longer fiber path without regeneration and can only sustain data rates lower than $\lambda 1$'s. Thus, optical bypass can lower the achievable capacity over a channel, in turn hampering the network's ability to meet traffic demands. Second, IP links resulting from signals that optically bypass regions are susceptible to failure if *any constituent fiber link* fails. This expansion of shared risk link groups (SRLGs) can reduce the network's resilience to physical link failures [55].

We tackle these challenges by analyzing the optical signal quality on fiber of a commercial cloud provider and find that the signal quality of 75% of optical channels on fiber is sufficiently high to sustain transmission over longer distances. The remaining optical channels must downgrade their data rates to traverse longer fiber paths without regeneration (§3.1). We leverage these empirical insights to make the following key contributions towards the design of a bypass-enabled cloud backbone with resilience to link failures:

Optimal optical bypass in cloud WANs. We develop Shoofly, an optical backbone design tool that formulates the problem of identifying optical bypass opportunities in a cloud network with the goal of *minimizing the hardware costs of long-haul capacity*. Shoofly-proposed backbone topology can reduce the hardware cost of long-haul capacity by 40% while continuing to meet up to 8X the present-day demands using *existing hardware and fiber deployments* (§3.2, §3.3).

Failure resilience with optical bypass. We extend Shoofly to incorporate the goals of failure resilience while identifying bypass opportunities in the WAN. We show that Shoofly can provision bypass-enabled topologies resilient to both stochastic [11] and deterministic [62] link failures by sacrificing at most 20% of hardware cost savings (§3.4).

Low logistical burden of enabling optical bypass. We quantify the logistical burden of provisioning the Shoofly proposed bypasses on cloud operators. We show that the bypass-enabled topology can keep up with demand growth over time, up to 8X the present-day demands. Implementing bypasses requires modifications to the physical connections between optical terminals and routers. We show that a small number of such changes are needed since 25% of optical bypasses achieve 80% of the cost savings (§4.5).

The design of optical networks has been well studied by service and content providers [17, 40, 6, 5]. In contrast, cloud WANs with centralized software-defined traffic engineering present a unique opportunity to rethink conventional backbone designs due to the *predictability* [84] of intra-WAN traffic in cloud networks.

3.1 Quantifying the opportunity

Cloud providers lease or purchase optical fiber across the world to provision their WAN. This fiber is connected to optical equipment, namely, *optical terminals* and *optical amplifiers* to transmit optical signals over hundreds of kilometers. Signals can originate, terminate or pass through an optical terminal. Signals that terminate undergo a conversion from the optical to electrical domain. The corresponding electrical signals are then demultiplexed onto ports of a router or switch. Similarly, electrical signals from router ports are converted to optical signals and multiplexed onto wavelengths of light that traverse the fiber. In contrast, signals can *pass through* an optical terminal without originating or terminating at it. These signals are said to *optically bypass* a router (*e.g.,* US-Central is bypassed by wavelength $\lambda 2$ in Figure 3.3).

3.1.1 Point-to-point regional backbones

The commercial cloud provider we study has provisioned their wide area backbone in a point-to-point regional topology. In this design, optical signals undergo regeneration at all regional hops. As per conventional wisdom, this design offers two main benefits:

Fine-grained control via Layer-3 traffic engineering. Optical signals are converted to the electrical domain at all regional hops in this design, allowing the Layer-3 traffic

engineering algorithms hop-by-hop autonomy to route traffic to suitable next-hops. These algorithms can leverage fine-grained, cross-layer telemetry from the network to make on-the-fly decisions that achieve network-wide goals, *e.g.,* minimum end-to-end latency and maximum throughput.

Flexibility to meet new demands. Since wavelengths undergo conversion to electrical signals at every region in a point-to-point WAN, traffic from one region can be IP routed to any other region in the network. This flexibility allows the network to meet new and emerging traffic demands between regions without requiring any changes to the optical backbone. In this design, the optical and IP topology of the network bear close resemblance. Each optical terminal maps to an IP router and fiber connections between neighboring regions underpin the IP links between the corresponding routers (Figure 3.4). Enabling optical bypass can hamper the ability of certain regions from being the origin or destination of traffic. In an extreme case, a region could be bypassed entirely by all optical wavelengths (*e.g.,* region B in Figure 3.4).

Despite the flexibility and control offered by point-to-point wide area backbones, we propose to rethink this design in the context of traffic flow patterns in the network. In this section, we demonstrate that the nature of inter-regional traffic flow enables a significant potential to save hardware costs of capacity provisioning in point-to-point backbones.

3.1.2 Wasteful OEO Conversions

We study the traffic flow between geographical regions of a large commercial cloud provider. The cloud provider has datacenters in approximately 100 geographical regions in the world, connected by a dedicated optical backbone. Demands between regions

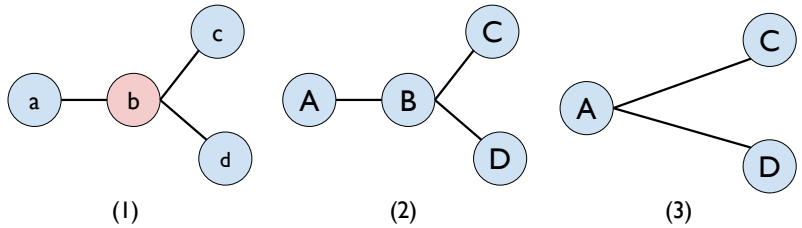
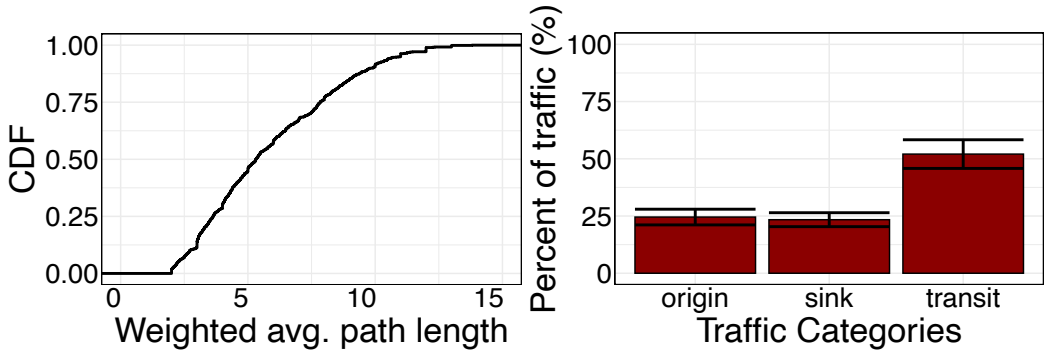


Figure 3.4: (1) shows the physical topology of a network with 4 optical terminals (a, b, c, d) and the fiber between them. (2) shows the IP layer topology of the same physical network in a point-to-point design – each terminal maps to a router (A, B, C, D). (3) shows the IP layer topology of the physical network in (1) where all signals optically bypass router B.

are routed through a centralized software-defined traffic engineering (TE) controller [46, 49]. At a high level, the TE controller solves a k-shortest path formulation of the multi-commodity flow problem [2]. Demands are denoted by their source and destination regions. For every demand pair, there exist pre-computed k paths across which flow is distributed. We measure hourly demands between regions in the WAN and the corresponding k paths over which they are routed from August 1, 2020 to December 31, 2020.

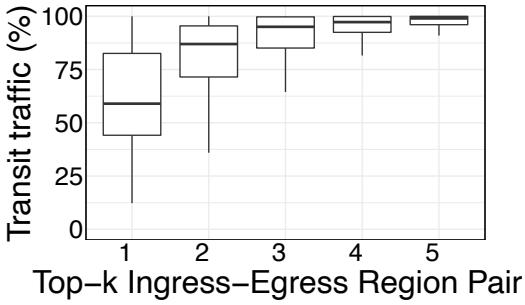
All traffic transits at least one geographical region. For every demand source and destination pair, we measure the fraction of traffic that traverses WAN paths of different lengths. Each hop on these paths is a geographical region in the WAN. We compute the average length of the WAN path for all demand pairs, weighted by the fraction of demand traffic carried by the paths. Nearly all demands are routed through indirect regional paths – with 75% demands encountering at least 3 intermediate regional hops (Figure 3.5a).

Majority of traffic routed at any region is *pass-through*. For each region, we measure the three categories of traffic it observes: *origin traffic* originates at the region, *sink traffic* terminates at the region and *transit traffic* passes through the region. We find that all regions observe a high volume of transit traffic. For 30% of geographical regions, the



(a) Avg. path length.

(b) Traffic types in regions.



(c) Transit traffic in regions.

Figure 3.5: 3.5a shows the average regional path length for demands in the WAN weighted by traffic volume. 3.5b shows the three categories of traffic observed by regions. 4.3a shows the transit traffic volume per $k = 1, \ldots, 5$ top ingress and egress neighbors of a region. Over 60% of transit traffic through all geographical regions is between one pair of neighbors.

transit traffic volume is over 60%. On average, over 50% of traffic observed at a region is transit traffic (Figure 3.5b). The high volume of transit traffic through regions contributes to wasteful OEO conversions that occupy routers ports and transceivers. The cost of long-haul capacity can be lowered if the traffic optically bypassed transit regions, staying in the optical domain for longer distances until reaching its destination.

Few regional pairs contribute most of transit traffic. Finally, we measure the neighboring regions that ingress and egress transit traffic through a region in the WAN. We

calculate the fraction of transit traffic through a region that is contributed by one ingress-egress neighbor pair. We find that over 50% of the transit traffic through any region is exchanged between two neighbors of the region (Figure 4.3a). The distribution of transit traffic through regions by ingress-egress regional pairs sorted on transit traffic volume, in Figure 4.3a shows that optically bypassing the region between few ingress-egress region pairs will achieve most of the cost savings of optical bypass in WANs.

While the flexibility of point-to-point backbones (§3.1.1) is an important feature, our analysis of inter-regional traffic flow in the cloud WAN over five months shows that the traffic flow between geographical regions of the cloud WAN is suitable for a bypass enabled design. Despite the seasonality of demands, predictable traffic engineering algorithms have imparted *stability* to traffic patterns – few regions are responsible for most of the transit traffic. We propose to leverage this stability to design a cost-efficient optical backbone.

3.2 Optical bypass with Shoofly

We propose to leverage insights from inter-regional traffic patterns in the WAN to enable optical bypass in the network. Our goal is to reduce the cost of long-haul capacity by preventing wasteful OEO conversions while using existing network hardware and software.

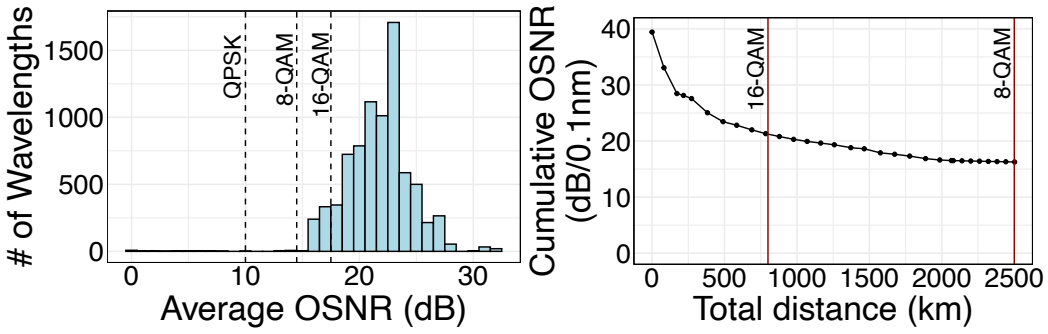
Cost of long-haul capacity. Large cloud providers either purchase fiber or acquire it through long-term leases that span decades, making fiber an infrequent capital investment. Cloud providers also purchase optical (*e.g.,* amplifiers, terminals, transceivers) and electrical equipment (*e.g.,* routers) to provision capacity between different regions of their

wide-area backbone. Over time, wavelengths are *lit* on the fiber to meet growing traffic demands. In steady state, the cloud provider has existing fiber deployments. Thus, provisioning additional capacity requires *lighting* a new wavelength or signal on the fiber that originates and terminates at the target source and destination regions. The marginal cost of provisioning the wavelength is determined by the router and optical line ports it uses. The combined cost of a router and optical port can be as high as tens of thousands of dollars. In point-to-point networks, each wavelength uses at least one router port and an optical port at every intermediate regional hop in addition to the source and destination regions.

How does bypass save cost? Optical bypass of a region by a wavelength of light saves two router ports and up to two line ports in an industry standard deployment of routers and optical terminals, allowing a significant reduction in the $/Gbps cost of long-haul capacity. However, the ability to bypass a region is conditional on physical constraints on the network topology and signal quality on fiber. We elaborate on these constraints in the next section.

3.2.1 Physical constraints on optical bypass

Optical bypass of geographical regions forces the signals to travel longer distances on fiber before they can be regenerated. Signals traveling longer distances undergo more attenuation leading to lower optical signal quality. Signal quality, measured through optical signal-to-noise ratio (OSNR), ultimately decides the data rate of the optical signal – higher the OSNR, higher the signal data rate. State-of-the-art optical transponders support three data rates per wavelength of light: 200 Gbps, 150 Gbps and 100 Gbps by modulating the signals in 16-QAM, 8-QAM and QPSK formats respectively. Thus, optical bypass



(a) Optical signal quality.

(b) OSNR vs. fiber distance.

Figure 3.6: 3.6a shows the histogram of OSNR of wavelengths in the cloud WAN. 3.6b shows the decline in OSNR as the transmission distance increases. (The graph is truncated at 2,500 km since the decline in OSNR is slow.)

by some wavelengths in the WAN can lower their achievable data rates due to increased transmission distance. In this section we explore the constraint posed by distance on potential optical bypasses in the cloud network. Table 3.1 shows the relationship between modulation formats, their OSNR thresholds and data rates.

Optical signal quality in the WAN. The OSNR of all wavelengths in the cloud WAN is higher than 15 dB (Figure 3.6a), highlighting that nearly all wavelengths in the cloud network currently support data rates of 150 Gbps (8-QAM) or higher. 75% of the wavelengths have an OSNR over 20 dB which is higher than the threshold OSNR for the highest data rate of 200 Gbps in the cloud network. Since their OSNR is over 3 dB above the threshold for 16-QAM and 5.5 dB above the threshold for 8-QAM formats, these wavelengths can travel longer distances without regeneration while still maintaining the same data rate. The amount of extra distance that a wavelength can travel without requiring a downgrade in modulation format depends on its current OSNR and modulation format. We discuss this in the next sub-section.

Modulation format	QPSK	8-QAM	16-QAM
Minimum OSNR	10dB	14.5 dB	17 dB
Data rate	100 Gbps	150 Gbps	200 Gbps
Optical reach	5,000 km	2,500 km	800 km

Table 3.1: OSNR thresholds, data rates and optical reach of modulation formats of signal on fiber.

