Joint Optimization for Robust Network Design and Operation

Jennifer Gossels

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Advisor: Professor Jennifer Rexford

April 2020

Abstract

The Internet is an essential part of modern life, and Internet Service Provider (ISP) backbone networks are integral to our Internet experience. Therefore, ISPs must build networks that limit congestion, even when some equipment fails. This network design problem is complicated, because an optimal network design must consider the eventual runtime configuration. An ISP makes *network design* decisions, such as purchasing and placing equipment, on long timescales (months or years) and *network operation* decisions, such as routing packets, on short timescales (seconds). Design and operation interact such that the ISP must solve the network operation problem as a sub-problem of network design, rendering the network design problem difficult to formulate and computationally complex.

Today ISPs resort to a variety of simplifications; they fail to take advantage of the reconfigurability offered by modern optical equipment or the opportunity to mix more- and less-powerful switches throughout their networks. In this dissertation, we show how ISPs can incorporate each of these factors into their network design and operation models to produce less expensive networks without compromising robustness.

In Chapter 2, we explain how reconfigurable optical switches fundamentally change the network design and operation problems by shifting the boundary between what is fixed at design time and what is reconfigured at runtime. Then, we present an optimal formulation for this new problem and heuristics to help our solution scale. In Chapter 3, we describe a failure recovery protocol that allows ISPs to realize many of the benefits of outfitting their networks with a homogeneous collection of powerful, "smart" switches, while instead using a combination of these expensive boxes and less expensive, "dumb" switches.

We make three contributions in each chapter. First, we formulate the network design optimization by extending the multicommodity flow framework to leverage colorless and directionless Reconfigurable Optical Add/Drop Multiplexers (Chapter 2) or heterogeneous nodes (Chapter 3). Second, we devise heuristics to scale our designs to larger topologies. Finally, we evaluate our ideas on a realistic backbone topology and traffic demands.

Acknowledgments

Princeton

Advisor	Jennifer Rexford
Thesis Committee	Gagan Choudhury
	Michael Freedman
	Kai Li
	Wyatt Lloyd

Chapter 2

AT&T	Gagan Choudhury
	Kathy Meier-Hellstern
	Simon Tse
Comments	Mina Tahmasbi Arashloo
	Manya Ghobadi
	Xin Jin
	Sanjay Rao

Chapter 3

Collaborator	Ronaldo A. Ferreira
Data	Gagan Choudhury/AT&T

Funding

National Science Foundation	Graduate Research Fellowship
	Grant CCF-1837030

Miscellaneous

Proofreading	Bonnie Gossels
	Jamie Gossels
Chapter Titles	Boston Red Sox
Everything	Family

Contents

List of Tables

List of Figures

Bibliographic Notes

A version of Chapter 2 appears in the August 2019 edition of the Journal of Optical Communications and Networking [27], co-authored with Gagan Choudhury and Jennifer Rexford. Chapter 3 is derived from joint work with Ronaldo A. Ferreira and Jennifer Rexford.

Chapter 1

Introduction

The Internet plays an integral role in nearly all aspects of modern life, and Internet Service Provider (ISP) backbone networks play a critical role in the functioning of the Internet. The Internet is a hierarchical "network of networks," and these ISP backbones sit at the top of the hierarchy (Figure 1.1). End systems such as laptops, cell phones, web servers, mail servers, and toasters connect to home, enterprise, or data center networks, which in turn connect to one or more *access ISP* networks. *Regional ISP* networks join together several of these access ISP networks, and regional ISPs in turn communicate with each other via *Tier 1 ISPs* or directly by connecting at *Internet exchange points (IXPs).* Tier 1 ISPs such as AT&T and Verizon maintain backbone networks that span an entire country and serve as the "glue" connecting thousands of these smaller networks.

Thus, given their position at the top of the hierarchy, Tier 1 ISP backbones carry traffic from millions of users each day. These users rely on these networks to provide reliable, rapid communication at all times — a surgeon simply can't afford for the Internet to "break" in the midst of a procedure, nor can a Red Sox fan streaming an important playoff game tolerate a disruption. Unfortunately for ISPs, the physical links and switches comprising their networks sometimes *do* fail. It is their responsibility to design and operate their networks such that these failures are largely transparent to their customers.

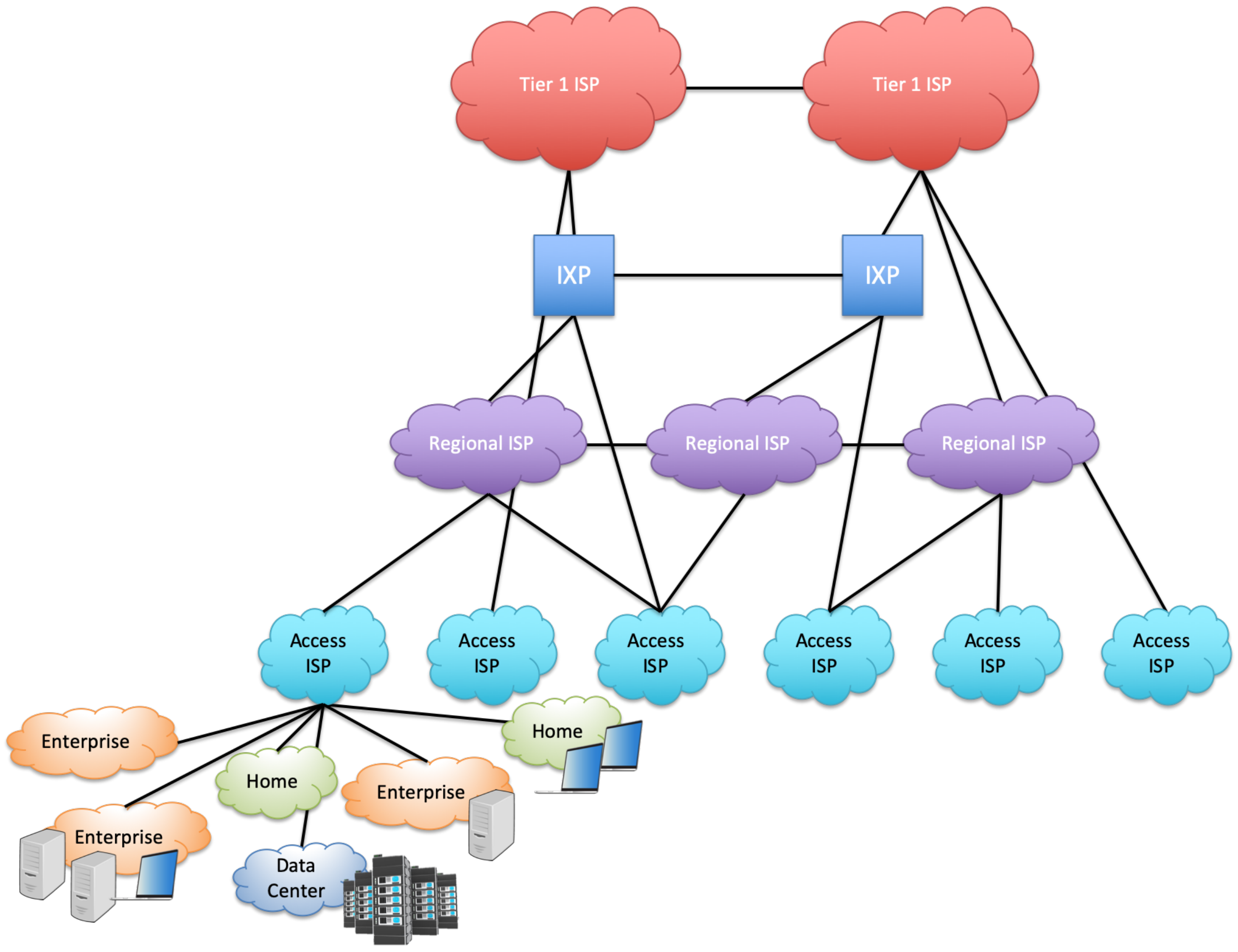


Figure 1.1: Hierarchical structure of the Internet. End systems connect to enterprise, home, or data center networks, which connect to access ISP networks. Multiple access ISP networks join together at each regional ISP network, which are themselves joined together by Tier 1 ISPs. Figure adapted from Figure 1.15 of [42].

In the next section, we explore the network design and operation problems in more detail, defining some key terminology and discussing how ISPs solve these problems today.

1.1 Network Design and Operation Today

An ISP makes some decisions, such as how much and what type of equipment to purchase and where to place this equipment, on large timescales on the order of months, and other decisions, such as exactly how to route packets from a given flow, on small timescales on the order of seconds. We say that these large-timescale decisions constitute *network design*,

Problem	**Inputs**	**Outputs**	**Constraints**	**Objective**
design	traffic matrix PoP/fiber locations failure scenarios	network topology	carry all traffic	min. equipment costs
operation	traffic matrix topology (failure scenarios)	routing of traffic over links	link bandwidths	min. congestion

Table 1.1: Inputs, outputs, constraints, and objectives of the network design and operation problems.

and these small-timescale decisions constitute *network operation*. For the ISP, both network design and operation require solving *constrained optimization* problems; the ISP must use *input* information about aspects of its network that it can't change to produce a set of *output* decisions, with the goal of optimizing a particular business *objective* and subject to certain real-world *constraints*. In the following subsections, we describe the inputs, outputs, constraints, and objectives of each, and Table 1.1 summarizes this discussion.

1.1.1 Network Design

A computer network is a graph whose nodes are switches and routers and whose edges are links. We use the terms switches and routers interchangeably, since the distinction between them is unimportant for our work. A network *topology* is the physical structure of this graph: the number, type, and locations of switches and how these switches are connected, including the bandwidth capacity of each link. Network design entails specifying a network topology, subject to the constraint that the chosen topology must be able to carry the offered traffic matrix. The *traffic matrix* is a specification of the (nonnegative but possibly zero) amount of traffic that each edge switch wants to send to each other edge switch. Of course, the ultimate sources and destinations of all packets are the end systems, but since we are concerned with the design and operation of ISP backbones, for our purposes these individual demands are aggregated at edge switches. Unlike in private backbones such as Google's B4 [35], an ISP has no control over the amount of traffic that each end host sends or in the timing of when packets enter the network. Further, packets must be delivered in a timely fashion; the ISP need not send them on the absolute shortest paths, but it can't use excessively circuitous routes, nor can it buffer packets at busy times and wait to send them when the network is less congested.

In reality, the network will be expected to carry a variety of traffic matrices, as traffic patterns change throughout the day and over the course of months and years. When we refer to the single traffic matrix, we assume that these demands represent the maximum demands the network will ever be responsible for; we assume that this traffic matrix "covers" all others. It is straightforward but beyond the scope of this thesis to extend our techniques to adapt to changing traffic matrices.

Hence, solving the network design problem amounts to answering the following questions:

- How many and what type of switches and routers should be placed at each potential location?

  For our purposes, choosing a set of potential locations for switches and routers is *not* part of network design; we assume that an ISP has already fixed the set of cities in which it wants to place points-of-presence (PoPs) and where in each city its PoP will be located. These PoPs are the potential locations for switches and routers, and network design involves placing switches at each PoP.

- How should these switches and routers be connected? What bandwidth should each link support?

  We do *not* choose where to lay fiber as part of network design; we assume that the ISP has already laid its fiber in the ground. Importantly, the amount of fiber laid between PoPs is not a limiting factor in the bandwidth of the links between said PoPs. Rather, the limiting factor is the number of interfaces located on switches at each PoP, along with the number of *optical regenerators* located along an IP link's path to ensure that its optical signal reaches far away destinations. Hence, network design involves determining what bandwidth IP link each stretch of fiber should support, but it does not allow for specifying additional paths for new fiber installations.

Robustness to Failures

A key part of the network design problem is ensuring that the topology will be robust to equipment failures. A fiber link can fail if, for example, it is accidentally cut during a construction project. Switches can fail if, for example, they lose power, are misconfigured, or encounter a hardware problem. Sometimes one or two of a switch's ports can fail while the rest remain operational. We use the term *failure scenario* to refer to any given simultaneous combination of specific switch or link failures.

In practice ISPs generally design their networks to be robust to certain classes of failure scenarios such as any single link failure, any single switch failure, any single link or switch failure, any combination of two link/switch failures, etc. Another important consideration is *shared link risk groups*. Failures are not randomly distributed throughout the network. Rather, if a natural disaster or construction accident destroys one IP link on the East Coast of the United States, it likely will affect nearby links, too. Hence, links that are geographically close together or are built upon common stretches of fiber are said to be part of a single shared link risk group. Often, an ISP designs its network to be robust to all the links in each shared link risk group failing simultaneously.

The need to account for failure scenarios greatly increases the complexity of the network design problem. Even without considering multiple failure scenarios, problem complexity grows as a function of the size of the network, because the solution must specify the locations for more switches and links. Failure scenarios exacerbate this problem, since the number of failure scenarios itself increases as the size of the network increases; the more nodes and links a network has, the more failure scenarios are included in e.g., the set consisting of all single node or link failures.

How Network Design is Done Today

A strawman approach to finding a feasible solution to the network design problem is to dramatically overprovision the network. If the ISP builds, e.g., two extra, redundant links for each link it actually needs, the network likely will not get too congested, even in the case of failures. However, buying the equipment to provision this network is costly for the ISP; while this idea gives a feasible solution to the constrained optimization problem, it does a poor job of achieving the ISP's objective of minimizing equipment costs.

A better approach is to formulate the network design problem as an integer linear program (ILP) whose solution represents a minimal cost placement of IP links, including which PoPs each link should connect and what each link's bandwidth capacity should be. The ISP can figure out where to place its IP switches and which optical paths its IP links should take based on the resulting output.

1.1.2 Network Operation

Solving the network operation problem amounts to deciding how the traffic matrix should be routed through the network topology. As in the network design problem, the traffic matrix is one input of the network operation problem. However, whereas the network design problem takes only the locations of PoPs and fiber spans as its second input, the network operation problem take as input the full network topology specified as the output of network design. The network operation problem may or may not take the set of failure scenarios as a third input. If this information is available and incorporated into the network operation formulation, then the network will be able to recover from failures faster than if a new routing must be computed in response to each failure. The constraints of the network operation problem are that no link can be assigned more traffic than it has the bandwidth capacity to carry. The objective of the network operation problem is typically to minimize congestion, though it can be to e.g., minimize or bound delay.

How Network Operation is Done Today

The job of switches in the network is to forward packets along their correct paths as fast as possible. Since the network operation problem is complex, "as fast as possible" does not leave time for computing a packet's next-hop on the fly, as each packet arrives. Instead, switches maintain *forwarding tables* to store the pre-computed solution to the network operation problem. This solution can be pre-computed in a variety of ways, for example in a distributed manner by the switches themselves or by a central controller which then installs forwarding rules on each switch. Regardless, these forwarding tables are populated dynamically as the network topology and traffic matrix change.

When a packet arrives, the switches *match* on various header fields, and their forwarding tables indicate the *action* the switch should take for any given match. For our purposes, the action specifies the next-hop switch to which the packet should be forwarded. This simple match-action computation can be done in hardware, enabling switches to forward packets at rates upwards of 100 gigabits per second (Gbps).

Today's networks generally employ one of two forwarding models, which differ according to which of the various header fields they use for matching: hop-by-hop forwarding or edge-to-edge tunneling. In hop-by-hop forwarding, all switches match on the packet's full header, or, in practice, the packet's destination IP address, and forward as appropriate. In contrast, with edge-to-edge tunneling, edge switches play a special role. The ingress edge switch matches on the packet's full header and determines the egress edge switch at which the packet should exit the backbone. It then labels the packet accordingly and sends the labeled packet into the network. Interior switches match and forward only on this label. In this model, edge switches are "smart" and maintain forwarding tables with enough entries for all destination IP prefixes in the Internet, while interior switches are "dumb" and have forwarding tables only large enough to store next-hop information for each egress switch in the network.

To protect against failures, ISPs generally configure their switches with backup paths. This approach, called fast reroute (FRR), can be used with hop-by-hop forwarding [51] or edge-to-edge tunneling [8,60]. In either case, switches' forwarding tables store precomputed next-hops for both the failure-free network topology and for each failure for which the ISP wants to prepare. FRR is generally effective at preventing disruption in the case of failures, as it reduces the time during which packets are lost to 50 milliseconds. Unfortunately, FRR imposes significant overhead, because the number of failure scenarios serves as a multiplier on the number of rules that each switch's forwarding table must store per packet. In addition, setting up FRR backup paths requires switches to engage in complex protocols.

1.1.3 Coupling of Network Design and Operation

Thus far, we have explained the job of an ISP as neatly divided into two components, network design and network operation. However, in reality the two are interconnected; the ISP cannot solve the network design problem without simultaneously also solving the network operation problem, and it cannot solve the network operation problem without fixing a network design. Figure 1.2 helps illustrate precisely why the two problems must be considered jointly. Network design decisions about where to place edge and core switches and optical regenerators constrain the possibilities for the network operation decisions of how to route traffic. Indeed, we saw in Section 1.1.2 that the network operation problem takes as input the topology produced as an output of the network design problem. From this perspective, we would think ISPs ought to solve the network design problem first.

However, to solve this network design problem, the ISP needs a sense of how traffic will be routed, because the specific routing determines how much capacity is needed in different regions. From this perspective, we would think that the ISP ought to solve the network operation problem first. Thus, we see that the ISP cannot solve either the network design problem or the network operation problem before the other and instead must solve them as one joint optimization.