Optical reach. Optical reach is the maximum distance a light signal can travel before it must be re-generated. If the signal is not re-generated within this distance, the OSNR of the signal is too low to merit error-free decoding at the destination. While state-of-the-art transceivers and routers have forward error correction (FEC) abilities to correct transmission errors, if the OSNR is lower than the FEC threshold, error-free transmission is not possible. The reach of a signal depends on characteristics of the physical network, including the launch power of the signal, noise on fiber, gain of amplifiers and optical span lengths. We gather these parameters from the cloud network and simulate the relationship between the OSNR of signals and the distance traversed by them. This OSNR estimation is approximate and is used for network planning. Figure 3.6b shows the decline in the simulated OSNR as the distances traversed by the signal increases. We observe that after 1,000 km, the OSNR of signals nears the 16-QAM threshold of 17 dB. If the signal is transmitted over this distance threshold, it must be modulated in a lower order format like 8-QAM. Similarly, after traversing more distance, the signal OSNR drops below the 8-QAM threshold.

We subtract a margin of over 200 km from the optical reach estimates derived from Figure 3.6b to make the reach estimates conservative. Table 3.1 summarizes the optical reach of a signal as a function of its modulation format – lower order modulation (*e.g.,* QPSK) formats can travel longer distances without re-generation compared to high order

modulation formats (*e.g.,* 8-QAM, 16-QAM). Thus, higher order modulation formats enable higher data rates but have lower optical reach. Due to limited optical reach, each signal can bypass a fixed number of regions before a re-generation becomes essential. The number of regions that can be bypassed depends not only on the traffic patterns between the regions but also on the modulation format for the signals. If bypassing a region pushes the transmission of a signal over the optical reach of its modulation, the operator must lower the modulation format and consequently the data rate of the signal. Given the set of demands in the network, lowering the modulation can reduce the network's ability to meet demands. Therefore, selection of regions and wavelengths for optical bypass must navigate the balance between cost saving and the ability of the network to meet demands.

3.2.2 Bypass as network shortcuts

Our empirical analysis of traffic patterns (§3.1) shows the potential for optically bypassing regions to save on the hardware cost of capacity in the WAN. However, identifying the set of wavelengths and regions that can be bypassed is hard. The space of potential optical bypasses is constrained by physical factors (*e.g.,* signal quality and optical reach), traffic demands and network tunnels [20] over which they are routed. To effectively enumerate and search the space of potential optical bypasses, we introduce the graph abstraction of *network shortcuts* to represent an optical bypass.

The bypass of one or more regions by a wavelength on fiber introduces a new edge in the corresponding IP network. We refer to this bypass-induced edge as a *network shortcut*. In Figure 3.7, the bypass of region A by a wavelength between regions E and B introduces the shortcut EB in the IP network. At the physical layer, the only change is the optical bypass of region A implemented by changing physical connections between the terminal

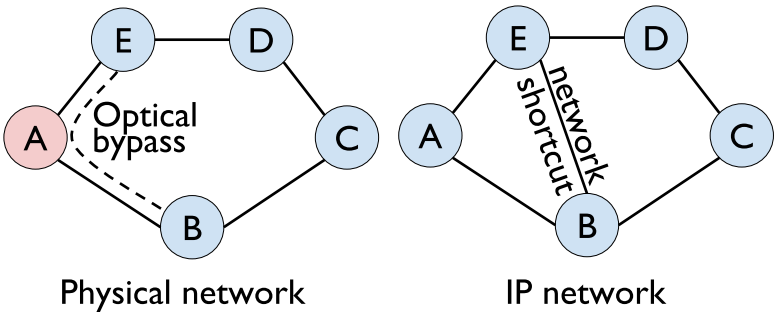


Figure 3.7: Figure shows optical bypass at the physical layer translates to adding a *shortcut* edge in the IP network, *e.g.,* one wavelength from node E bypasses node A and terminates at node B. This change adds a shortcut edge between node A and B.

and router at region A. But, higher layers of the networking stack observe a direct connection between nodes E and B as a result of this change. We define one instance of bypass by the corresponding shortcut and its underlying fiber path , *e.g.,* the bypass in Figure 3.7 is defined by the shortcut EB and $E \rightarrow A \rightarrow B$.

Feasible network shortcuts. Using the network shortcut abstraction, we enumerate all potential shortcuts in the cloud backbone. To do this, we compute the shortest regional path between every pair of regions in the network. After 3 intermediate regional hops between the source and destination regions, the OSNR is too low and the signal must be regenerated regardless of the other physical constraints. Thus, feasible network shortcuts can have up to 5 hops including the source and destination region. This enumeration yields a list of all potential shortcuts in the network.

Wavelengths as minimum unit of capacity. After enumerating all feasible bypasses in the network as shortcuts, the remaining unknown is the capacity of each shortcut. Our goal is to find the capacity of enumerated shortcuts such that physical constraints and traffic de-

mand constraints are met while the bypass-enabled cost savings are maximized. Similar to the capacity of existing IP links in the network, the capacity of a shortcut is the product of the number of wavelengths that constitute the shortcut and the data rates of their modulation formats. These wavelengths originate at the start of the shortcut, terminate at the end and bypass all intermediate nodes. The modulation format of a shortcut is determined by the OSNR and shortcut length. While the modulation formats of wavelengths in the original cloud topology are known, we determine the modulation formats of wavelengths on shortcuts using the shortcut length (Table 3.1). If the shortcut length is higher than the optical reach of the wavelengths' modulation format, the shortcut can sustain transmission at a lower modulation format.

3.2.3 Optimal optical bypass

We propose Shoofly, a tool that formalizes the task of minimizing the hardware cost of long-haul capacity by optically bypassing regions in the WAN. Shoofly leverages the network shortcut abstraction (§3.2.2) and enumerates all feasible shortcuts, $s \in S$. Since the shortcuts are pre-computed, the per-wavelength data rates (u_s) of shortcuts are also an input to Shoofly. The operator can prune the set of feasible shortcuts to impose policy decisions, *e.g.,* only allow the bypass of one region at a time or only bypass regions in Europe.

Decision Variables. Shoofly allocates wavelengths to each network shortcut, s. The wavelengths on s are bounded by the number of wavelengths on each edge constituting the shortcut in the original network. The decision of allocating wavelengths w_s to shortcut s implies that w_s light signals between the start and end regions of the shortcut do not regenerate at intermediate regional hops by bypassing them. The capacity of a shortcut is

Algorithm 2: Optimal Optical Bypass in WANs

Inputs:

$G\langle V, E\rangle$: network G, vertices V and edges E
c_e: capacity of edge e
u_e: capacity of one wavelength of edge e
D_d: traffic demand between src_d and dst_d
T_d: set of tunnels for demand d
s: network shortcut due to optical bypass
u_s: capacity of one wavelength of shortcut s

Outputs:

$flow_t$	$\in \mathcal{R}_{\geq 0}$	flow allocated over tunnel t	
x_e^t	$\in \mathcal{R}_{\geq 0}$	flow allocation on edge e for tunnel t	
w_s	$\in \mathcal{N}$	number of wavelengths on shortcut s	**Maximize:** $\sum_s \lvert s\rvert \cdot w_s$
y_s^t	$\in \mathcal{R}_{\geq 0}$	flow allocation on shortcut s for t	

subject to:

(3.1)	$D_d \leq \sum_{t \in T_d} flow_t,$	$\forall d \in D$
(3.2)	$0 \leq x_e^t,$	$\forall t \in T, e \in t$
(3.3)	$0 \leq y_s^t,$	$\forall t \in T, s \in t$
(3.4)	$flow_t \leq x_e^t + \sum_{s \ni e} y_s^t,$	$\forall t \in T, e \in t$
(3.5)	$\sum_{t \ni s} y_s^t \leq u_s \cdot w_s,$	$\forall s$
(3.6)	$x_e := \sum_{t \ni e} x_e^t,$	$\forall e$
(3.7)	$x_e + u_e \cdot \sum_{s \ni e} w_s \leq c_e,$	$\forall e$
(3.8)	$w_s = w_{\overleftarrow{s}},$	$\forall s$

a product of the wavelengths assigned to it by Shoofly and the data rate of the wavelengths' modulation format, u_s. We note that w_s is an integer. The remaining decision variables in the optimization are auxiliary and we define them in the following.

Objective function. To maximize the cost saving from optical bypass, Shoofly maximizes the number of router and optical ports that are *freed* by allocating wavelengths to shortcuts. The OEO conversion of each wavelength at a regional hop occupies a router port and optical port in both ingress and egress directions. Thus, a shortcut s with w_s

wavelengths frees 2 router and optical ports per wavelength at every intermediate hop in the shortcut. The cost saving from a shortcut s is proportional to $w_s \cdot |s|$ where $|s|$ is the number of hops in the shortcut.

Key Insight. Shoofly's goal is not to design the optical backbone from scratch but to leverage existing software and hardware placement to reduce the amortized cost of long-haul capacity – while continuing to meet existing traffic demands in the network. Therefore, we design Shoofly as an algorithm that computes flow allocations on tunnels in the original regional network topology – similar to traffic engineering algorithms [46, 49]. However, in addition to allocating flows, Shoofly *siphons* as much of the traffic allocations from tunnels to the network shortcuts as possible to increase the number of wavelengths that can participate in bypass. The combination of the siphoned flow (y_s^t) and the flow on existing edges (x_e^t) must meet the traffic demands between regions. Wavelengths on shortcuts must be enough in capacity to carry the siphoned flow on the shortcut. By siphoning flow to the shortcuts, Shoofly facilitates the bypass of network capacity while meeting traffic demands.

Demand Constraints. Each demand d between two regions in the WAN has a demand amount D_d and a set of tunnels T_d associated with it. Tunnels are the pre-computed set of k shortest paths between the demand source and destination regions. The set of tunnels T is the union of tunnels $\bigcup_d T_d$ over all demands d. The sum of flow allocated to all tunnels of a demand, should meet the demand:

$$D_d \leq \sum_{t \in T_d} flow_t \tag{3.1}$$

Flow conservation constraints. A shortcut s is a path or a sequence of adjacent edges. The shortcut s is said to be *on a tunnel* t if all edges $e \in s$ belong to t. Similarly, $s \in t$ denotes that shortcut s is on the tunnel t. For each tunnel t, edge e, shortcut s on t, we associate non-negative output variables x_e^t and y_s^t, where y_s^t is the flow that passes shortcut s on t, and x_e^t is the flow that passes edge e, outside of all shortcuts.

$$0 \leq x_e^t, \quad \forall t \in T, e \in t \tag{3.2}$$

$$0 \leq y_s^t, \quad \forall t \in T, s \in t \tag{3.3}$$

The flow allocated to a tunnel t must be carried either on the edges along the tunnel or shortcuts along it. We prove that Equation (3.4) ensures conservation of flow as it is siphoned to shortcuts in Appendix A.1.1.

$$\mathit{flow}_t \leq x_e^t + \sum_{s \ni e} y_s^t, \quad \forall t \in T, e \in t \tag{3.4}$$

Wavelength constraints. The total flow siphoned off to a shortcut must be bounded by the shortcut's capacity, *i.e.,* the product of the number of wavelengths on the shortcut (w_s) and their corresponding data rates (u_e). For instance, if a shortcut consists of two wavelengths that can support 8-QAM modulation, the total flow siphoned to this shortcut must be bounded by 300 Gbps.

$$\sum_{t \ni s} y_s^t \leq u_s \cdot w_s, \quad \forall s \tag{3.5}$$

Capacity constraints. The total flow on edges is the sum of allocations across all tunnels.

$$x_e := \sum_{t \ni e} x_e^t, \quad \forall e \tag{3.6}$$

The capacity of edges, c_e, is reduced due to the migration of some wavelengths from edges to shortcuts that contain the edges. The reduced capacity of edges must be sufficient to meet the total flow allocated to the edges. The reduction in capacity is a product of the number of wavelengths bypassing the edge and the modulation format of the edge. For instance, a wavelength that contributed 200 Gbps on an edge (u_e = 200 Gbps) can be assigned to a shortcut containing the edge, thus reducing the edge capacity by 200 Gbps. This wavelength may only contribute 150 Gbps to the shortcut it is becoming a part of since u_s can be lesser than u_e.

$$x_e + u_e \cdot \sum_{s \ni e} w_s \leq c_e, \quad \forall e \tag{3.7}$$

Bi-directional equality constraints. Links in optical networks can be assumed to be bi-directional. Thus, every shortcut s is also bi-directional and has a reverse $\overleftarrow{s}$. We ensure that shortcuts and their reverse siblings are allocated the same number of wavelengths.

$$w_s = w_{\overleftarrow{s}}, \quad \forall s \tag{3.8}$$

Alg. 2 summarizes Shoofly's optimization formulation using equations (3.1)-(3.8). We will discuss other algorithms that use a subset of the constraints. Thus, we define:

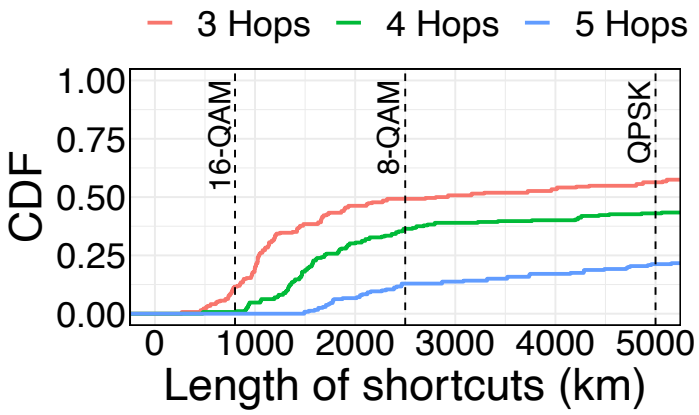


Figure 3.8: The distribution of fiber lengths of network shortcuts. The dotted lines represent the threshold distance for QPSK, 8-QAM and 16-QAM modulation formats.

Definition 3.2.1 (Traffic Allocation Constraints) *Equations (3.2)-(3.8) capture all constraints related to traffic allocation in a network with optical bypass. We define the predicate* $AllocationConstraints$ *as the conjunction of these constraints.*

3.3 Cost savings with Shoofly

We implement Shoofly's optimization algorithm using Python 3 bindings of the commercial optimization solver Gurobi [43]. We note that Alg. 2 solves a mixed integer program (MIP). In practice, Gurobi solves the problem efficiently using a linear programming (LP) relaxation and a MIP gap of 0.1%. Solution to the LP relaxation provides an upper bound for the maximization problem of Alg. 2. The MIP gap defines the break condition for the optimization solver *i.e.,* the solver continues to search for a solution using the branch-and-bound strategy until it finds one within 0.1% of the LP optimal. All instances of Alg. 2 we formulate were solved within 10 seconds of runtime, which is acceptable for build planning.

Fiber lengths of shortcuts. We first evaluate Shoofly on the network topology, traffic demand matrix and optical signal quality of a large commercial cloud provider. The cloud provider we analyze has a global footprint with presence in approximately 100 geographical regions. We enumerate all potential shortcuts of 3, 4 and 5 total regional hops in the network. After 5 regional hops, the signal must undergo regeneration and thus shortcuts of more than 5 hops are not feasible. Figure 3.8 shows the distribution of the lengths of the fiber path in each shortcut. As discussed in Table 3.1, optical signal quality is too low to sustain transmission even at the lowest possible modulation format of QPSK after traversing 5,000 km on fiber without regeneration. Since the regions in the cloud provider are geo-distributed globally, the length of network shortcuts can span several thousand kilometers. In fact, over 50% of shortcuts of all hop lengths are longer than 5,000 km, rendering these shortcuts infeasible (Figure 3.8). Lengths of the remaining feasible shortcuts decide the modulation format that signals on those shortcuts can support. Nearly all 3-hop shortcuts can support 8-QAM or 150 Gbps of data rate per wavelength. Higher hop-count shortcuts can be longer and thus support lower data rates *e.g.,* 100 Gbps.

3.3.1 Reducing hardware costs of capacity

Shoofly identifies wavelengths in the cloud provider's network that can optically bypass regional hops by allocating capacity to pre-computed feasible shortcuts in the network. In this section, we evaluate the cost savings achieved by Shoofly of various practical topologies.

Impact of shortcut length. We formulate three instances of Alg. 2 – first instance considers shortcuts of 3 hops, second considers shortcuts of 3 and 4 hops and third considers

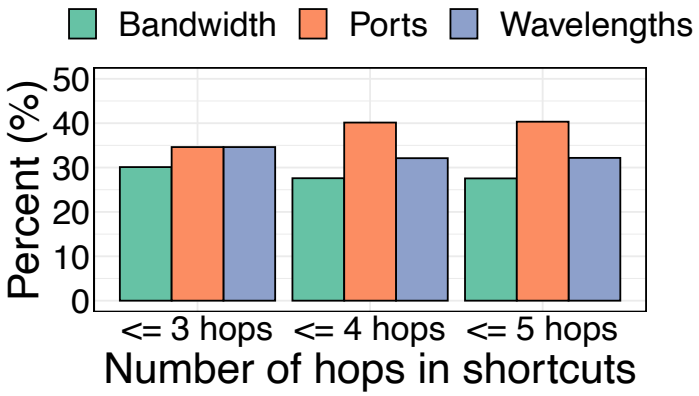


Figure 3.9: Percentage of bandwidth bypassed, ports saved and wavelengths bypassed by Shoofly.

shortcuts of 3, 4, and 5 hops. The shortcut path lengths embody a critical trade-off for Shoofly: longer shortcuts enable higher cost savings by freeing more ports but reduce the data rate of wavelengths on the shortcut. We consider the three different instances of Shoofly based on the maximum permissible shortcut lengths to evaluate this trade-off. We solve the three MIP instances and plot the percentage of total bandwidth allocated to shortcuts, total ports saved by the shortcuts and total wavelengths migrated to shortcuts in Figure 3.9. We observe that while longer length shortcuts save more ports, the total bandwidth on the shortcuts reduces with hop length. This is a direct consequence of the length vs. data rate trade-off. The number of wavelengths migrated to shortcuts remain similar regardless of the shortcut lengths as they are a function of the traffic matrix which remains the same in all three problem instances. The results of Figure 3.9 show that **Shoofly can save over 40% of the hardware costs of long-haul capacity** by freeing expensive router and optical line ports at regional hops.

Impact of over-provisioning in networks. Shoofly ensures that existing traffic demands of the network continue to be met in the bypass-enabled topology. However, cloud wide

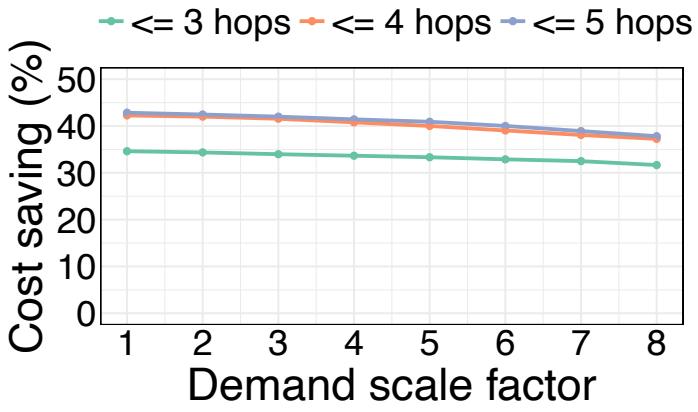


Figure 3.10: Impact of scaling demands on Shoofly's cost savings. Cost savings reduce by 2% as demand is scaled to 8X.

area backbones are often over-provisioned in an attempt to future-proof the network for potential increase in demands. To ensure that Shoofly does not render the bypass-enabled topology incapable of handling increased demands in the future, we evaluate Shoofly on demands that are scaled to 8X the maximum inter-region demands observed in December 2020. Figure 3.10 shows that there is a very small decline (¡2%) in the potential cost savings of bypass as traffic demands are scaled to 8X the maximum present-day demands. Thus, network operators can provision bypasses proposed by Shoofly using scaled traffic demands to make future-proof bypass decisions without sacrificing on cost savings.

Impact of network topology. Next, we evaluate Shoofly on different network topologies. We have detailed information about the network of the cloud provider we study, referred to as cloud provider WAN or CP-WAN in the figures. Additionally, we evaluate Shoofly using the network topology and demand matrices for prominent production networks, released by previous work [11, 57]. We assume that these networks operate a point-to-point optical backbone. Figure 3.11 shows the percentage of ports saved by Shoofly for the backbone networks of Abilene, B4, Nextgen, CP-WAN and a custom topology from previous

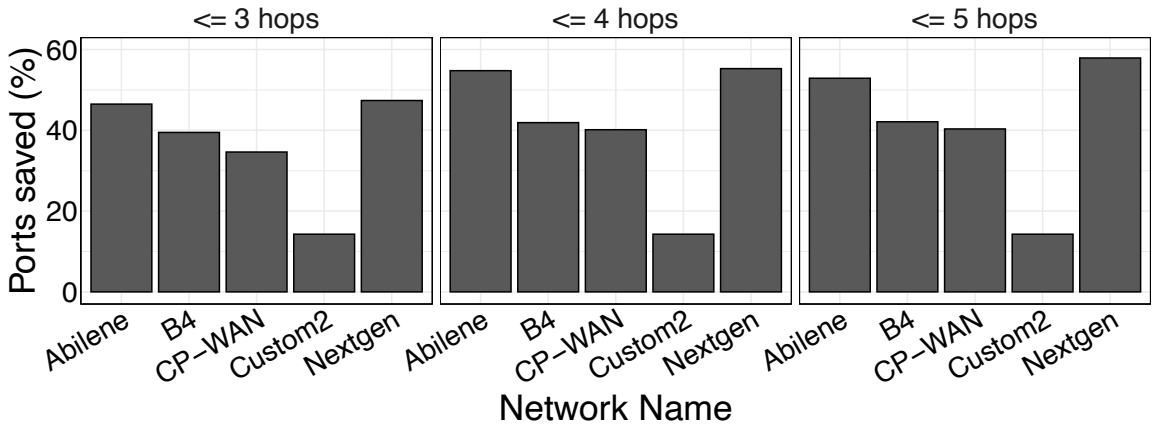


Figure 3.11: shows that Shoofly can save over 40% of hardware costs of long-haul capacity in CP-WAN, B4, Abilene and Nextgent topologies.

work. We find that Shoofly shows a consistent potential of saving hardware costs in all network topologies, ranging from over 55% cost saving in the Nextgen topology to 15% cost saving in the Custom network topology. Figure 3.12 shows the fraction of regions in the networks that participate in optical bypass. Over 50% of the regions in the CP-WAN get bypassed by one or more wavelengths, realizing the bypass potential we found in the inter-regional traffic matrices (§3.1).

3.3.2 Lower data-rates from optical bypass

One key concern raised by optical bypass is that by forcing signals to travel longer distances, a bypass-enabled topology can reduce the capacity between regions. Our evaluation of Shoofly shows that the bypass-enabled topology can not only meet 8X the present-day traffic demands (§3.3.1) but also enable 30-40% hardware cost savings (Figure 3.10).

The reduction in capacity between regions occurs due to a downgrade in signal modulation formats on shortcuts on account of increased transmission distance. Figure 3.13

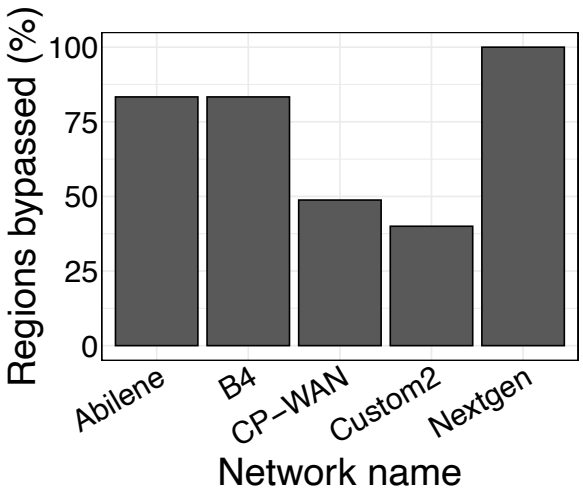


Figure 3.12: shows that majority of regions in all network topologies get bypassed by one or more wavelengths.

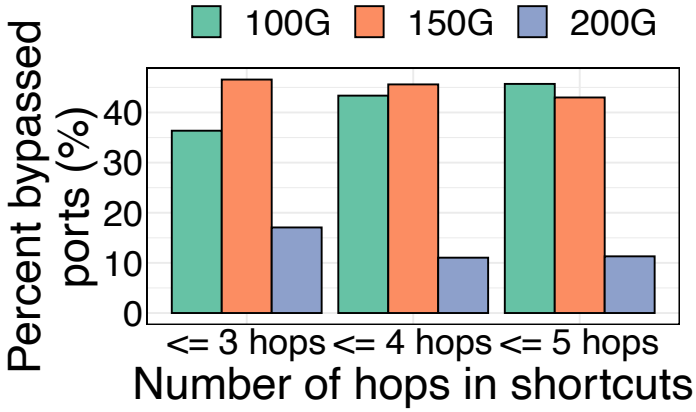


Figure 3.13: Modulation formats on shortcuts of different lengths. As the number of shortcut hops increases, the fraction of ports in higher modulation formats reduce.

shows the modulation formats of the bypasses enabled by Shoofly. We observe that fewer shortcut hops allow Shoofly to keep majority of the wavelengths in higher order modulation formats *e.g.,* of the 3-hop only shortcuts allocated by Shoofly, over 45% can sustain 150 Gbps, 17% can sustain 200 Gbps per wavelength. As Shoofly considers shortcuts with higher hops, it can save more cost (Figure 3.9) but this higher saving comes at the cost of longer shortcut lengths and consequently lower data rates. Figure 3.13 shows that fraction of bypasses that can support higher modulation formats reduce as hop lengths increase. When allowed up to 5-hop shortcuts, only 11% of shortcuts can sustain 200 Gbps. We compare the split of *all* links based on modulation formats in the original and bypass-enabled networks in Figure 1 of Appendix A.1.2.

Network	original	$\leq$ 3-hop	$\leq$ 4-hop	$\leq$ 5-hop
Capacity	160 G	157.5 G	153.5 G	153.5 G

Table 3.2: Norm. per-link capacity of the original network and networks with 3, 4 and 5 hop shortcuts.

To mitigate the concern of lowered network capacity due to bypass, we compute the normalized per-link capacity of the original network and the bypass-enabled networks in Table 3.2. The normalized capacity of a link in the original network is 160 Gbps whereas bypass enabled networks lower the capacity by 4% in the worst-case.

3.4 Failure resilient optical bypass

Enabling optical bypasses fundamentally changes the impact of individual physical link failures on the IP network's ability to carry traffic. For instance, in point-to-point networks, there is a one-to-one mapping from physical to IP links. Therefore, the failure of a physical link (*e.g.,* fiber cut, amplifier failure) leads to an individual IP link's failure.

However, in a bypass-enabled topology one physical link can underpin several IP links and the failure of one physical link can cause *multiple IP links* to fail. Thus, we must revisit failure resilience of the backbone network with optical bypasses.

There is a rich body of work that explores link failure resilience in the *context of traffic engineering* in the WAN. We incorporate their methods of achieving failure resilience to the *capacity provisioning problem* of Shoofly. In doing so, not only do we provision network topologies with link failure resilience baked in, we also show that Shoofly's optimization can be extended to use various reliability objectives. Of these, we design and implement two objectives: resilience to deterministic [62] and stochastic link failures [11]. The work on forward fault correction (FFC) ensures that the cloud TE is resilient to up to k deterministic IP link failures. TEAVaR introduced the concept of TE in the presence of *probabilistic link failures* to meet availability guarantees [11]. In this section we incorporate the resilience to *possible* (§3.4.1) and *probable* link failures (§3.4.2) in Shoofly.

3.4.1 K-wise link failures

First, we discuss provisioning bypass-enabled cloud topologies resilient to the *possibility* of k simultaneous physical link failures. This resilience guarantees that even if k physical links were to fail, the resulting network after bypasses can continue to meet traffic demands. This is important since k physical link failures can translate to more than k failures in the bypass-enabled topology. Today, most cloud providers provision their network to be resilient to $k \leq 2$ link failures.

We formulate the problem of provisioning bypasses under k simultaneous link failures by building on Alg. 2. In addition to the objective and constraints of Alg. 2, this formulation includes a set of constraints for each link failure scenario i, with set of failing links

SRLG_i. For instance, when $k = 1$, each SRLG_i contains each duplex edge. Thus, for each failure scenario i and its set of failing links the following constraints ensure that there is a feasible allocation of flow that meets all demands in the bypass-enabled network:

Definition 3.4.1 (Shoofly under k-wise failures)

Maximize: $\sum_s |s| \cdot w_s$

subject to:

(3.1) $D_d \leq \sum_{t \in T_d} flow_t, \forall d \in D$

(3.2)-(3.8) $AllocationConstraints(flow, x, y, w)$

for each i

$(w1_i)$ $D_d \leq \sum_{t \in T_d} flow_t^i, \quad \forall d \in D$

$(w2_i)$ $AllocationConstraints(flow^i, x^i, y^i, w)$

$(w3_i)$ $flow_t^i \leq 0, \quad \forall t, e \in t, e \in \mathsf{SRLG}_i$

Provisioning under k-wise failures solves for the objective of maximizing cost savings (as described in Alg. 2) while constraining the problem with the goal of finding feasible flow allocations under *all link failure scenarios*. The failure scenario constraints in Def. 3.4.1 share the wavelength decision variables, w_s (Alg. 2) but solve for individual flow allocations ($flow_t^i, x^i, y^i$) for each failure scenario i. Thus, the optimization finds wavelength assignments for optical bypasses while ensuring that feasible flow assignments are found for the original network and the network under every failure scenario. The number of constraints of this formulation grow with the number of failure scenarios considered. Recent work has shown ways of translating such optimization formulations to efficiently solvable models [15]. Our current experiments use the less scalable encoding; which is easier to encode and was sufficient for the current evaluation.