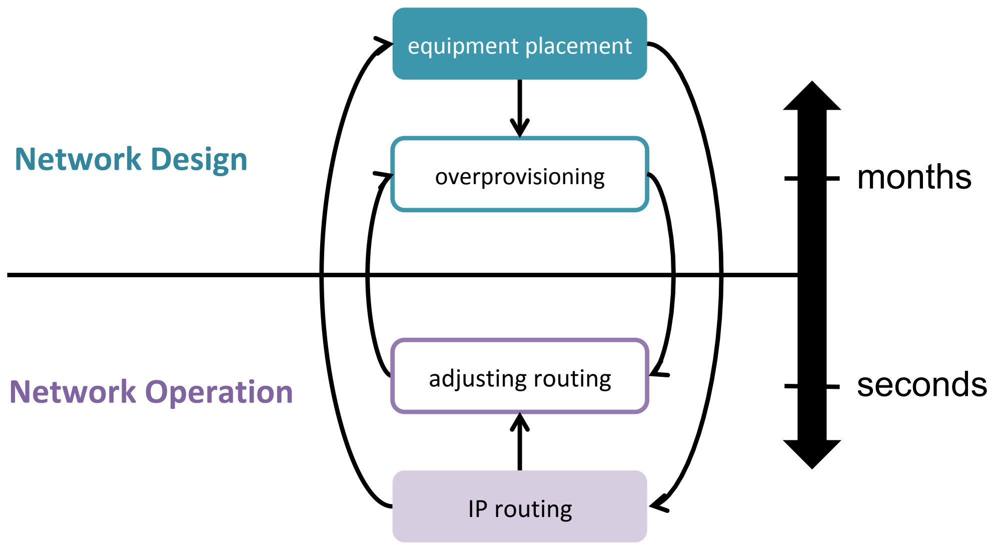


Figure 1.2: Illustration of how the network design and operation problems are interdependent. The vertical dimension is timescale.

The situation gets even more complicated if we consider the possibility of failures. On a large timescale, the ISP must provision its network such that if some components fail, there will still be enough capacity to carry all packets without too much congestion. On a small timescale, the ISP needs to immediately adjust routing to avoid dropping packets in flight. How much is enough capacity to provision depends on how the routing will be adjusted, but how the routing will be adjusted depends on how the network has been provisioned.

At a high level, what must happen is that the ISP solves the network operation problem as an “inner loop” within the network design computation; for each candidate network design, it determines the optimal routing for the topology remaining under each failure scenario. Of course, this is not what happens mathematically, as an algorithm that searches through every possible network design as its outer loop would be completely intractable. An important part of the challenge of generating an ideal network design and associated routing plan is to formulate the problems in a way that can be solved relatively efficiently.

1.2 Network Design and Operation Tomorrow

Thus far, we have defined the network design and operation problems and described how ISPs solve them today. In this section, we will explain how two factors, namely (i) emerging, reconfigurable optical technology and (ii) the realization that we can make use of the resources of "smart" interior switches that traditional routing models overlook, each present opportunities for us to improve upon these existing solutions; we can reformulate the network design and operation problems to better navigate the tradeoff between minimizing equipment costs and provisioning robust, minimally congested networks.

1.2.1 Reconfigurable Optics

Traditionally, the network design problem was to both place switches at link endpoints and fix how they should be connected with IP links. Link placement was part of network design by necessity; if an ISP wanted to modify either (i) the optical paths traversed by its IP links; or (ii) the IP-layer connectivity itself, then it needed to deploy workers to the relevant sites to manually make the changes. This lack of flexibility required ISPs to design their IP-layer links to be robust to failures, which forced them to install extra, expensive switches whose resources lay idle except in a few rare failure scenarios. However, this forced coupling of IP switches and their incident links did allow us to simplify the network design optimization problem, obviating the need to explicitly model the optical layer.

The optical layer is relevant for generating the IP topology in two ways:

- The length of the optical path underlying each IP link plays a significant role in determining the monetary cost of the IP link, because we need to place optical regenerators every 1000 miles to ensure that the optical signal remains strong.
- It helps determine which links are part of a single shared link risk group.

When switches were permanently tied to a single IP link and each IP link was permanently tied to a single optical path, we could always minimize cost by assigning each IP link to its shortest possible optical path. Hence, we could calculate the cost of a potential IP link without explicitly modeling every hop of its optical path; we only needed its total length. We didn't need to incorporate the optical topology into the optimization problem for determining shared link risk groups, either, because we could precompute the set of failure scenarios.

Now, new optical technology allows both (i) establishing new IP links; and (ii) changing the optical paths of existing IP links on the fly, as long as sufficient switches have been placed at the PoPs and sufficient regenerators have been placed along the optical path. Therefore, this technology provides the opportunity to shift IP link placement from network design into network operation (Figure 1.3). But, switch placement remains part of network design, because we still need to deploy workers to each PoP to install new endpoints.

Unfortunately, this increased flexibility comes at the cost of complicating the network design problem. Since we can now reuse regenerators and switches across failure scenarios, we can no longer assume that the least expensive design sends every IP link over its shortest optical path; it might be less expensive to use slightly longer paths for some links in some failure scenarios if those paths allow for reusing regenerators and switches across failure scenarios. For this reason, we must explicitly model the optical topology in our network design formulation.

1.2.2 Heterogeneous Nodes

As we mention above (Section 1.1.2), existing routing models generally fall into one of two categories: hop-by-hop forwarding or edge-to-edge tunneling. Each of these network operation paradigms is closely tied to a particular network design (Figure 1.4). Because hop-by-hop forwarding requires all nodes to perform equally complex matches and store equally massive forwarding tables, this approach is generally used when all switches in the network have similar processing power and memory. We call this the *all-nodes-equal* network model. In contrast, because edge-to-edge tunneling requires edge switches to match on a more complex set of header fields and store larger forwarding tables than interior nodes,

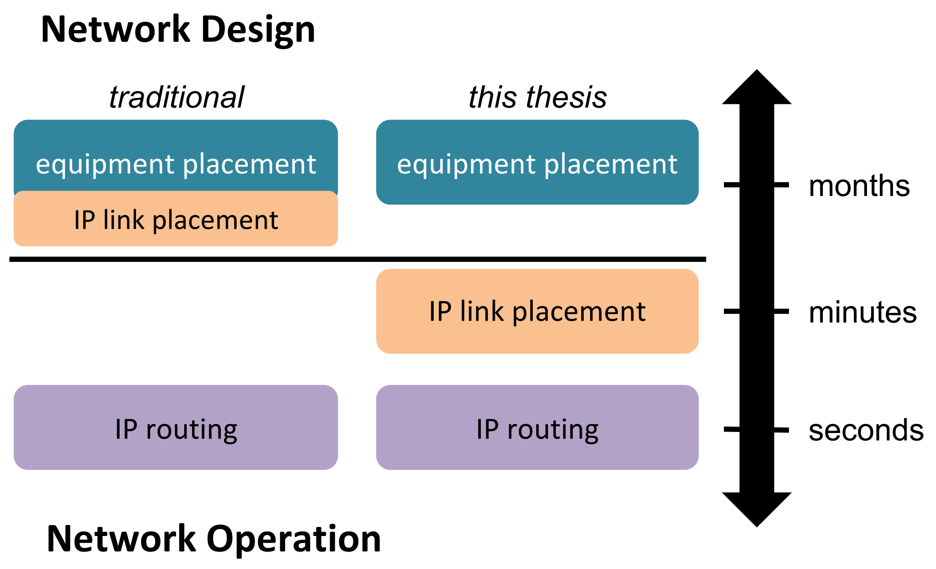


Figure 1.3: Components of network design vs. network operation in both traditional (left) and modern (right) networks.

this approach is generally used when edge switches have more processing power and memory than their interior counterparts. We call this the *smart-edge-dumb-interior* network model.

This traditional approach makes some sense; hop-by-hop forwarding *is* well suited to a network in which all nodes are equal, and edge-to-edge tunneling *is* well suited to a network in which all edge nodes are "smart" and all interior nodes are "dumb." However, we argue that confining ourselves to these two extreme design points is missing an opportunity to achieve the best of both worlds with a hybrid architecture. We propose that the ISP upgrade a select few interior switches to make them "smart" enough to play the role of traditional edge switches. Then, the ISP can route packets by tunneling them between these "smart" *big nodes*, rather than from edge-to-edge. This hybrid approach is superior to hop-by-hop routing from a cost perspective, because it doesn't require *all* interior nodes to be "smart." In addition, it is superior to edge-to-edge tunneling, because it allows the network to recover more rapidly in the event of a switch or link failure.

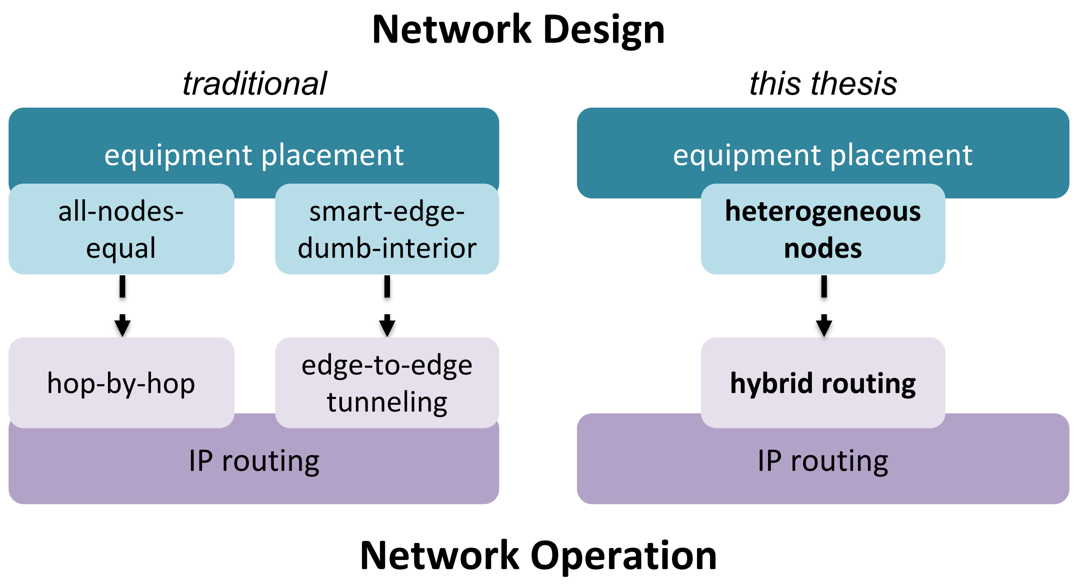


Figure 1.4: Traditionally, networks consist of either homogeneous switches throughout or "smart" switches at the edge and "dumb" switches in the interior. In the former case, ISPs generally employ hop-by-hop routing, and in the latter case they use edge-to-edge tunneling. We propose an intermediate design, in which heterogeneous nodes are mixed throughout the network, and a corresponding routing model in which packets are tunneled from "smart" big node to "smart" big node.

1.3 Contributions

We have now explained that reconfigurable optics and heterogeneous nodes allow for creating less expensive, more robust networks than were previously possible. However, to generate these better network designs and routing models, we must reformulate the traditional design and operation optimization problems. The existing optimization problems each operate on one type of entity: network design takes place entirely at the IP layer, and network operation assumes either homogeneous switches throughout the network or an edge made up entirely of "smart" switches and an interior composed entirely of "dumb" switches. In contrast, our two new network design formulations require *jointly optimizing* across the IP and optical layers and across various types of switches mixed throughout the network, respectively.

In both cases, formulating the problems in this more complex way is necessary *because ISPs need their networks to be robust to failures.* If switches and links could never fail, we would have no need to reconnect IP links during network operation (assuming no drastic changes in the traffic matrix), and we wouldn't derive any benefit from explicitly modeling both the IP and optical layers. Likewise, in the absence of failures, edge-to-edge tunneling is essentially as good at limiting congestion as our big node-to-big node tunneling model.

Put simply, emerging optical technology and heterogeneous switches give us the ability to reformulate the traditional design and operation problems. The need to create networks that are robust to failures gives us the motivation to do so.

Concretely, we make the following three contributions:

1. We adapt the common multicommodity flow optimization formulation in two different ways, one which takes advantage of reconfigurable optics (Chapter 2) and one which takes advantage of heterogeneous nodes (Chapter 3).

2. For each optimization formulation, we also present heuristics that scale better to larger topologies.

3. We evaluate our formulations and heuristics in a realistic backbone setting. In Chapter 2, our algorithms reduce equipment costs by up to 29%. In Chapter 3 our failure recovery algorithm allows ISPs to achieve most of the benefits of fully outfitting their networks with expensive switches while actually upgrading only about 20% of interior nodes.

1.4 Summary

In Chapter 2 we describe how we take advantage of reconfigurable optical technology, a project which we call SOX: *S*trategic *O*ptical/IP *X*-Layer Network Design. In Chapter 3 we describe how we take advantage of heterogeneous nodes, a project which we call Red: A Communist Approach to Network Operation. We conclude in Chapter 4.

Chapter 2

Sox: *S*trategic *O*ptical/IP *X*-Layer Network Design

2.1 Introduction

Over the past several years, improvements in optical switching technology, along with advances in software control, have given network operators more flexibility in configuring their in-ground optical fiber into an IP network. Whereas traditionally, at network design time, each IP link was assigned a fixed optical path and bandwidth, modern remote software controllers can program colorless and directionless Reconfigurable Optical Add/Drop Multiplexers (CD ROADMs) to remap the IP topology to the optical underlay on the fly, while the network continues carrying traffic and without deploying technicians to remote sites (Figure 2.1) [12, 20, 21, 55].

In the traditional setting, if a router failure or fiber cut causes an IP link to go down, all resources that were being used for said IP link are rendered useless. There are two viable strategies to recover from any single optical span or IP router failure. First, we could independently restore the optical and IP layers, depending on the specific failure; we could perform pure optical recovery in the case of an optical span failure or pure IP recovery in the case of an IP router failure. Note that the strategy we refer to as "pure optical recovery," of

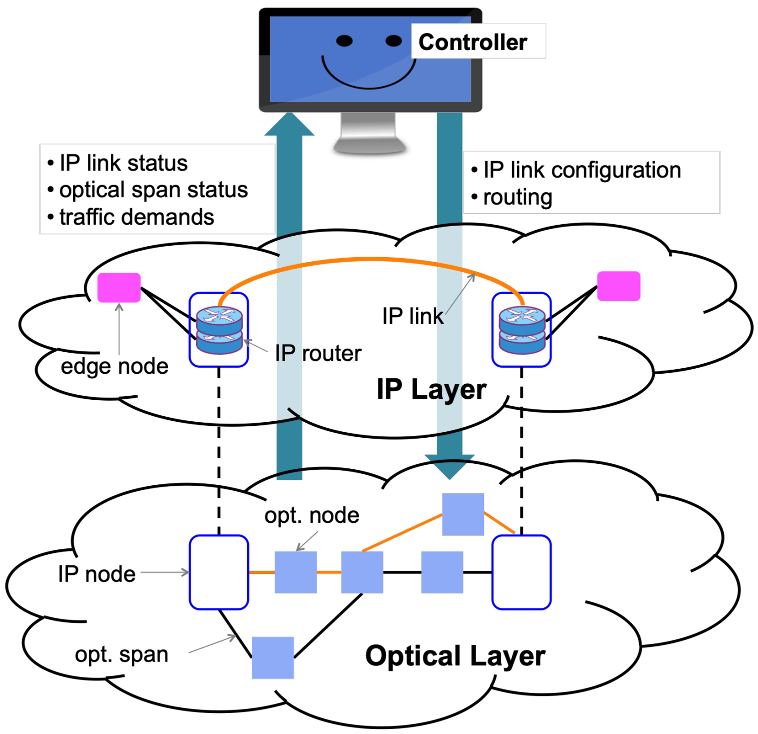


Figure 2.1: Layered IP/optical architecture. The highlighted orange optical spans comprise one possible mapping of the orange IP link to the optical layer. Alternatively, the controller could remap the same orange IP link to follow the black optical path.

course, involves reestablishing the IP link over the new optical path. We call it "pure optical recovery" because once the link has been recreated over the new optical path, the change is transparent to the IP layer. Second, we could design the network with sufficient capacity and path diversity that we can perform pure IP restoration at runtime. In practice, ISPs have used the latter strategy, as it is generally more resource efficient [18].

Now, the optical and electrical equipment can be repurposed for setting up the same IP link along a different path, or even for setting up a different IP link. In the context of failure recovery, the important upshot is that joint multilayer (IP and optical) failure recovery is now possible at runtime. The controller is responsible for performing this remote reprogramming of both CD ROADMs and routers.

Thus, programmable CD ROADMs shift the boundary between network design and network operation (Figure 2.2). We use the term *network design* to refer to any changes that happen on a human timescale, e.g., installing new routers or dispatching a crew to fix a failed

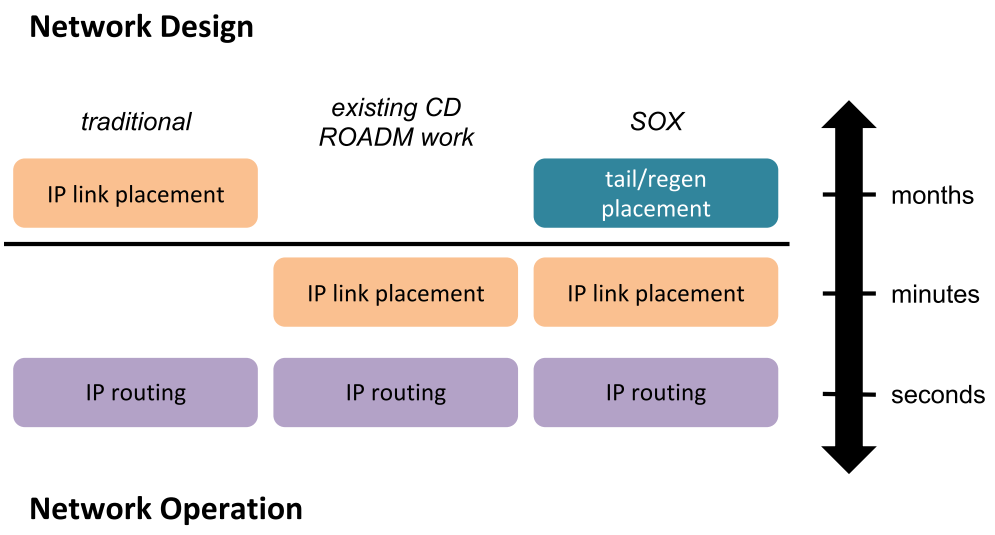


Figure 2.2: Components of network design vs. network operation in, from left to right: traditional networks, existing studies on how best to take advantage of CD ROADMs, and Sox. The vertical dimension is timescale. Cf. Figure 1.3.

link. We use *network operation* to refer to changes that can happen on a smaller timescale, e.g., adjusting routing in response to switch or link failures or changing demands.