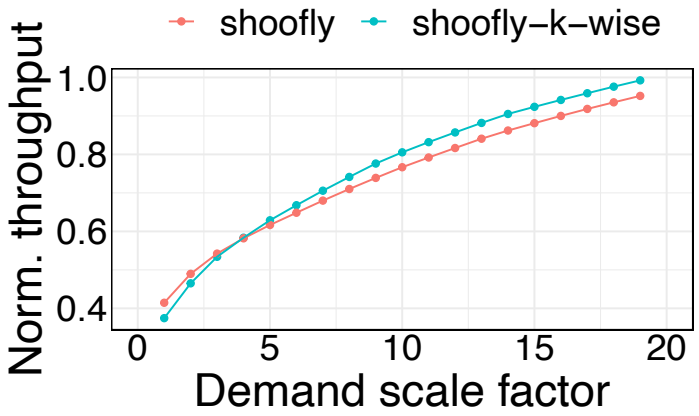


Figure 3.14: Throughput of failure resililent bypass topology.

Shoofly formulates the k failure resilient bypass provisioning problem using Alg. 2 and Def. 3.4.1 constraints for all single and double link failures. We compute vanilla (no additional failure resilience), single link failure resilient and double link failure resilient bypass-enabled topologies. We find the cost savings from failure resilient topologies are virtually indistinguishable from the vanilla topologies (Figure 3 in Appendix). Thus, making Shoofly failure resilient *does not reduce hardware cost savings*.

Evaluation. We implement and solve maximum flow traffic engineering on the two bypass-enabled network topologies: the first topology is without failure resilience and the second is resilient to 2 link failures. We solve several instances of the traffic engineering problem by failing 2 randomly selected links in both topologies for each instance. Thus, each TE problem finds traffic allocations on the bypass-enabled topologies in the event of 2 simultaneous link failures. Figure 3.14 shows the throughput of traffic engineering on the topologies as the demand between nodes is scaled from present-day demands in the cloud network to 20X the present-day demands. As expected, the throughput rises linearly as demand scale increases, until the increase becomes sub-linear due to the network

capacity limits. At low demand scales, both networks achieve similar throughput since double link failures do not stress over-provisioned cloud networks. However, at high demand scales, the failure resilient Shoofly topology achieves 5% higher throughput than the vanilla topology.

3.4.2 Probabilistic link failures

Recent work has proposed a cloud traffic engineering algorithm, TEAVaR, that computes flow allocations that minimize the expected un-met traffic demands, called *loss* or VaR_β, under probabilistic link failure scenarios [11]. The algorithm takes as input the likelihood of link failure scenarios (Q) and target network availability (β) to compute flow allocations. The minimal loss (VaR_β) is guaranteed with probability β and the expectation over all scenarios where the loss is greater than VaR_β is calculated by $CVaR_\beta$ or conditional value at risk. We augment Shoofly with TEAVaR's demand constraints to provision bypasses resilient to probabilistic link failure scenarios. VaR_β and $CVaR_\beta$ are approximated by the outputs α and F_β, in the following optimization problem:

Definition 3.4.2 (Shoofly with TEAVaR)

Minimize: $F_\beta(\alpha)$

subject to:

(3.1)	$D_d \leq \sum_{t \in T_d} flow_t,$	$\forall d$
(3.2)-(3.8)	$AllocationConstraints(flow, x, y, w)$	
(S)	$\sum_s \lvert s \rvert \cdot w_s \geq S$	$\forall s$
$(t1)$	$F_\beta(\alpha) := \alpha + \frac{1}{1-\beta} \sum_{q \in Q} p_q \cdot s_q$	
$(t2)$	$s_q \geq t_{d,q} - \alpha,$	$\forall d, q$
$(t3)$	$s_q \geq 0,$	$\forall q$
$(t4)$	$t_{d,q} := 1 - \frac{\sum_{t \in T_d} flow'_t z_t(q)}{D_d}$	$\forall d, q$
$(t5)$	$AllocationConstraints(flow', x', y', w)$	

In the formulation, p_q is the probability of the failure scenario q, s_q is the loss in failure scenario q, $z_t(q)$ is 0 iff an SLRG on the tunnel t fails in scenario q. The constraints $(t1) - (t5)$ from Def. 3.4.2 can be used for fixed values of w for online traffic engineering to minimize the conditional value at risk, $F_\beta(\alpha)$. For optimizing cost savings through bypass, we add constraints (3.1)-(3.8) from Alg. 2. Since there are now two competing optimization objectives, maximizing shortcut savings vs. provisioning for stochastic reliability, we introduce constraint (S) for selected lower bounds of savings. By choosing different values of S, Shoofly can find Pareto optical bypasses that meet traffic demands in case of probabilistic failures and save a minimum of S in cost by minimizing the objective $F_\beta(\alpha)$.

Setup. We first we enumerate the likelihoods of link failures in the original network topology by sampling from a Weibull distribution with $k = 0.8$ and $\lambda = 0.0001$ to populate

Q, similar to previous work. We use the traffic demands, network topology and enumerated shortcuts of the cloud network (as in §3.2). Using the simulated failure scenarios in Q, we enumerate availability values (β) for the cloud topology to solve TEAVaR's flow allocation problem. Of the enumerated βs, we choose the one for which TEAVaR can find allocations on the original network with 0 loss and $F_\beta(\alpha) < 0.01$. Equipped with Q and β, we solve for bypass allocations (w_s) using Shoofly and Def. 3.4.2.

Evaluation. We set S in the savings constraint $\sum_s |s| \cdot w_s \geq S$ to fractions of total savings possible with Shoofly and measure the minimum conditional value at risk (CVaR) calculated in the optimization solution. The total savings possible are found by solving the vanilla Shoofly formulation, without additional failure resilience constraints. Figure 3.15 shows the relationship between cost savings and CVaR for different maximum shortcut hops. We note that Shoofly with only 3-hop shortcuts can achieve 80% of the cost savings possible but when the savings constraint is applied to achieve 100% of the cost savings, a shortcut allocation is not possible. With higher number of shortcut hops, it is possible to achieve the maximum cost savings but the risk of un-met traffic demands increases between 80% and 100% cost saving.

3.5 Operational safety & logistics

In this section we discuss the implications of a bypass-enabled cloud network topology on the uses it is put to. We focus on the logistical burden of deploying Shoofly's proposed network topology in the cloud WAN. We also discuss the impact of bypass-enabled network on traffic engineering algorithms since these systems and algorithms rely on the network topology to make efficient use of the resources.

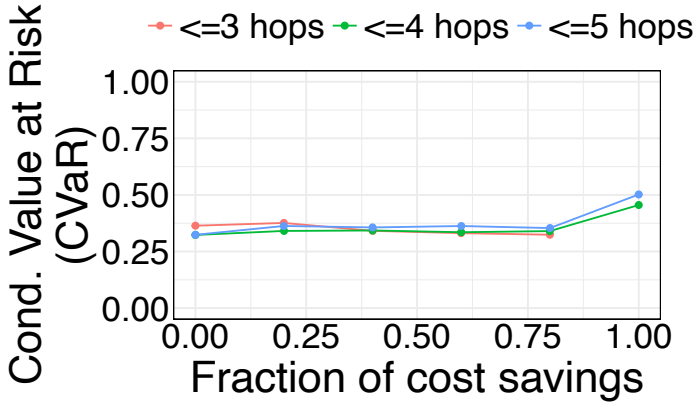


Figure 3.15: Achievable cost savings vs. CVaR with Shoofly.

3.5.1 Bypass implementation plan

We devise a *bypass plan* to simplify the logistics of deploying optical bypass in the cloud WAN. We evaluate the contribution of each instance of optical bypass to the hardware cost savings discussed in Section 3.3. We identify an instance of optical bypass as a triplet of regions $A \rightarrow B \rightarrow C$ where Shoofly proposes that some wavelengths traversing the regions bypass region B en route from A to region C. Thus, each bypass is identified by the triplet of regions and it represents a unit of logistical overhead faced by the cloud operators to implement a bypass-enabled network. We compute the fraction of total savings enabled by every bypass Shoofly computes. Figure 3.16 shows the relationship between fraction of shortcuts and the cumulative savings enabled by them. 25% of bypasses contribute to 80% of all hardware cost savings. This shows high return on logistical investment in implementing Shoofly's recommended optical bypasses.

3.5.2 Traffic engineering with shortcuts

To use Shoofly's bypass-enabled network for TE, the operator must convert the TE tunnels of the original network to new tunnels in the bypass-enabled topology. This change

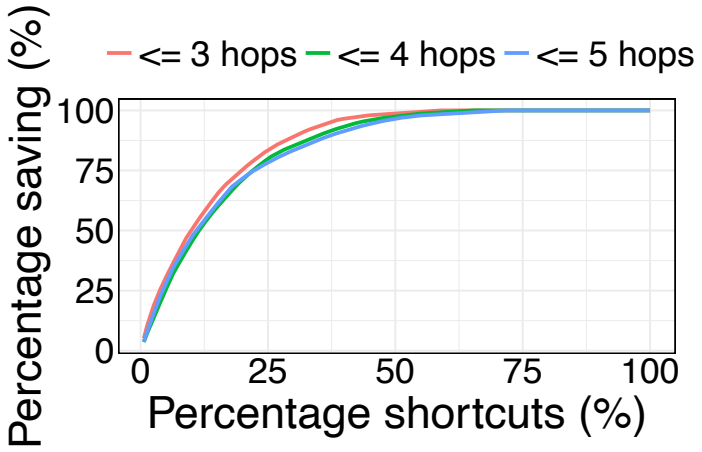


Figure 3.16: 25% of all bypasses contribute to 80% of all hardware cost savings proposed by Shoofly.

is needed since (1) edges composing tunnels in the original network may have gone away after bypasses are enabled (2) new edges may have been added between the nodes on the tunnels as shortcuts. Therefore, each original tunnel can spawn several tunnels between a pair of nodes. We evaluate the growth in the number of tunnels with the following experiment. For every tunnel in the original network, we find the new edges added by bypass and adjust the capacity of existing edges based on Shoofly's output. We calculate all simple paths between the tunnel start and end nodes. We plot the increase in the number of tunnels in the bypass-enabled topology and find that the number of tunnels can increase by a factor of 3 (Figure 3.17). We suggest that network operators prune the set of new tunnels to ensure similar run time of traffic engineering algorithms on the new topology.

In place of adding separate tunnels, the operator can formulate the TE solver to be shortcut-aware. In this formulation, the standard TE capacity constraints ($\sum_{t \ni e} flow_t \leq c_e$) are replaced by the capacity constraints from Alg. 2, where the output variables w_s are fixed. The remaining output variables capture the allocation of flow on edges and

Figure 3.17: Expansion in number of total tunnels after introducing optical bypasses compared to the original network.

shortcuts. Once the shortcuts capacities have been determined by fixing the number of wavelengths assigned ($\mathbf{w_s}$) to all shortcuts in the network, TE on the resulting network is defined as:

Definition 3.5.1 (TE with shortcuts)

Maximize: $\sum_{t \in T} flow_t$

subject to:

*(*TE*1)*	$D_d \geq \sum_{t \in T_d} flow_t,$	$\forall d \in D$
(3.2)-(3.8)	$AllocationConstraints(flow, x, y, \mathbf{w})$	

The inequality on demands is reverse from Alg. 2 since the goal of TE is to maximize throughput using fixed resources.

3.5.3 Impact of bypass on TE tunnels

Since optical bypasses re-allocate capacity between old and new edges in the network, they can limit communication between some node pairs. Shoofly ensures that demands between all node pairs that communicate *in the present-day cloud network* can be met even

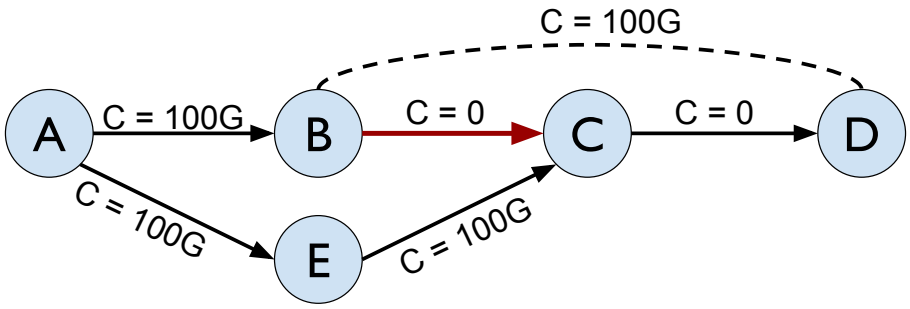


Figure 3.18: Tunnel AECD is removed when wavelengths of edges BC and CD are allocated to shortcut BD. Nodes B and C can no longer communicate in the bypass-enabled network.

if the demand between them increases 8-fold. However, there is no direct traffic demand between some nodes at present and Shoofly can allocate bypasses that prevent the nodes from communication directly. Figure 3.18 illustrates a case where shortcut allocations can *starve* traffic patterns permissible in the original network.

We can prevent Shoofly from starving traffic patterns that are possible in the original network topology by ensuring that a minimal amount of capacity remains available on edges and tunnels after the allocation of shortcuts. Adding inequalities (3.9) and (3.10) to Shoofly's Alg. 2 will achieve this:

$$x_e \geq lb_e, \quad \forall e \tag{3.9}$$

$$x_e + \sum_{s \in t, e \in s} u_s \cdot w_s \geq lb_t, \quad \forall t \in T, e \in t \tag{3.10}$$

3.6 Related Work

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

Optical network design. Service providers have studied the design of optical networks in depth [17, 40, 6, 68, 9, 69, 12]. We bring two unique perspectives to this rich field of research: first, our analysis leverages optical signal quality and traffic matrices from a production network. Second, our analysis focusses on a large commercial cloud provider. Cloud networks are designed for different workloads than service networks and the demand matrix between regions is a function of the centralized TE algorithms in addition to user demands. Internal traffic patterns in SDN controlled cloud networks tend to be stable and predictable [84] helping the design of Shoofly.

Cross-layer network optimizations. Recently, researchers have proposed cross-layer optimizations between IP and physical layers to achieve latency gains for deadline-driven bulk transfers [53]. While related, Shoofly is solving a provisioning problem and not a scheduling one. Similar to the ideas explored by Shoofly, researchers have found that transceivers in data centers can be "stretched" to lower the cost of data center networks [92]. Unlike Shoofly, these works take advantage of high signal quality on data center links to use transceivers for longer distance connections.