As Figure 2.2 shows, network design *used to* comprise IP link placement. To describe what it now entails, we must provide background on IP/optical backbone architecture (Figure 2.3). The limiting resources in the design of an IP backbone are the equipment housed at each IP and optical-only node. Specifically, an IP node's responsibility is to terminate optical links and convert the optical signal to an electrical signal, and to do so it needs enough *tails* (*tail* is shorthand for the combination of an optical transponder and a router port). An optical node must maintain the optical signal over long distances, and to do so it needs enough *regenerators* or *regens* for the IP links passing through it. Therefore, we precisely state the new network design problem as follows: *Place tails and regens in a manner that minimizes cost while allowing the network to carry all expected traffic, even in the presence of equipment failures.*

This new paradigm creates both opportunities and challenges in the design and operation of backbone networks [17]. Previous work has explored the advantages of joint multilayer optimization over traditional IP-only optimization [12,20,21,55] (e.g., see Table 1 of [21]). However, these authors primarily resorted to heuristic optimization and restoration algorithms, due to the restrictions of routing (avoiding splitting flows into arbitrary proportions), the need for different restoration and latency guarantees for different quality-of-service classes, and the desirability of fast run times.

Further complicating matters is that network components fail, and when they do a production backbone must reestablish connectivity within seconds. Tails and regens cannot be purchased or relocated at this timescale, and therefore our network design must be *robust* to a set of possible failure scenarios. Importantly, we consider as *failure scenarios* any single optical fiber cut or IP router failure. There are other possible causes of failure (e.g., single IP router port, ROADM, transponder, power failure), which allow for various alternative recovery techniques, but we focus on these two.

Thus, we overcome three main challenges to present an exact formulation and solution to the network design problem.

1. The solution must be a single tail and regen configuration that works for all single IP router and optical fiber failures. This configuration should minimize cost under the assumption that the IP link topology will be reconfigured in response to each failure.

2. The positions of regens relative to each other along the optical path determine which IP links are possible.

3. The problem is computationally complex because it requires integer variables and constraints. Each tail and each regen supports a 100 Gbps IP link. Multiple tails or multiple regens can be combined at a single location to build a faster link, but they can't be split into e.g., 25 Gbps units that cost 25% of a full element.

These challenges arise because the recent shift in the boundary between network design and operation fundamentally changes the design problem; simply including link placement in network operation optimizations does not suffice to fully take advantage of CD ROADMs.

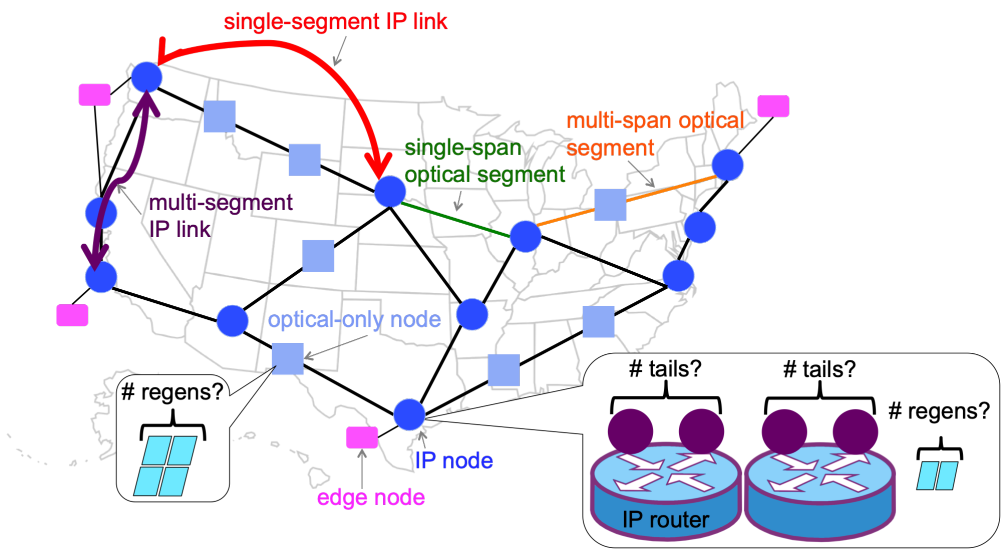


Figure 2.3: IP/optical network terminology.

A network design is optimal relative to a certain set of assumptions about what can be reconfigured at runtime. Hence, traditional network designs are only optimal *under the assumption that tails and regens are fixed to their assigned IP links.* With CD ROADMs, the optimal network design must be computed *under the assumption that IP links will be adjusted* in response to failures or changing traffic demands.

To this end, we make three main contributions.

1. After describing the importance of jointly optimizing over the IP and optical layers in Section 2.2, we formulate the optimal network design algorithm (Section 2.3). In this way we address challenges (1) and (2) from above.

2. We present two scalable, time-efficient approximation algorithms for the network design problem, addressing the computational complexity introduced by the integer constraints (Section 2.4), and we explain which use cases are best suited to each of our algorithms (Section 2.4.3).

3. We evaluate our three algorithms in relation to each other and to legacy networks (Section 2.5).

We discuss related work in Section 2.6 and conclude in Section 2.7.

2.2 IP/Optical Failure Recovery

In this section we provide more background on IP/optical networks. We begin by defining key terms and introducing a running example (Section 2.2.1). We then use this example to discuss various failure recovery options in both traditional (Section 2.2.2) and CD ROADM (Section 2.2.3) IP/optical networks.

2.2.1 IP/Optical Network Architecture

As shown in Figure 2.3, an IP/optical network consists of optical fiber, the IP nodes where fibers meet, the optical nodes stationed intermittently along fiber segments, and the edge nodes that serve as the sources and destinations of traffic. We do not consider the links connecting an edge router to a core IP router as part of our design problem; we assume these are already placed and fault tolerant.

Each IP node houses one or more IP *routers*, each with zero or more tails, and zero or more optical regens. The optical regens at an IP node are only used for IP links that pass through that node without terminating at any of its routers. Each optical-only node houses zero or more optical regens but cannot contain any routers (Figure 2.3). While IP and optical nodes serve as the endpoints of optical spans and segments, specific IP routers serve as the endpoints of IP links.

We say that an *optical span* is the smallest unit describing a stretch of optical fiber; an optical span is the section of fiber between any two nodes, be they IP or optical-only. Optical-only nodes can join multiple optical spans into a single *optical segment*, which is a stretch of fiber terminated at both ends by IP nodes. The path of a single optical segment may contain one or more optical-only nodes. The physical layer underlying each *IP link* comprises one or more optical segments. An IP link is terminated at each end by a specific IP router and can travel over multiple optical segments if its path traverses an intermediate IP node without terminating at one of that node's routers. Figure 2.3 illustrates the roles of optical spans and segments and IP links. The locations of all nodes and optical spans are fixed and cannot be changed, either at design time or during network operation.

An optical signal can travel only a finite distance along the fiber before it must be regenerated; every REGEN_DIST units the optical signal must pass through a regen, where it is converted from an optical signal to an electrical signal and then back to optical before being sent out the other end. The exact value of REGEN_DIST varies depending on the specific optical components, but it is roughly 1000 miles for our setting of a long-distance ISP backbone with 100 Gbps technology. We use the value of REGEN_DIST = 1000 miles throughout this thesis.

Example network design problem. The network in Figure 2.4 has two IP nodes, $\mathcal{I}1$ and $\mathcal{I}2$, and five optical-only nodes, O1-O5. $\mathcal{I}1$ and $\mathcal{I}2$ each have two IP routers (I1, I2 and I3, I4, respectively). Edge routers E1 and E2 are the sources and destinations of all traffic. The problem is to design the optimal IP network, requiring the fewest tails and regens, to carry 80 Gbps from E1 to E2 while surviving any single optical span or IP router failure. We do not consider failures of E1 or E2, because failing the source or destination would render the problem trivial or impossible, respectively.

If we don't need to be robust to any failures, the optimal solution is to add one 100 Gbps IP link from I1 to I3 over the nodes $\mathcal{I}1$, O1, O2, O3, and $\mathcal{I}2$. This solution requires one tail each at I1 and I3 and one regen at O2, for a total of two tails and one regen.

2.2.2 Failure Recovery in Traditional Networks

In traditional networks, the design problem is to place IP links; in this setting, once an IP link is placed at design time, its tails and regens are permanently committed to it. If one optical span or router fails, the entire IP link fails and the rest of its resources lie idle. During network operation, we may only adjust routing over the established IP links.

In general, this setup allows for four possible types of failure restoration. Two of these techniques are inadequate because they cannot recover from all relevant failure scenarios

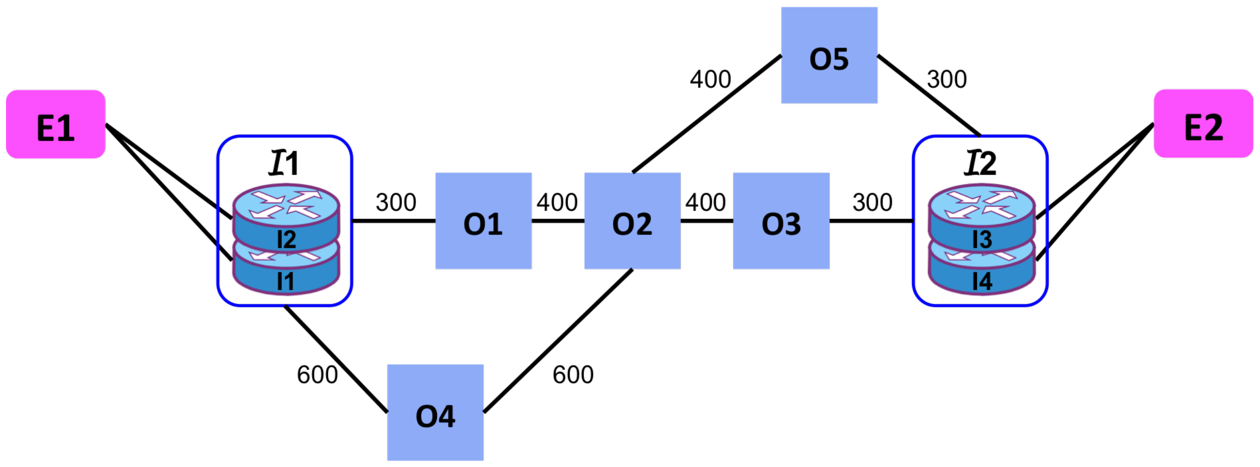


Figure 2.4: Example optical network illustrating the different options for failure restoration. The number near each edge is the edge's length in miles.

Recovery Technique	**# Tails**	**# Regens**	**IP?**	**Optical?**
pure optical	2	2	✗	✔
pure IP, shortest path	4	4	✔	✗
pure IP, any path	4	3	✔	✔
separate IP and optical	4	4	✔	✔
joint IP/optical	**4**	**2**	✔	✔

Table 2.1: Properties of various failure recovery approaches. The first four techniques are possible in legacy and CD ROADM networks, while the fifth requires CD ROADMs.

(first two rows of Table 2.1). The other two are effective but suboptimal in their resource requirements (second two rows of Table 2.1). We describe these four approaches below, guided by the running example shown in Figure 2.4. In Section 2.2.3 we show that CD ROADMs allow for a network design that meets our problem's requirements in a more cost-effective way.

Inadequate recovery techniques. In *pure optical layer* restoration, if an optical span fails, we reroute each affected IP link over the optical network by avoiding the failed span. The rerouted path may require additional regens. In the example shown in Figure 2.4, this amounts to rerouting the IP link along the alternate path $\mathcal{I}1$-O4-O2-O5-$\mathcal{I}2$ whenever any optical span fails. This path requires one regen each at O4 and O2. However, because the ($\mathcal{I}1$, $\mathcal{I}2$) link will never be instantiated over both paths simultaneously, the second path can reuse the original regen O2. Hence, we need only buy one extra regen at O4, for a total of two tails (at $\mathcal{I}1$ and $\mathcal{I}2$) and two regens (at O2 and O4). The problem with this pure optical restoration strategy is that it cannot protect against IP router failures.

In *pure IP layer restoration with each IP link routed along its shortest optical path*, we maintain enough fixed IP links such that during any failure condition, the surviving IP links can carry the required traffic. If any component of an IP link fails, then the entire IP link fails and even the intact components cannot be used. In large networks, this policy usually finds a feasible solution to protect against any single router or optical span failure. However, it may not be optimally cost-effective due to the restriction that IP links follow the shortest optical paths. Furthermore, in small networks it may not provide a solution that is robust to all optical span failures.

If we only care about IP layer failures, the optimal strategy for our running example is to place two 100 Gbps links, one from I1 to I3 and a second from I2 to I4 and both following the optical path $\mathcal{I}1$-O1-O2-O3-$\mathcal{I}2$. Though this design is robust to the failure of any one of I1, I2, I3, and I4, it cannot protect against optical span failures.

Correct but suboptimal recovery techniques. In contrast to the two failure recovery mechanisms described above, the following two techniques can correctly recover from any single IP router or optical span failure. However, neither reliably produces the least expensive network design.

Pure IP layer restoration with no restriction on how IP links are routed over the optical network is the same as IP restoration over shortest paths except IP links can be routed over any optical path. With this policy, we always find a feasible solution for all failure conditions, and it finds the most cost-effective among the possible pure-IP solutions. However, its solutions still require more tails or regens than those produced by our ILP, and solving for this case is computationally complex. In terms of Figure 2.4, pure IP restoration with no restriction on IP links' optical paths entails routing the (I1, I3) IP link along the $\mathcal{I}1$-O1-O2-O3-$\mathcal{I}2$ path and the (I2, I4) IP link along the $\mathcal{I}1$-O4-O2-O5-$\mathcal{I}2$ path. This requires two tails plus one regen (at O2) for the first IP link and two tails plus two regens (at O4 and O2) for the second IP link, for a total of four tails and three regens.

The final failure recovery technique possible in legacy networks, without CD ROADMs, is *pure IP layer restoration for router failures and pure optical layer restoration for optical failures.* This policy works in all cases but is usually more expensive than the two pure IP layer restorations mentioned above. In terms of our running example, we need two tails and two regens for each of two IP links, as we showed in our discussion of pure IP recovery along shortest paths. Hence, this strategy requires a total of four tails and four regens.

In summary, the optimal network design with legacy technology that is robust to optical and IP failures requires four tails and three regens.

2.2.3 Failure Recovery in CD ROADM Networks

A modern IP/optical network architecture is identical to that described in Section 2.2.1 aside from the presence of a remote controller. This single logical controller receives notifications of the changing status of any IP or optical component and also any changes in traffic demands between any pair of edge routers and uses this information compute the optimal IP link configuration and the optimal routing of traffic over these links. It then communicates the relevant link configuration instructions to the CD ROADMs and the relevant forwarding table changes to the IP routers.

As in the traditional setting, we cannot add or remove edge nodes, IP nodes, optical-only nodes, or optical fiber. But, now the design problem is to decide how many tails to place on each router and how many regens to place at each IP and optical node; no longer must we commit to fixed IP links at design time. Routing remains a key component of the network design problem, though it is now joined by IP link placement.

Any of the four existing failure recovery techniques is possible in a modern network. In addition, the presence of software-controlled CD ROADMs allows for a fifth option, joint IP/optical recovery. In contrast to the traditional setting, IP links can now be reconfigured at runtime. As above, suppose the design calls for an IP link between routers $\mathcal{I}1$ and $\mathcal{I}2$ over the optical path I1-O1-O2-O3-I4. Now, these resources are *not* permanently committed to this IP link. If one component fails, the remaining tails and regens can be repurposed either to reroute the ($\mathcal{I}1$, $\mathcal{I}2$) link over a different optical path or to (help) establish an entirely new IP link.

Returning to our running example, with joint IP/optical restoration, we can recover from any single IP or optical failure with just one IP link from I1 to I3. If there is any optical link failure then this link shifts from its original shortest path, which needs a regen at O2, to the path $\mathcal{I}1$-O4-O2-O5-$\mathcal{I}2$, which needs regens at O2 and O4. Importantly, the regen at O2 can be reused. Hence, thus far we need two tails and two regens. To account for the possibility of I1 failing, we add an extra tail at I2; if I1 fails then at runtime we create an IP link from I2 to I3 over the path $\mathcal{I}1$-O1-O2-O3-$\mathcal{I}2$. Since this link is only active in the case that I1 has failed, it will never be instantiated at the same time as the (I1, I3) link and can therefore reuse the regen we already placed at O2. Finally, to account for the possibility of I3 failing, we add an extra tail at I4. This way, at runtime we can create the IP link (I1, I4) along the path $\mathcal{I}1$-O1-O2-O3-$\mathcal{I}2$. Again, only one of these IP links will ever be active at one time, so we can reuse the regen at O2. Therefore, our final joint optimization design requires four tails and two regens. Hence, even in this simple topology, compared to the most cost efficient traditional strategy, joint IP/optical optimization and failure recovery saves the cost of one regen.