Wide area performance monitoring. Researchers have studied optical signal quality in the WAN [75, 27, 34, 36] and found that existing optical signals can be utilized to enable data rates. However, these studies have not utilized the high OSNR to reduce OEO conversion like Shoofly does.

Intra-WAN Traffic engineering (TE). Cloud providers have embraced software-defined, centralized TE controllers to assign flow in their WANs to maximize their utilization, guarantee fairness and prevent congestion [46, 49, 75, 60].

Wide-area failure recovery has been extensively studied in the context of network engineering. The original work in [31] considered safe re-allocation of routes without incurring congestion or breaking reachability under network and demand changes. Bringing these ideas to TE, K-wise failure resiliency was developed in [62]. It was first solved using an optimized encoding of enumerating failure scenarios using sorting networks and reformulated using LP dualities in [63]. A general framework based on LP dualities was initiated in [16], and shown practical in [15] based on insights that ensure strong LP dualities. SLA guarantees through TE in probabilistic failure scenarios were explored in [11].

3.7 Conclusion

We analyzed the inter-regional traffic patterns in a cloud WAN and found that 50% of the traffic observed by a region is *passing through* – neither originating or terminating at the region. We propose that such traffic optically bypass regions and stay in the optical domain for as long as is possible, thereby saving hardware costs of long-haul capacity. We propose a tool, Shoofly to find optical bypass opportunities in the WAN such that the hardware cost of long-haul capacity is minimized. We show that despite the physical constraints of limited optical reach of signals on fiber, Shoofly provisions failure-resilient backbones that save 40% of hardware cost using existing network hardware without impacting the network's ability to meet traffic demands.

CHAPTER 4

COST-EFFECTIVE CLOUD EDGE TRAFFIC ENGINEERING

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 [49] and Microsoft [46] 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 [87, 72]. 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 [79] 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.

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

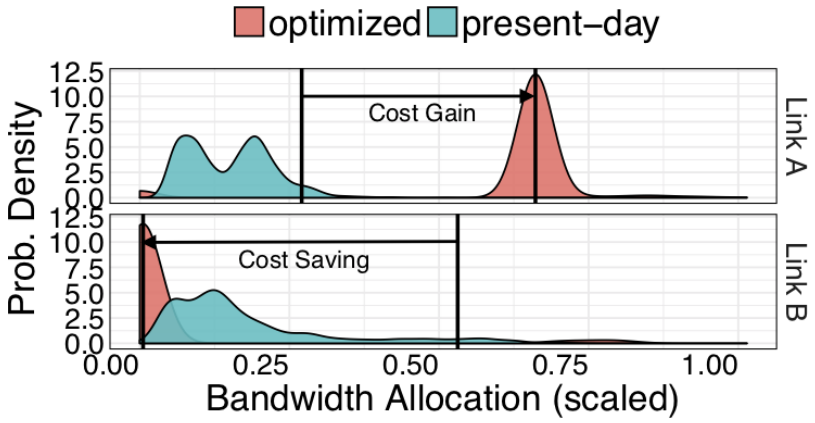


Figure 4.1: Present-day and Cascara-optimized bandwidth allocation distributions for one week, across a pair of links between a large cloud provider and tier-1 North American ISPs. Costs depend on the 95th-percentile of the allocation distributions (vertical lines). Cascara-optimized allocations reduce total costs by 35% over the present-day allocations while satisfying the same demand.

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 [18]. As a result, most clients are reachable over several short and *latency-equivalent* paths from the cloud provider [71]. We illustrate the cost saving potential due to latency-equivalent links with an example in Figure 4.1. 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 95^{th} percentile of the bandwidth distribution 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 (§4.2). 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 (§4.2) and offering a significant improvement over related approaches in [50, 38, 91].

2. Practical and cost-efficient online edge TE. Since optimizing percentile costs is NP-Hard [50], finding optimal solutions can take several hours. We impose structure on the optimization problem based on 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).

3. Flexibility to balance the cost-latency tradeoff. Cascara incorporates the latency of primary and alternate peer paths from the cloud [72, 14] 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 (§4.4).

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 low-priority, 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 (§4.5).

4.1 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

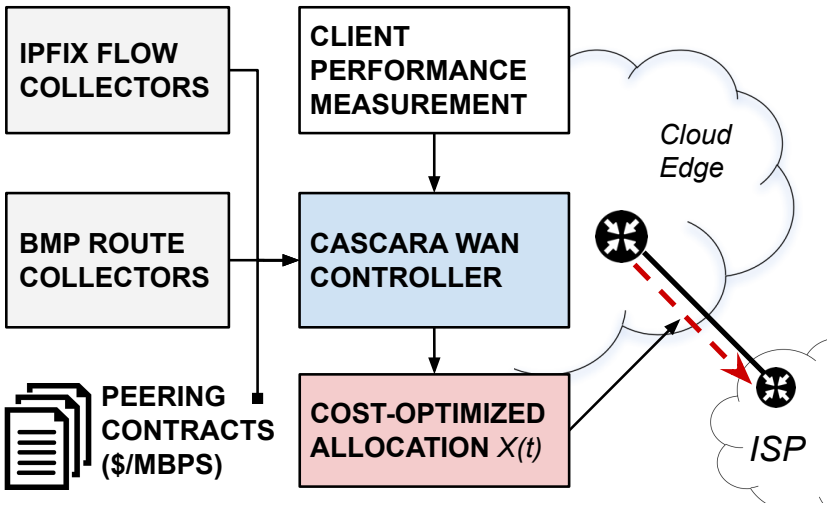


Figure 4.2: Design of our traffic engineering framework, Cascara.

safety levers to the operator: configurable variation in the traffic allocations to peer links, incremental deployability and bounded impact on client latency. Figure 4.2 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) [83] 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 (§4.2) and online allocation to meet demands by Cascara (§4.3 and §4.4).

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.

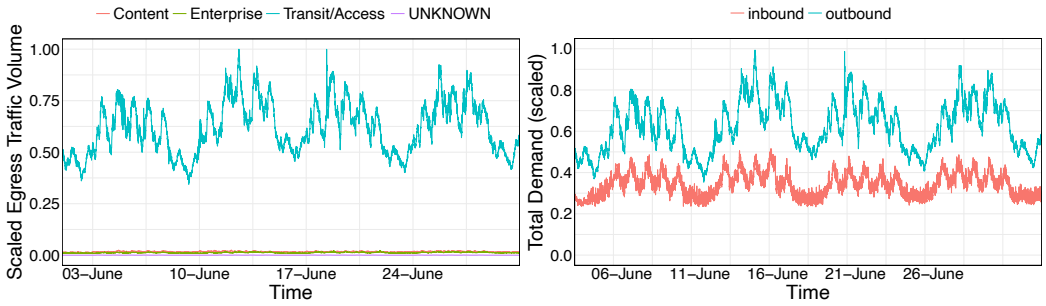
Peering Contracts. We feed the billing models and peering rates for all BGP peers of the cloud provider to Cascara. Since peering rates remain stable over short durations of time, we use snapshot of this information from June 2019.

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 [87, 72]. 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 (§4.2), then develop an efficient, online and near-optimal algorithm for Cascara to realize the saving potential (§4.3). Finally, we put Cascara to test with realistic client performance and route availability constraints in §4.4.

4.2 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 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



(a) Outbound traffic types.

(b) Inbound/Outbound traffic.

Figure 4.3: (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.

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 4.3a 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 [13].

4.2.1 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 [24]. 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 $P1$ to peer $P2$ 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 $P1-P2$ link. The link cost for a billing cycle is given by, $B = c_i * \text{MAX}\{B_{out}, B_{in}\}$, 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 [24].

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 4.3a). Therefore, traffic exchanged with transit ISPs is the main contributor to bandwidth costs incurred by the cloud provider.

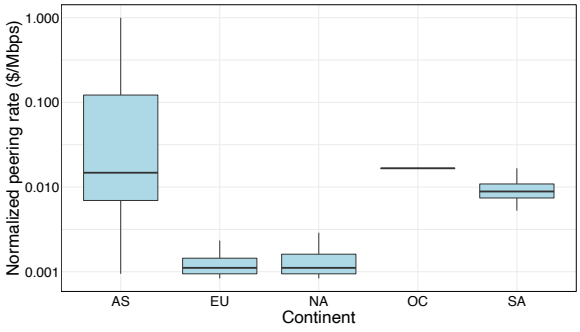


Figure 4.4: Large differences in the cost per unit bandwidth in different parts of the world *e.g.,* the median peering cost in Asia is over 10X the median peering cost in North America.

Outbound traffic is twice the inbound. For the cloud WAN, outbound traffic volume is nearly twice the inbound (Figure 4.3b), highlighting that the cost computation based on link utilizations can be simplified to $c_i * B_{out}$ for clouds networks.

Links with only three ISPs contribute majority of costs. Due to the large variance in peering rates (seen in Figure 4.4) 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.

4.2.2 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 4.3b), 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 $P = \{P_1, P_2, ..P_m\}$ 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 P, for every timeslot $t_j, j \in [1, .., n]$. Let x_{ij} be the decision variable, where x_{ij} is the flow assigned to peering link P_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 P_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:

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

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 [50]. We later show that techniques from previous work are ineffective in saving bandwidth costs of edge links (§4.3.3). 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 timeslots 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:

$$(\lambda_{ij} == 0) \implies z_i \geq x_{ij}, \forall i, j \tag{4.1}$$

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 [42]. 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 3 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 [22] framework and solve it with the commercial optimization solver, GUROBI [43] on a machine with 12 cores and 48 Gb RAM. Our choice of solver is motivated by the computational complexity of Algorithm 3. Commercial solvers like GNU LPK [37] and CPLEX [47] were orders of magnitude slower than GUROBI in solving our formulation.

Algorithm 3: WAN Egress Traffic Allocation

Inputs:

n: number of five-minute timeslots in a month

m: number of peering links in the WAN

P_i: Peering link $i, i \in [1, .., m]$

C_i: capacity of peering link P_i

c_i: peering rate (USD/Mbps) for link P_i

d_j: egress flow from the WAN in $t_j, j \in [1, .., n]$

$k = \frac{n}{20}$

M: large integer constant

Outputs:

x_{ij}: traffic allocation to link P_i in timeslot t_j

λ_{ij}: binary variables that discount top-k x_{ij}s

z_i: billable bandwidth on link P_i

Minimize: $\sum_i z_i * c_i$

subject to:

$\forall i, 0 \leq x_{ij} \leq C_i$; `// capacity constraints`

$\lambda_{ij} \in \{0, 1\}$; `// binary constraints`

$\forall j, \sum_i x_{ij} = d_j$; `// demands constraints`

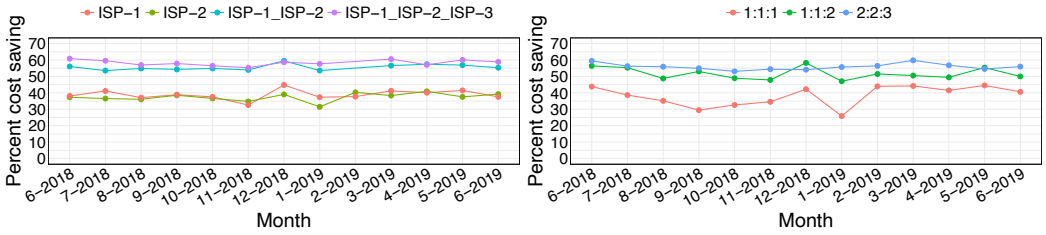
$\forall i, \sum_j \lambda_{ij} = k - 1$

$\forall j, z_i > x_{ij} - M * \lambda_{ij}$

4.2.3 Generalizable and large saving potential

Using the set of peering links (P), 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. 3. 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. 3, one for each 1-month billing period. We discuss the implementation details and assumptions in §4.2.4 and offer a preview of the results here. We compare the cost of allocations computed by Alg. 3 with the real allocation cost incurred by the cloud provider

and find that Algorithm 3 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 4.5a).



(a) Impact of participating links on saving. (b) Impact of peering rates on cost saving.

Figure 4.5: (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.

Impact of participating links. When the input to the optimization is an *individual* 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 4.5a. 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 Figure 4.5a, 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 4.5b 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 geo-political 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%.

4.2.4 Computing optimal traffic allocations

We now discuss the details of our implementation of Alg. 3.

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 3. We take advantage of our findings from §4.2.1 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 3 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 3, 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 3 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 P) *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 3 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.1.1).

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.

4.3 Online cost-optimization with Cascara

Results of the offline allocation scheme (4.2.2) 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 3) 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 3. 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 (4.2.2) 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 4.2.4, 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 4.6 shows the density distribution of optimal allocations on an edge link by Algorithm 3. We note that the optimal allocations

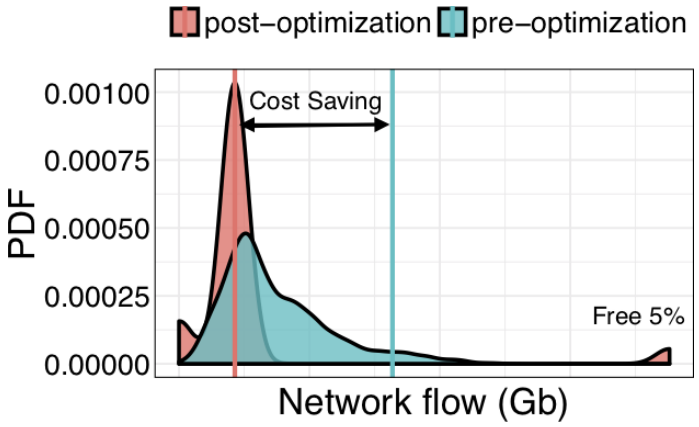


Figure 4.6: Optimized allocations on an edge link for a month. The vertical lines show the pre- and post-optimization 95^{th} percentile utilization on the link. (X-axis labels removed.)

reduce the link's 95^{th} percentile utilization to $\approx 15\%$ of its capacity. 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 1).

4.3.1 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 [7] 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 timeslots. 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.1.2). Since u_is are derived from C_f, the key input parameter to Cascara is C_f. Algorithm 4 shows the online traffic allocation scheme of Cascara in brief (details in Appendix Algorithm 5).

Insufficient C_f and infeasible allocation. If the initial capacity fraction assigned by Cascara ends up being insufficient to meet the demand in a timeslot, 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

Algorithm 4: Online Traffic Allocation Per-Timestep

```
1  Function allocate_timestep(d, f):
2      if d ≤ C_f then
3          allocate C_f to links in L
4          return true
5      else
6          d = d − C_f
7          while linkqueue do
8              l = pop(linkqueue)
9              augment l
10             decrement l's priority and free slots
11             decrement d by l's augmented capacity
12             if d ≤ 0 then
13                 return true
14             end
15         end
16         return false
17     end
```

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.