A Note on Transient Disruptions

As shown in Figure 2.2, IP link configuration operates on the order of minutes, while routing operates on sub-second timescales. IP link configuration takes several minutes because the process entails the following three steps:

1. Adding or dropping certain wavelengths at certain ROADMs;

2. Waiting for the network to return to a stable state; and

3. Ensuring that the network is indeed stable.

A “stable state” is one in which the optical signal reaches tails at IP link endpoints with sufficient optical power to be correctly converted back into an electrical signal. Adding or dropping wavelengths at ROADMs temporarily reduces the signal’s power enough to interfere with this optical-electrical conversion, thereby rendering the network temporarily unstable. Usually, the network correctly returns to a stable state within seconds of reprogramming the wavelengths (i.e., steps (1) and (2) finish within seconds). However, to ensure that the network is always operating with a stable physical layer (step (3)), manufacturers add a series of tests and adjustments to the reconfiguration procedure. These tests take several minutes, and therefore step (3) delays completion of the entire process. Researchers are working to bring reconfiguration latency down to the order of milliseconds [19], similar to the timescale at which routing currently operates. However, for now we must account for a transition period of approximately two minutes when the link configuration has not yet been updated and is therefore not optimal for the new failure scenario.

During this transient period, the network may not be able to deliver all the offered traffic. We mitigate this harmful traffic loss by immediately reoptimizing routing over the existing topology while the network is transitioning to its new configuration. As we show in Section 2.5.4, by doing so we successfully deliver the vast majority of offered traffic under almost all failure scenarios. Many operational ISPs carry multiple classes of traffic, and their service level agreements (SLAs) allow them to drop some low priority traffic under failure or extreme congestion. At one large ISP, approximately 40-60% of traffic is low priority. We always deliver at least 50% of traffic just by rerouting.

2.3 Network Design Problem

We now describe the variables and constraints of our ILP for solving the network design problem. After formally stating the objective function in Section 2.3.1 we introduce the problem’s constraints in 2.3.2 and 2.3.3. To avoid cluttering our presentation of the main ideas of the model, throughout 2.3.1 - 2.3.3 we assume exactly one router per IP node. In 2.3.4 we relax this assumption, which is necessary if we want the network to be robust to any single router failure. We also explain how to extend the model to changing traffic demands.

For ease of explanation, we elide the distinction between edge nodes and IP nodes; we treat IP nodes as the ultimate sources and destinations of traffic.

2.3.1 Minimizing Network Cost

Our inputs are (i) the optical topology, consisting of the set $\mathcal{I}$ of IP nodes, the set of optical-only nodes, and the fiber links (annotated with distances) between them; and (ii) the demand matrix D.

We use the variable T_u to represent the number of tails that should be placed at router u, and R_u represents the number of regens at node u. An optical-only node can't have any tails.

The capacity of an IP link $\ell = (\alpha, \beta)$ is limited by the number of tails dedicated to ℓ at α and β and the number of regens dedicated to ℓ. Technically, the original signal emitted by α is strong enough to travel REGEN_DIST, and ℓ doesn't need regens there. However, for ease of explanation, we assume that ℓ does need regens at α, regardless of its length. This requirement of regens at the beginning of each IP link is necessary only for the mathematical model and not in the actual network. We add a trivial postprocessing step to remove these regens from the final count before reporting our results. An IP link may require placing regens at an IP node along its path, if it doesn't terminate at that node. We don't remove these regens in postprocessing. Table 2.2 summarizes our notation.

Our objective is to place tails and regens to minimize the ISP's equipment costs while ensuring that the network can carry all necessary traffic under all failure scenarios. Let c_T and c_R be the cost of one tail and one regen, respectively. Then the total cost of all tails is

		Definition
Inputs	$\mathcal{I}$	set of IP nodes
	I	set of IP routers
	N	set of all nodes (optical-only + IP)
	D	demand matrix, where $D_{st} \in D$ is demand from IP node s to IP node t
	F	set of all possible failure scenarios $F = \{f_1, f_2, \ldots, f_n\}$
	$dist_{uvf}$	shortest dist. from optical node u to optical node v in failure scenario f
Outputs (Network Design)	T_u	# tails placed at IP router u
	R_u	total regens placed at node u
Outputs (Network Operation)	$X_{\alpha\beta f}$	capacity of IP link (α, β) in failure scenario f
	$Y_{st\alpha\beta f}$	amount of (s,t) traffic routed on IP link (α, β) in failure scenario f
Intermediate Values	$R_{\alpha\beta uvf}$	# regens at u for optical segment (u, v) of IP link (α, β) in failure f
	R_{uf}	# regens needed at node u in failure scenario f

Table 2.2: Notation.

$c_T \sum_{u \in I} T_u$, the total cost of all regens is $c_R \sum_{u \in N} R_u$, and our objective is

$$\min \quad c_T \sum_{u \in I} T_u + c_R \sum_{u \in N} R_u.$$

The stipulation that the output tail and regen placement work for all failure scenarios is crucial. Without some dynamism in the inputs, be it from a changing topology across failure scenarios or from a changing demand matrix, CD ROADMs' flexible reconfigurability would be useless. We focus on robustness to IP router and optical span failures because conversations with one large ISP indicate that failures affect network conditions more than routine demand fluctuations. Extending our model to find a placement robust to both equipment failures and changing demands should be straightforward.

2.3.2 Robust Placement of Tails and Regens

In traditional networks, robust design requires choosing a single IP link configuration that is optimal for all failure scenarios under the assumption that routing will depend on the specific failure state [17]. With CD ROADMs, robust network design requires choosing a single tail/regen placement that is optimal for all failure scenarios under the assumption that both routing and the IP topology will depend on the specific failure state. In either case, solving the network design problem requires solving the network operation problem as an "inner loop"; to determine the optimal network design we need to simulate how a candidate network would operate, in terms of IP link placement and routing, in each failure scenario.

At the mathematical level, CD ROADMs introduce two additional sets of decision variables to the traditional network design optimization. With the old technology, the problem is to optimize over two sets of decision variables: one set for where to place IP links and what the capacities of those links should be, and a second set for which links different volumes of traffic should traverse. In traditional network design, there is no need to explicitly model tails and regens separate from link placement, because each tail or regen is associated with exactly one IP link. Now, any given tail or regen is not associated with exactly one IP link. Thus, we must decide not only link placement and routing but also the number of tails and regens to place at each IP node and the number of regens to place at each optical node. We describe these two novel aspects of our formulation in turn.

Constraints governing tail placement. Our first constraint requires that the number of tails placed at any router u is enough to accommodate all the IP links u terminates:

$$\begin{aligned} \sum_{\alpha \in I} X_{\alpha u f} &\leq T_u && (2.1) \\ \sum_{\beta \in I} X_{u \beta f} &\leq T_u && (2.2) \\ & \quad \forall u \in I, \forall f \in F \end{aligned}$$

As shown in Table 2.2, $X_{\alpha u f}$ is the capacity of IP link (α, u) in failure scenario f. Hence, $\sum_{\alpha \in I} X_{\alpha u f}$ is the total incoming bandwidth terminating at router u, and Constraint (2.1) says that u needs at least this number of tails. Analogously, $\sum_{\beta \in I} X_{u \beta f}$ is the total outgoing bandwidth from u, and Constraint (2.2) ensures that u has enough tails for these links, too. We don't need T_u greater than the sum of these quantities because each tail supports a bidirectional link.

Constraints governing regen placement. The second fundamental difference between our model and existing work is that we must account for relative positioning of regens both within and across failure scenarios. Because of physical limitations in the distance an optical signal can travel, no IP link can include a span longer than REGEN_DIST without passing through a regenerator. As a result, the decision to place a regen at one location depends on the decisions we make about other locations, both within a single failure scenario and across changing network conditions. Therefore, we introduce auxiliary variables $R_{\alpha\beta uvf}$ to represent the number of regens to place at node u for the link between IP routers (α, β) in failure scenario f *such that the next regen traversed will be at node v.*

Ultimately, we want to solve for R_u, the number of regens to place at u, which doesn't depend on the IP link, next-hop regen, or failure scenario. But, we need the $R_{\alpha\beta uvf}$ variables to encode these dependencies in our constraints. We connect R_u to $R_{\alpha\beta uvf}$ with the constraint

$$R_u \geq \sum_{\substack{\alpha,\beta \in I \\ v \in N}} R_{\alpha\beta uvf} \quad \forall u \in N, \forall f \in F. \tag{2.3}$$

We use four additional constraints for the $R_{\alpha\beta uvf}$ variables. First, we prevent some node v from being the next-hop regen for some node u if the shortest path between u and v exceeds REGEN_DIST:

$$\begin{aligned} R_{\alpha\beta uvf} &= 0 \\ & \forall \alpha, \beta \in I, \\ & \forall u, v \text{ such that } dist_{uvf} > \text{REGEN\_DIST}. \end{aligned}$$