4.3.2 Finding Cascara's hyperparameters

We show that by setting C_f effectively, Cascara's online traffic allocation (Algorithm 4) can be nearly as effective as the offline solutions of Algorithm 3. 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[1] 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 optimal C_f since links' augmentation slots are wasted before the increment is made. Thus, preemptively

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

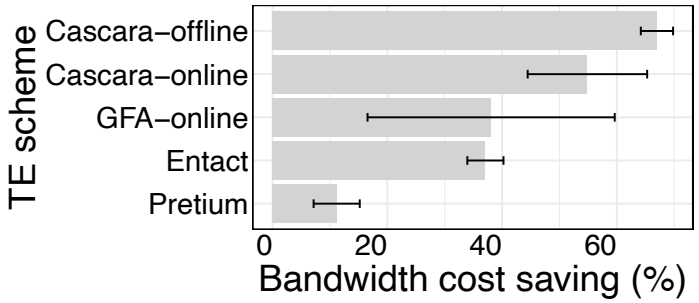


Figure 4.7: Costs savings from Cascara and related approaches. Bars show the mean and wiskers show the standard deviation.

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.1.5. The best values of α and β are used to for the following discussion.

4.3.3 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 4.7). 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 [39, 58] as these are not viable for the cloud provider we study (§4.6). The three main systems from related work are:

Pretium for dynamic file transfers in the WAN [50]. Pretium focuses on optimizing percentile costs of internal WAN links for dynamic transfers within the WAN [50]. They

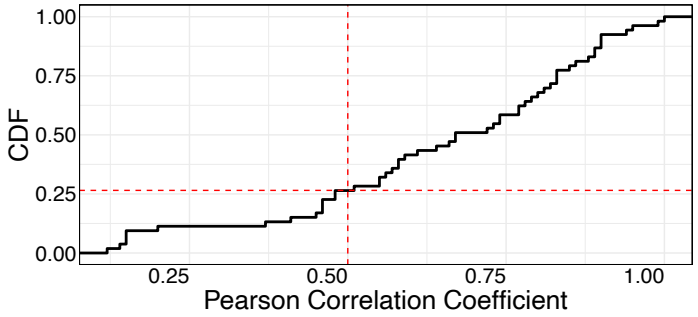


Figure 4.8: 95^{th} %-ile and avg. of top 10% correlation.

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* [50]. We evaluate this assumption using the utilizations of over 50 peering links from the cloud WAN to large ISPs in N. America. Figure 4.8 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.

Entact for cost minimization in clouds [91]. 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 [91]. 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 4.7'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 [38]. Finally, authors of [38] 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 4.7. 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 §4.4.

4.3.4 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 assigning 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 4.9 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 is considerably more stable than present-day allocation which is driven by user-facing demands. There are two reasons for this. First, a pre-selected fraction of a link's capacity is the utilization on the link for 95% of billing slots and changes are made to this fraction only when it is essential. 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 as opposed to being prepared for arbitrary spikes in traffic.

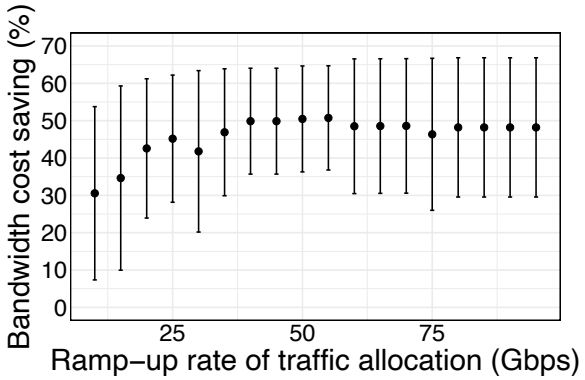


Figure 4.9: The impact of ramp-up rate on cost saving potential (mean and std. deviation calculated over 12 billing cycles).

Incremental deployability. The input links and demands to Cascara can be selected by the cloud provider incrementally by first deploying it in a region or a PoP. To demonstrate 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 4.10 shows that Cascara-online can achieve near-optimal cost (Cascara-offline) savings across all peer links in North America (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. 3) due to the LP gap in the solution of the MILP (§4.2.4).

4.4 Performance-aware cost saving

We have demonstrated that there is significant potential of saving inter-domain bandwidth costs in a cloud network (§4.2) and Cascara's efficient online algorithm can realize this potential by achieving near-optimal cost saving (§4.3). In this section we discuss prac-

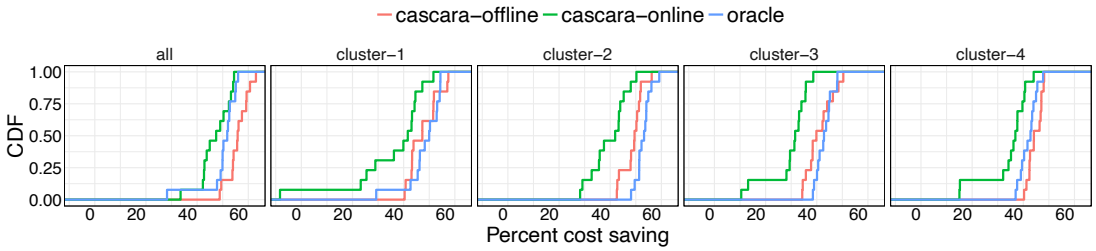


Figure 4.10: shows distributions of percent cost saving by Cascara-offline, Cascara-online and the oracle (Algorithm 3) over 12 billing cycles. Cascara-online achieves near-optimal saving with different sets of links and corresponding demands as input.

tical aspects of achieving cost savings, namely, feasibility of engineering egress traffic in a WAN and the impact of Cascara on client latency.

4.4.1 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 blackholed 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 4.11). 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 significantly reduces the scale of the problem. This grouping of client prefixes within the same AS and small geographical locality has been used effec-

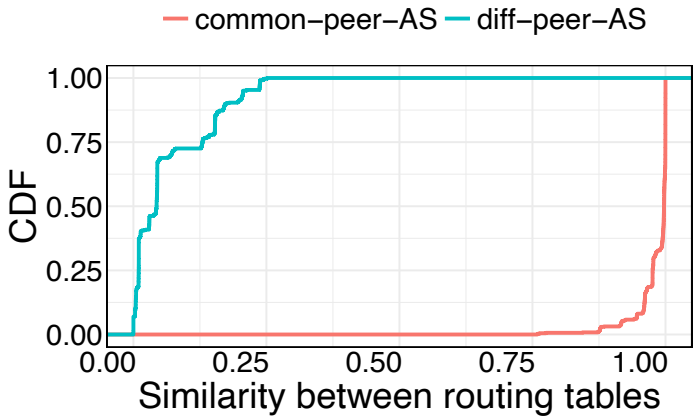


Figure 4.11: shows the distribution of address space similarity between peer links of ISP-1, ISP-2 and ISP-3 at different PoPs of the cloud. Each ISP announces nearly the same address space at different PoPs but the overlap in address space across ISPs is very small.

tively in previous work [14]. 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. 4 that maps clients demands to peer links.

4.4.2 Bounded impact on client performance

After ensuring that traffic towards clients is engineered to peer links with the necessary BGP routes, 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

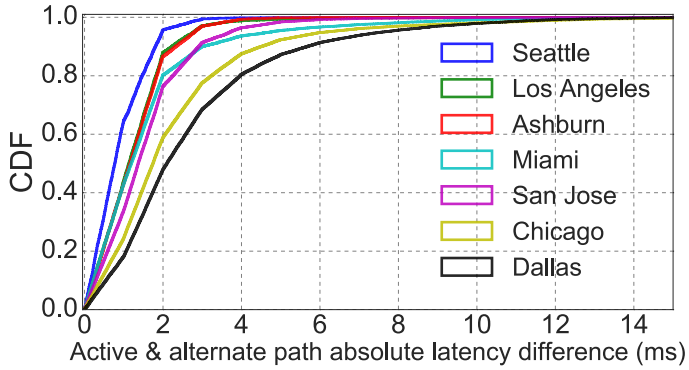


Figure 4.12: The difference in median latencies between primary and alternate BGP paths calculated from client measurements in 15 minute intervals in August 2020. At all PoPs, the difference between latencies of primary and alternate paths is small.

at eight PoPs [91, 72, 71]. These PoPs were selected as they carry high traffic volume – approximately 47% of all the cloud provider's North American traffic, and the presence of 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 [71], 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 4.12 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 over-provision backbone capacity [19] and manage intra-WAN link utilizations with centralized controllers like SWAN [46] and B4 [49] 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 4) 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 4.12).

We use Cascara to engineer traffic at each PoP and compute the offline cost optimal solutions (Figure 4.13) for comparison. We find that 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. 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 worst-case latency degradation that is acceptable while saving bandwidth costs.

4.5 Discussion

We dig into the source of Cascara's cost saving potential and explain how it outperforms common heuristic-based cost saving methods popular in the industry, namely, load balancing of traffic over links and using BGP *localpref* to prefer cheaper links at edge routers. Finally, we discuss potential implications of our findings on peering contracts with ISPs and bandwidth pricing on the Internet.

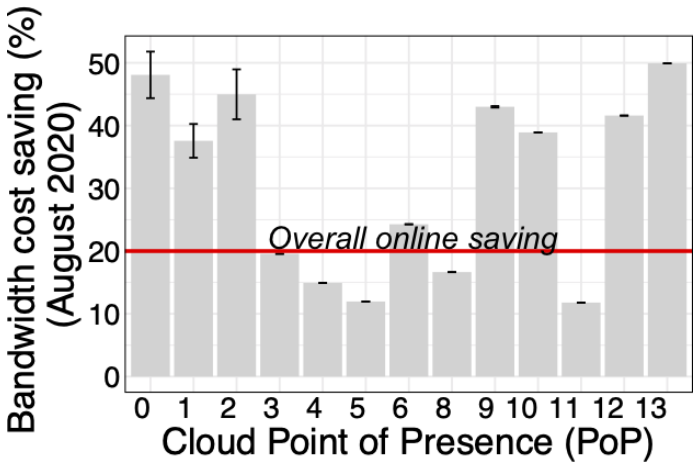


Figure 4.13: shows cost savings with Cascara when engineering traffic on a per-PoP basis while limiting the impact on client latency to 3 ms in the worst case (mean and standard deviation computed over three runs of Cascara).

4.5.1 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 and based cost saving is sub-optimal. We illustrate the benefit of Cascara with a small example using only 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 demand in the network is the total traffic that needs to be assigned to Link 1 and 2 in any given time slot (Figure 4.14). Traffic should 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 in three time slots. Say, the network follows the strategy of load- balancing the 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 4.14) for both links. The median utilization

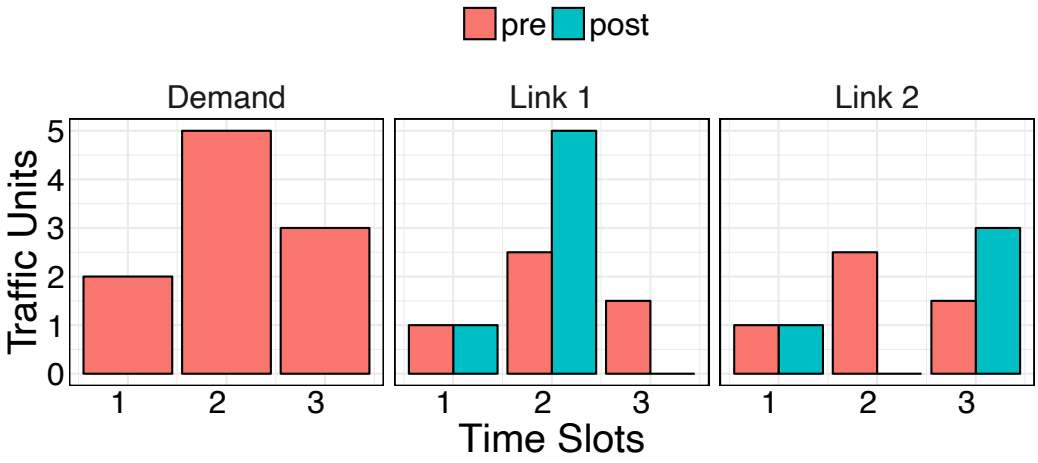


Figure 4.14: A toy example. Outbound demand must be assigned to two links in three time slots. The orange bars show the demand and its allocation on the links when the allocation strategy is to load-balance. The links are billed by the median of their utilization. The blue bars show an alternate strategy that uses the *free* time slot of each link to reduce their combined median cost from 3 to 2 units.

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 4.14, 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.

Cascara utilizes free time slots effectively. As seen in the example, 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 and using it to absorb peaks in demands can reduce the cost. Similar to the example, bandwidth on the Internet is priced using 95^{th} percentile billing, allowing for 5% of 5-minute time slots in a given month to be *free* for each link. This im-

plies 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.

4.5.2 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 [18, 86]. 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 [4, 71]; also allowing these networks to optimize bandwidth costs with stable performance.

Cost optimization cannot be a one-time effort. Traffic patterns across billing slots do change – demands have been rising steadily at 30-40% per year and the surge in demand [21] from the COVID-19 global pandemic has only made traffic patterns more dynamic. We have evaluated Cascara using more than 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.

4.5.3 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 [59]. 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.

4.5.4 Implications for bandwidth pricing

With findings from 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 [64], inter-domain bandwidth pricing models and rates determine paths taken by traffic to customers and subsequently the end-user performance. With the changing structure of the Internet topology, 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 [32, 82]. 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. Recent proposals suggest ways to better align the cost of Internet transit and the revenue gained by networks [45, 90].

4.6 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 [46, 49, 60]. Bandwidth costs in the context of WANs were considered in Pretium [50] (comparison with Cascara in Section 4.2.4). Stanojevic *et al.* used Shapley values to quantify the value of individual flows under percentile pricing [78].

Engineering the WAN egress. Recent work has proposed a software-defined edge to manage outbound flows from their networks [87, 72]. 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 [70, 38] and we compare Cascara with a representative set of work from this space (§4.2.4).

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 [58]. This work is complementary to Cascara since 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 [87] 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 [71,

4]. 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 [64]. 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.

4.7 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. We evaluate this scheme using traffic patterns from the production network and find that the cost savings are robust to changes in peering rates and traffic patterns. Finally, we demonstrate how Cascara can balance the cost-performance tradeoff by achieving 11-50% cost savings per cloud PoP without degrading client latency significantly.

APPENDIX

SHOOFLY APPENDICES

A.1 Appendix

A.1.1 Flow conservation in Shoofly

Lemma 1 *Every solution that satisfies $AllocationConstraints$ also has a solution where inequalities (3.4) are tight and corresponds to a network flow. That is, the flow entering each internal node in a tunnel equals the flow leaving.*

Proof 1 *First note that the inequalities on y_s^t and x_e^t are convex, equations (3.2) and (3.3) remain satisfied when decreasing their values. Therefore a solution to the inequalities (3.4) implies that there is a solution where the inequalities are tight equalities. Let v be an internal vertex on a tunnel t with incoming edge e and departing edge e', then a tight solution to (3.4) implies that*

$$x_e^t + \sum_{s \ni e} y_s^t = x_{e'}^t + \sum_{s \ni e'} y_s^t.$$

Shortcuts that don't terminate in v are included on both sides of the equalities. They cancel out. The equality is preserved for the remaining flows.

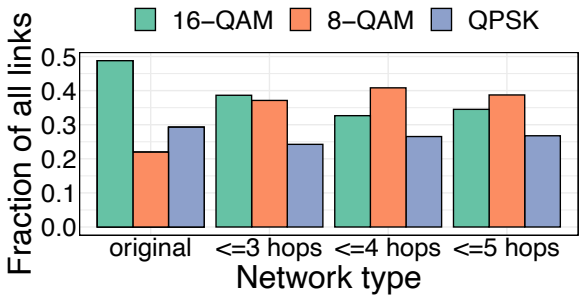


Figure 1: Modulation formats of all links in the original and Shoofly proposed networks.

A.1.2 Overall link modulation formats

We discussed the modulation formats of signals in bypass-enabled topologies in Section 3.3. In Figure 1 we show the comparison of modulation formats of signals between the original network topology and the bypass-enabled topologies. Nearly 50% of signals in the original network could sustain 16-QAM format but this fraction declines by nearly 10% in the bypass-enabled topology.

A.1.3 Throughput of TE on bypass-enabled networks.

We discussed the decline in per-wavelength capacity due to bypasses in Section 3.3. In addition to the average link capacity decline, we simulate the throughput of traffic engineering on both Shoofly provisioned topologies without additional failure resilience and with resilience to 2 simultaneous link failures. First, we show that at as demand is scaled higher, the failure resilient topologies achieve higher network throughput (Figure 2). Second, regardless of the scale of demands, topologies with longer shortcut lengths achieve less throughput, especially at high demand scale factors. At present day demands, the difference between the throughput of 3, 4 and 5 hop shortcut topologies is very similar.

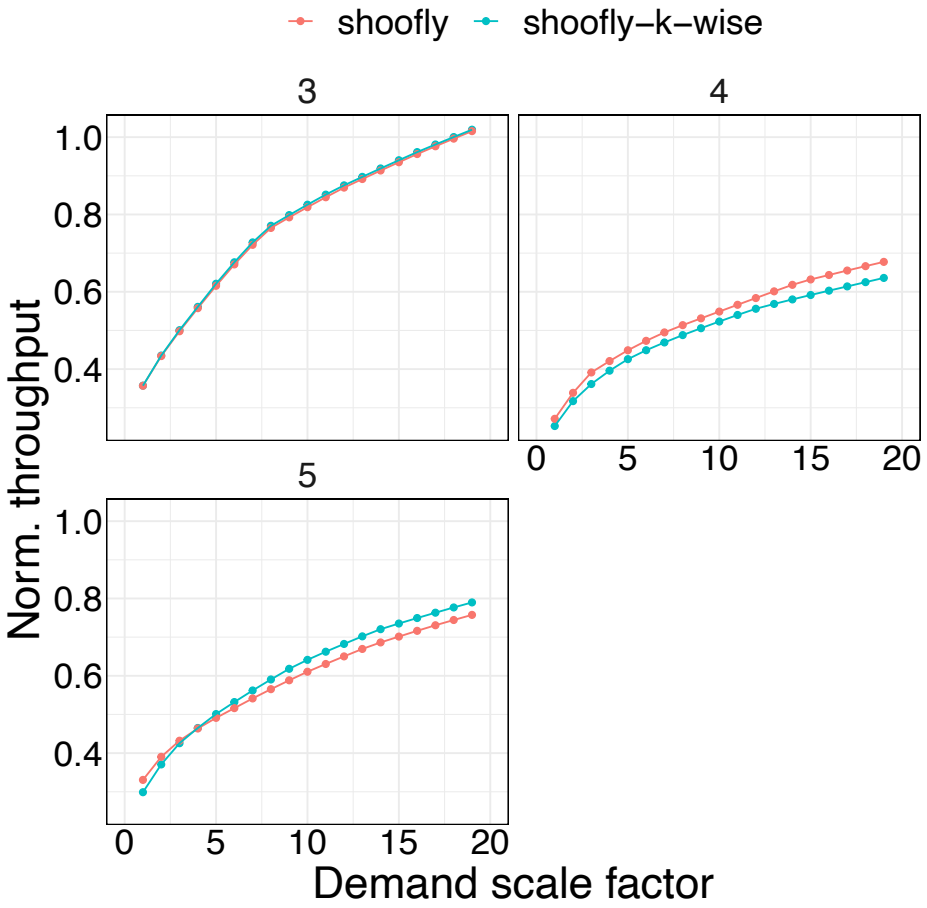


Figure 2: Norm. throughput of traffic engineering on 3, 4, and 5 hop shortcut topologies proposed by Shoofly.

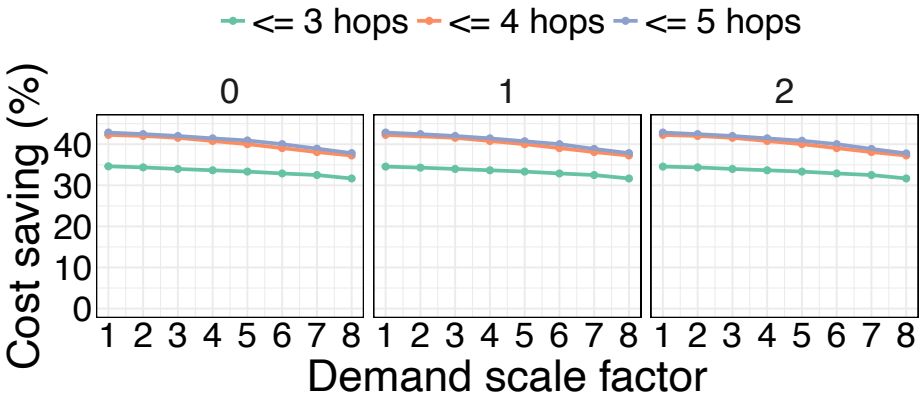


Figure 3: Cost savings by Shoofly in k = 0, 1 and 2 failure resilient scenarios. The cost savings are indistinguishable in all three cases showing that Shoofly can enable k-wise failure resilience without sacrificing cost savings.

A.1.4 Failure resilience vs. cost saving

In Section 3.4 we mentioned that k-wise link failure resilience can be incorporated in Shoofly without sacrificing cost savings. We demonstrate this in Figure 3 – cost savings are indistinguishable in case of k = 0, 1 and 2, showing that Shoofly can enable k-wise failure resilience without sacrificing cost savings even at highly scaled demands.

APPENDIX

CASCARA APPENDICES

A.1 Appendix

A.1.1 Speeding the MIP solution

In this section we briefly describe the intuitive 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 problem 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 far from optimal. In fact, the allocations obtained via this process did not show any significant improvement 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 the overall poor cost saving from the stitched traffic 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 autotune 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.1.2 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 c annot 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.1.3 Traffic allocation with Cascara

In this section we discuss details of the Cascara allocation algorithm which were omitted in Section 4.3 for brevity. The complete algorithm, Algorithm 5, expands on Algo-

rithm 4. 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 $\sum_k^m \text{CAPACITY}(l_i) \leq demand - C_f$.

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.

A.1.3.1 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. [1] 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 1 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.

[1] We have discussed the additional routing and client latency constraints on Cascara in §4.4.

Algorithm 5: 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 = $\frac{k}{100} * n$

$prio =$ freeslots ▷ Initial priority of all links

$linkq =$ PRIORITYQUEUE()

for *link* $\in L$ **do**

  linkq.insert(link, CAPACITY(link), freeslots, $prio$)

end

Function `allocate_timestep(`d, f`)`**:**

  link_alloc = {}

  augmented_links = []

  $C_f = f * C$ ▷ Fraction f of total capacity C

  if $d \leq C_f$ **then**

    link_alloc = bin_pack(L, C_f)

  else

    $d = d - C_f$

    while $d \geq 0$ **do**

      b_link = linkq.pop()

      if *!b_link* **then**

        return {}

      end

      augmented_links.add(b_link)

      if *b_link* $\in L_1$ **then**

        $d = d -$ (1 - f) $*$ CAPACITY(b_link)

      else

        $d = d -$ CAPACITY(b_link)

      end

      link_alloc [b_link]= CAPACITY(b_link)

    end

    for *link* $\in$ *augmented_links* **do**

      link.prio = link.prio -1

      link.free_slots = link.free_slots -1

    end

  end

  return link_alloc

End Function

while *not allocate_timestep*(d, f) **do**

  $f = f + \delta$

end

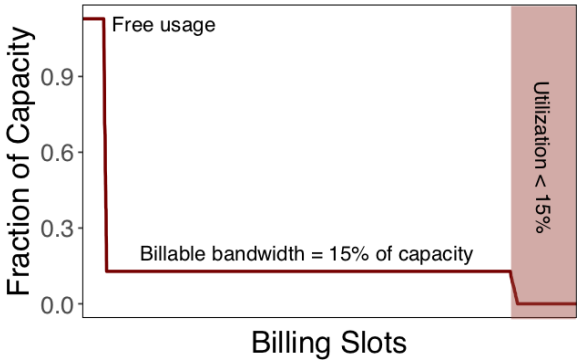


Figure 1: Utilization of a link as fraction of its capacity, sorted from high to low across billing slots in a month.

A.1.4 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 2 compares the cost savings per-month between our solutions from Algorithm 3 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* [50]. 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

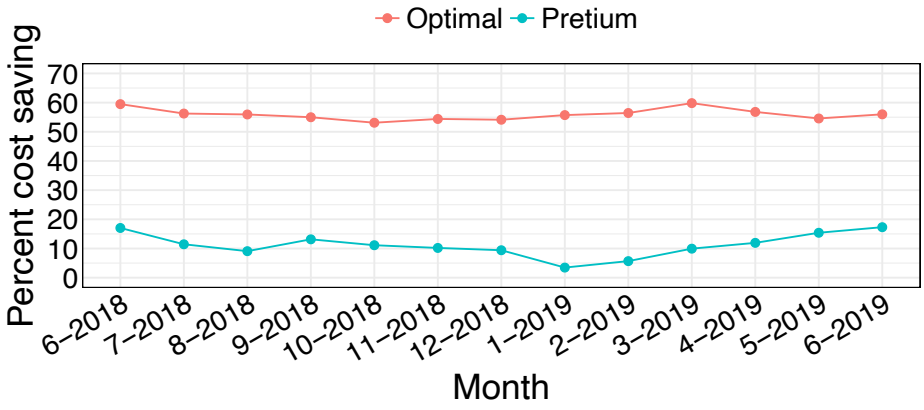


Figure 2: Cost saving by Cascara vs. Pretium [50] on a month-by-month basis.

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.

A.1.5 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 3).

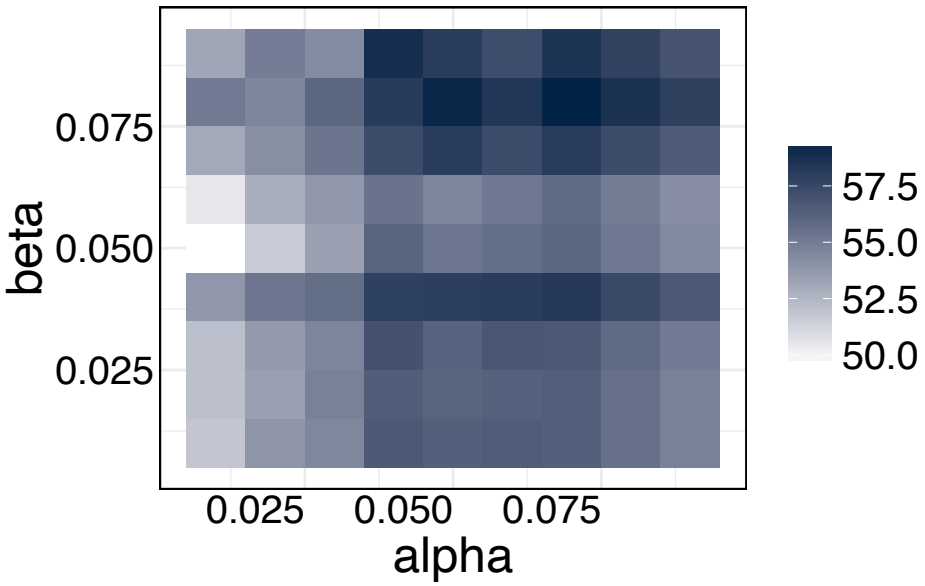


Figure 3: Average monthly bandwidth cost saving with Cascara as a function of the parameters α and β. We choose the best values of α and β for the evaluation in §4.4.

BIBLIOGRAPHY

[1] Acacia Communications. Acacia Bandwidth Variable Transceiver Module. http://ir.acacia-inc.com/phoenix.zhtml?c=254242&p=irol-newsArticle&ID=2103147, Mar. 2015.

[2] Ahuja, Ravindra K, Magnanti, Thomas L, and Orlin, James B. *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993.

[3] Arista Networks. Python client for Arista eAPI. https://github.com/arista-eosplus/pyeapi, Dec. 2017.

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

[5] Balasubramanian, Srivatsan, Ahuja, Satyajeet, Nagarajan, Gaya, Celletti, Andrea, and Foston, Frantisek. Multilayer planning for facebook scale worldwide network. In *2017 International Conference on Optical Network Design and Modeling (ONDM)* (2017), IEEE, pp. 1–6.

[6] Bangla, Ajay Kumar, Ghaffarkhah, Alireza, Preskill, Ben, Koley, Bikash, Albrecht, Christoph, Danna, Emilie, Jiang, Joe, and Zhao, Xiaoxue. Capacity planning for the google backbone network.

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

[8] Bicket, John C. Bit-rate selection in wireless networks. Master's thesis, Massachusetts Institute of Technology, 2005.

[9] Birk, M., Choudhury, G., Cortez, B., Goddard, A., Padi, N., Raghuram, A., Tse, K., Tse, S., Wallace, A., and Xi, K. Evolving to an sdn-enabled isp backbone: key technologies and applications. *IEEE Communications Magazine* (2016).

[10] Bjorklund, Martin. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, Oct. 2010.

[11] Bogle, Jeremy, Bhatia, Nikhil, Ghobadi, Manya, Menache, Ishai, Bjørner, Nikolaj, Valadarsky, Asaf, and Schapira, Michael. TEAVAR: striking the right utilization-availability balance in WAN traffic engineering. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019* (2019), ACM.

[12] Brzezinski, A., and Modiano, E. Dynamic reconfiguration and routing algorithms for ip-over-wdm networks with stochastic traffic. *Journal of Lightwave Technology 23*, 10 (2005), 3188–3205.

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

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

[15] Chang, Yiyang, Jiang, Chuan, Chandra, Ashish, Rao, Sanjay G., and Tawarmalani, Mohit. Lancet: Better network resilience by designing for pruned failure sets. In *Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems, Boston, MA, USA, June, 8-12, 2020* (2020), Edmund Yeh, Athina Markopoulou, and Y. C. Tay, Eds., ACM, pp. 53–54.

[16] Chang, Yiyang, Rao, Sanjay G., and Tawarmalani, Mohit. Robust validation of network designs under uncertain demands and failures. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017* (2017), Aditya Akella and Jon Howell, Eds., USENIX Association.

[17] Chiu, Angela L, Choudhury, Gagan, Clapp, George, Doverspike, Robert, Feuer, Mark, Gannett, Joel W, Jackel, Janet, Kim, Gi Tae, Klincewicz, John G, Kwon, Taek Jin, et al. Architectures and protocols for capacity efficient, highly dynamic and highly resilient core networks. *IEEE/OSA Journal of Optical Communications and Networking* (2011).

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

[19] Chou, David, Xu, Tianyin, Veeraraghavan, Kaushik, Newell, Andrew, Margulis, Sonia, Xiao, Lin, Ruiz, Pol Mauri, Meza, Justin, Ha, Kiryong, Padmanabha, Shruti, 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* (2019), pp. 430–446.

[20] Cisco. What is MPLS - Multiprotocol Label Switching. `https://www.cisco.com/c/en/us/products/ios-nx-os-software/multiprotocol-label-switching-mpls/index.html`, (Accessed on 2021-01-20).

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

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

[23] Diamond, Steven, and Boyd, Stephen P. CVXPY: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research 17* (2016), 83:1–83:5.

[24] 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).

[25] E. Rosen, Y. Rekhter. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, Feb. 2006.

[26] Filer, Mark, Gaudette, Jamie, Ghobadi, Monia, Mahajan, Ratul, Issenhuth, Tom, Klinkers, Buddy, and Cox, Jeff. Elastic optical networking in the microsoft cloud. *Journal of Optical Communications and Networking 8*, 7 (July 2016), A45–A54.

[27] Filer, Mark, Gaudette, Jamie, Yin, Yawei, Billor, Denizcan, Bakhtiari, Zahra, and Cox, Jeffrey L. Low-margin optical networking at cloud scale [invited]. *J. Opt. Commun. Netw. 11*, 10 (Oct 2019), C94–C108.

[28] Fischer, J. K., Alreesh, S., Elschner, R., Frey, F., Nˆlle, M., Schmidt-Langhorst, C., and Schubert, C. Bandwidth-variable transceivers based on four-dimensional modulation formats. *Journal of Lightwave Technology 32*, 16 (Aug 2014), 2886–2895.

[29] Fischer, J. K., Alreesh, S., Elschner, R., Frey, F., Nˆlle, M., and Schubert, C. Bandwidth-variable transceivers based on 4d modulation formats for future flexible networks. In *39th European Conference and Exhibition on Optical Communication (ECOC 2013)* (Sept 2013), pp. 1–3.

[30] Foerster, Klaus-Tycho, Schmid, Stefan, and Vissicchio, Stefano. Survey of consistent network updates. *CoRR abs/1609.02305* (Sept. 2016).

[31] Förster, Klaus-Tycho, Mahajan, Ratul, and Wattenhofer, Roger. Consistent updates in software defined networks: On dependencies, loop freedom, and blackholes. In *2016 IFIP Networking Conference, Networking 2016 and Workshops Vienna, Austria* (2016), IEEE Computer Society.

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

[33] Gass, Saul I., and Assad, Arjang A. *An Annotated Timeline of Operations Research: An Informal History*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[34] Ghobadi, Monia, Gaudette, Jamie, Mahajan, Ratul, Phanishayee, Amar, Klinkers, Buddy, and Kilper, Daniel. Evaluation of elastic modulation gains in Microsoft's optical backbone in North America. *Optical Fiber Communication Conference* (2016), M2J.2.

[35] Ghobadi, Monia, and Mahajan, Ratul. Optical layer failures in a large backbone. In *Internet Measurement Conference* (2016), ACM.

[36] Ghobadi, Monia, and Mahajan, Ratul. Optical layer failures in a large backbone. In *Proceedings of the 2016 Internet Measurement Conference* (2016), pp. 461–467.

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

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

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

[40] Gossels, Jennifer, Choudhury, Gagan, and Rexford, Jennifer. Robust network design for ip/optical backbones. *IEEE/OSA Journal of Optical Communications and Networking 11*, 8 (2019), 478–490.

[41] Govindan, Ramesh, Minei, Ina, Kallahalla, Mahesh, Koley, Bikash, and Vahdat, Amin. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *SIGCOMM Conference* (2016), ACM.

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

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

[44] Halperin, Daniel, Hu, Wenjun, Sheth, Anmol, and Wetherall, David. Predictable 802.11 packet delivery from wireless channel measurements. *SIGCOMM Comput. Commun. Rev. 41*, 4 (Aug. 2010), 12.

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

[46] Hong, Chi-Yao, Kandula, Srikanth, Mahajan, Ratul, Zhang, Ming, Gill, Vijay, Nanduri, Mohan, and Wattenhofer, Roger. Achieving high utilization with software-driven wan. *SIGCOMM Comput. Commun. Rev. 43*, 4 (Aug. 2013), 15–26.

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

[48] Infinera. Optical Line Systems. `https://www.infinera.com/optical-line-systems`, (Accessed on 2021-01-10).

[49] Jain, Sushant, Kumar, Alok, Mandal, Subhasree, Ong, Joon, Poutievski, Leon, Singh, Arjun, Venkata, Subbaiah, Wanderer, Jim, Zhou, Junlan, Zhu, Min, Zolla, Jon, Hölzle, Urs, Stuart, Stephen, and Vahdat, Amin. B4: Experience with a globally-deployed software defined wan. *SIGCOMM Comput. Commun. Rev. 43*, 4 (Aug. 2013), 3–14.

[50] Jalaparti, Virajith, Bliznets, Ivan, Kandula, Srikanth, Lucier, Brendan, and Menache, Ishai. Dynamic pricing and traffic engineering for timely inter-datacenter transfers. In *SIGCOMM'16*.

[51] Jia, Su, Jin, Xin, Ghasemiesfeh, Golnaz, Ding, Jiaxin, and Gao, Jie. Competitive analysis for online scheduling in software-defined optical WAN. In *INFOCOM* (2017), IEEE.

[52] Jin, Xin, Li, Yiran, Wei, Da, Li, Siming, Gao, Jie, Xu, Lei, Li, Guangzhi, Xu, Wei, and Rexford, Jennifer. Optimizing bulk transfers with software-defined optical WAN. In *SIGCOMM Conference* (2016), ACM.

[53] Jin, Xin, Li, Yiran, Wei, Da, Li, Siming, Gao, Jie, Xu, Lei, Li, Guangzhi, Xu, Wei, and Rexford, Jennifer. Optimizing bulk transfers with software-defined optical wan. In *Proceedings of the 2016 ACM SIGCOMM Conference* (2016), pp. 87–100.

[54] Jin, Xin, Liu, Hongqiang Harry, Gandhi, Rohan, Kandula, Srikanth, Mahajan, Ratul, Zhang, Ming, Rexford, Jennifer, and Wattenhofer, Roger. Dynamic scheduling of network updates. *SIGCOMM Comput. Commun. Rev. 44*, 4 (Aug. 2014), 539–550.

[55] Juniper Network. Shared Risk Link Groups for MPLS. `https://www.juniper.net/documentation/en_US/junos/topics/topic-map/srlg-for-mpls.html`, (Accessed on 2021-01-10).

[56] Kandula, Srikanth, Menache, Ishai, Schwartz, Roy, and Babbula, Spandana Raj. Calendaring for wide area networks. In *SIGCOMM'14*.

[57] Kumar, Praveen, Yuan, Yang, Yu, Chris, Foster, Nate, Kleinberg, Robert, Lapukhov, Petr, Lim, Chiun Lin, and Soulé, Robert. Semi-oblivious traffic engineering: The road not taken. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)* (2018).

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

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

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

[61] Liu, Hongqiang Harry, Kandula, Srikanth, Mahajan, Ratul, Zhang, Ming, and Gelernter, David. Traffic engineering with forward fault correction. *SIGCOMM Comput. Commun. Rev. 44*, 4 (Aug. 2014), 527–538.

[62] Liu, Hongqiang Harry, Kandula, Srikanth, Mahajan, Ratul, Zhang, Ming, and Gelernter, David. Traffic engineering with forward fault correction. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014* (2014), Fabián E. Bustamante, Y. Charlie Hu, Arvind Krishnamurthy, and Sylvia Ratnasamy, Eds., ACM, pp. 527–538.

[63] Liu, Hongqiang Harry, and Li, Jian. O(n) improve, (Accessed on 2020-12-19).

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

[65] Mahajan, Ratul, and Wattenhofer, Roger. On consistent updates in software defined networks. In *HotNets* (2013), ACM.

[66] Marian, T., Freedman, D.A., Birman, K., and Weatherspoon, H. Empirical characterization of uncongested optical lambda networks and 10gbe commodity endpoints. In *DSN* (2010).

[67] Mcgregor, Andrew, and Smithies, Derek. Rate adaptation for 802.11 wireless networks: Minstrel.

[68] Papanikolaou, P., Christodoulopoulos, K., and Varvarigos, E. Joint multi-layer survivability techniques for ip-over-elastic-optical- networks. *IEEE/OSA Journal of Optical Communications and Networking 9*, 1 (2017), A85–A98.

[69] Papanikolaou, P., Christodoulopoulos, K., and Varvarigos, E. Optimization techniques for incremental planning of multilayer elastic optical networks. *IEEE/OSA Journal of Optical Communications and Networking 10*, 3 (2018), 183–194.

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

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

[72] Schlinker, Brandon, Kim, Hyojeong, Cui, Timothy, Katz-Bassett, Ethan, Madhyastha, Harsha V, Cunha, Italo, Quinn, James, Hasan, Saif, Lapukhov, Petr, and Zeng, Hongyi. 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* (2017), ACM, pp. 418–431.

[73] Singh, Rachee, Agarwal, Sharad, Calder, Matt, and Bahl, Paramvir. Cost-effective cloud edge traffic engineering with cascara. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)* (Apr. 2021), USENIX Association, pp. 201–216.

[74] Singh, Rachee, Bjørner, Nikolaj, Shoham, Sharon, Yin, Yawei, Arnold, John, and Gaudette, Jamie. Cost-effective cloud edge traffic engineering with cascara. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (2021), SIGCOMM '21, Association for Computing Machinery.

[75] Singh, Rachee, Ghobadi, Manya, Foerster, Klaus-Tycho, Filer, Mark, and Gill, Phillipa. Radwan: Rate adaptive wide area network. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (New York, NY, USA, 2018), SIGCOMM '18, Association for Computing Machinery, p. 547–560.

[76] Singh, Rachee, Ghobadi, Monia, Foerster, Klaus-Tycho, Filer, Mark, and Gill, Phillipa. Run, walk, crawl: Towards dynamic link capacities. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (New York, NY, USA, 2017), HotNets-XVI, Association for Computing Machinery, p. 143–149.

[77] Spirent Communications. Spirent TestCenter. `https://www.spirent.com/Products/TestCenter`, Jan. 2018.

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

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

[80] TeleGeography. Wavelengths Pricing Data. https://www2.telegeography.com/wavelengths-pricing-data, (Accessed on 2020-01-20).

[81] Thomas Teipen, Brian, Eiselt, Michael, Grobe, Klaus, and Elbers, Jˆrg-Peter. Adaptive data rates for flexible transceivers in optical networks.

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

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

[84] Valadarsky, Asaf, Schapira, Michael, Shahaf, Dafna, and Tamar, Aviv. Learning to route. In *Proceedings of the 16th ACM workshop on hot topics in networks* (2017), pp. 185–191.

[85] Vutukuru, Mythili, Balakrishnan, Hari, and Jamieson, Kyle. Cross-layer wireless bit rate adaptation. *SIGCOMM Comput. Commun. Rev. 39*, 4 (Aug. 2009), 3–14.

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

[87] Yap, KK, Motiwala, Murtaza, Rahe, Jeremy, Padgett, Steve, Holliman, Matthew, Baldus, Gary, Hines, Marcus, Kim, TaeEun, Narayanan, Ashok, Jain, Ankur, Lin, Victor, Rice, Colin, Rogan, Brian, Singh, Arjun, Tanaka, Bert, Verma, Manish, Sood, Puneet, Tariq, Mukarram, Tierney, Matt, Trumic, Dzevad, Valancius, Vytautas, Ying, Calvin, Kallahalla, Mahesh, Koley, Bikash, and Vahdat, Amin. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *SIGCOMM'17* (2017).

[88] Yoshida, Y., Maruta, A., i. Kitayama, K., Nishihara, M., Tanaka, T., Takahara, T., Rasmussen, J. C., Yoshikane, N., Tsuritani, T., Morita, I., Yan, S., Shu, Y., Yan, Y., Nejabati, R., Zervas, G., Simeonidou, D., Vilalta, R., MuÒoz, R., Casellas, R., Martinez, R., Aguado, A., Lopez, V., and Marhuenda, J. Sdn-based network orchestration of variable-capacity optical packet switching network over programmable flexi-grid elastic optical path network. *Journal of Lightwave Technology 33*, 3 (Feb 2015), 609–617.

[89] Yoshida, Y., Maruta, A., Kitayama, K., Nishihara, M., Tanaka, T., Takahara, T., Rasmussen, J. C., Yoshikane, N., Tsuritani, T., Morita, I., Yan, S., Shu, Y., Channegowda, M., Yan, Y., Rofoee, B. R., Hugues-Salas, E., Saridis, G., Zervas, G., Nejabati, R., Simeonidou, D., Vilalta, R., MuÒoz, R., Casellas, R., Martinez, R., Svaluto, M., Fabrega, J. M., Aguado, A., Lopez, V., Marhuenda, J., de Dios, O. G., and Fernandez-Palacios, J. P. First international sdn-based network orchestration of variable-capacity ops over programmable flexi-grid eon. In *OFC 2014* (March 2014), no. Th5A.2, pp. 1–3.

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

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

[92] Zhuo, Danyang, Ghobadi, Monia, Mahajan, Ratul, Phanishayee, Amar, Zou, Xuan Kelvin, Guan, Hang, Krishnamurthy, Arvind, and Anderson, Thomas. RAIL: A case for redundant arrays of inexpensive links in data center networks. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (Mar. 2017), USENIX Association.