Second, we ensure that the set of regens assigned to an IP link indeed forms a contiguous path. That is, for all nodes u aside from those housing the source and destination routers, the number of regens assigned to u equals the number of regens for which u is the next-hop:

$$\begin{aligned} \sum_{v \in N} R_{\alpha\beta uvf} &= \sum_{v \in N} R_{\alpha\beta vuf} \\ & \forall u \in N, \forall \alpha, \beta \in I, \forall f \in F. \end{aligned}$$

We need sufficient regens at the source IP router's node a, and sufficient regens with the destination IP router's node b as their next-hop, for each IP link

$$\begin{aligned}\sum_{u \in N} R_{\alpha\beta auf} &\geq X_{\alpha\beta f} \\ \sum_{u \in N} R_{\alpha\beta ubf} &\geq X_{\alpha\beta f} \\ &\forall \alpha, \beta \in I, \forall f \in F;\end{aligned}$$

But, b can't have any regens, and a can't be the next-hop location for any regens

$$\begin{aligned}R_{\alpha\beta uaf} = R_{\alpha\beta buf} &= 0 \\ &\forall u \in N, \forall \alpha, \beta \in I, \forall f \in F.\end{aligned}$$

Additional practical constraints. We have two practical constraints which are not fundamental to the general problem but are artifacts of the current state of routing technology. First, ISPs build IP links in bandwidths that are multiples of 100 Gbps. We encode this policy by requiring $X_{\alpha\beta f}$, T_u, and R_u to be integers and converting our demand matrix into 100 Gbps units.

Second, current IP and optical equipment require each IP link to have equal capacity to its opposite direction. With these constraints, only one of (2.1) and (2.2) is necessary.

Finally, we require all variables to take on nonnegative values.

2.3.3 Dynamic Placement of IP Links

Thus far, we have described constraints ensuring that each IP link has enough tails and regens. But, we have not discussed IP link placement or routing. Although link placement and routing *themselves* are part of network operation rather than network design, they play central roles as *parts* of the network design problem. How many are "enough" tails and regens for each IP link depends on the link's capacity, and the link's capacity depends on how much traffic it must carry. Therefore, the network operation problem is a subproblem of our network design optimization.

These constraints are the well-known multicommodity flow (MCF) constraints requiring (a) flow conservation; (b) that all demands are sent and received; and (c) that the traffic assigned to a particular IP link cannot exceed the link's capacity. $Y_{st\alpha\beta f}$ gives the amount of (s,t) traffic routed on IP link (α,β) in failure scenario f. Hence, we express these constraints with the following equations:

$$\sum_{u\in I} Y_{stuvf} = \sum_{u\in I} Y_{stvuf} \qquad \forall(s,t)\in D, \quad \forall v \in I-\{s,t\}, \forall f\in F \tag{2.4}$$

$$\sum_{u\in I} Y_{stsuf} = \sum_{u\in I} Y_{stutf} = D_{st} \qquad \forall s,t\in D, \forall f \in F \tag{2.5}$$

$$\sum_{(s,t)\in D} Y_{stuvf} \le X_{uvf} \qquad \forall u,v\in I, \forall f\in F. \tag{2.6}$$

As before, X_{uvf} in Constraint (2.6) is the capacity of IP link (u,v) in failure scenario f.

Network design and operation in practice. Once the network has been designed, we solve the network operation problem for whichever failure scenario represents the current state of the network by replacing variables T_u and R_u with their assigned values.

2.3.4 Extensions to a Wider Variety of Settings

We now describe how to relax the assumptions we've made throughout Sections 2.3.1 - 2.3.3 that (a) each IP node houses exactly one IP router; and (b) traffic demands are constant.

Accounting for multiple routers colocated at a single IP node. If we assume that IP links connecting routers colocated within the same IP node always have the same cost as (short) external IP links (i.e., they require one tail at each router endpoint), then our model already allows for any number of IP routers at each IP node; if this assumption holds, then we simply treat colocated routers as if they were housed in nearby nodes e.g., one mile apart. However, in general this assumption is not valid, because intra-IP-node links require one port per router, rather than a full tail (combination router port and optical transponder) at each end. Hence, intra IP node links are cheaper than even the shortest external links. To accurately model costs we must account for them explicitly.

To do so, we add the stipulation to all the constraints presented above that, whenever one constraint involves two IP routers, these IP routers cannot be colocated. Then, we add the following:

Let U be the set of IP routers containing u and any other routers u' collocated at the same IP node with u. Let P_u be the number of ports placed at u for intra-node links. Let c_P be the cost of one 100 Gbps port. Our objective function now becomes

$$\min \quad c_T \sum_{u \in I} T_u + c_R \sum_{u \in N} R_u + c_P \sum_{u \in I} P_u.$$

Ultimately, we want to constrain the traffic traveling between u and any u' to fit within the intra-node links, as follows (c.f. Constraint (2.6)).

$$\sum_{(s,t) \in D} Y_{stuu'f} \leq X_{uu'f} \forall u, u' \in U, \forall U \in \mathcal{I}, \forall f \in F.$$

But, no $X_{uu'f}$ appear in the objective function; the links themselves have no defined cost. Hence, we add constraints to limit the capacity of the links to the number of ports P_u. Specifically, we use the analogs of (2.1) and (2.2) to describe the relationship between ports P_u placed at u (c.f. tails placed at u) and the intra-node links starting from (c.f. $X_{u\beta f}$

external IP links) and ending at (c.f. $X_{\alpha uf}$ external IP links) u.

$$\begin{aligned}
\sum_{u' \in U} X_{u'uf} &\leq P_u \\
\sum_{u' \in U} X_{uu'f} &\leq P_u \\
&\forall U \in \mathcal{I}, \forall u \in U, \forall f \in F
\end{aligned}$$

Accounting for changing traffic. Thus far, we have described our model to accommodate changing failure conditions over time with a single traffic matrix. In reality, traffic shifts as well. Adding this to the mathematical formulation is trivial. Wherever we currently consider all failure scenarios $f \in F$, we need only consider all (failure, traffic matrix) pairs. Unfortunately, while this change is straightforward from a mathematical perspective, it is computationally costly. The number of failure scenarios is a multiplicative factor on the model's complexity. If we extend it to consider multiple traffic matrices, the number of different traffic matrices serves as an additional multiplier.

2.4 Scalable Approximations

In theory, the network design algorithm presented above finds the optimal solution. We will call this approach Optimal. However, Optimal does not scale, even to networks of moderate size ($\sim$ 20 IP nodes). To address this issue, we introduce two approximations, Simple and Greedy.

Optimal is unscalable because, as network size increases, not only does the problem for any given failure scenario become more complex, but the number of failure scenarios also increases. In a network with ℓ optical spans, n IP nodes, and d separate demands, the total number of variables and constraints in Optimal is a monotonically increasing function $g(\ell, n, d)$ of the size of the network and demand matrix, multiplied by the number of failure scenarios, $\ell + n$. Thus, increasing network size has a multiplicative effect on Optimal's complexity. The key to Simple and Greedy is to decouple the two factors.

2.4.1 Simple Parallelizing of Failure Scenarios

In Simple, we solve the placement problem separately for each failure condition. That is, if Optimal jointly considers failure scenarios labeled $F = \{1, 2, 3\}$, then Simple solves one optimization for $F = \{1\}$, another for $F = \{2\}$, and a third for $F = \{3\}$. The final number of tails and regens required at each site is the maximum required over all scenarios. Each of the $\ell + n$ optimizations is exactly as described in Section 2.3; the only difference is the definition of F. Hence, each optimization has $g(\ell, n, d)$ variables and constraints. The problems are independent of each other, and therefore we can solve for all failure scenarios in parallel. As network size increases, we only pay for the increase in $g(\ell, n, d)$, without an extra multiplicative penalty for an increasing number of failure scenarios.

2.4.2 Greedy Sequencing of Failure Scenarios

Greedy is similar to Simple, except we solve for the separate failure scenarios in sequence, taking into account where tails and regens have been placed in previous iterations. In Simple, the $\ell+n$ optimizations are completely independent, which is ideal from a time efficiency perspective. However, one drawback is that Simple misses some opportunities to share tails and regens across failure scenarios. Often, the algorithm is indifferent between placing tails at router a or router b, so it arbitrarily chooses one. Simple might happen to choose a for Failure 1 and b for Failure 2, thereby producing a final solution with tails at both. In contrast, Greedy knows when solving for Failure 2 that tails have already been placed at a in the solution to Failure 1. Thus, Greedy knows that a better *overall* solution is to reuse these, rather than place additional tails at b.

Mathematically, Greedy is like Simple in that it requires solving $|F|$ separate optimizations, each considering one failure scenario. But, letting T'_u represent the number of tails already placed at u, we replace Constraints (2.1) and (2.2) with the following.

$$\sum_{\alpha \in I} X_{\alpha u f} \leq T_u + T'_u \tag{2.7}$$

$$\sum_{\beta \in I} X_{u \beta f} \leq T_u + T'_u \tag{2.8}$$

$$\forall u \in I, \forall f \in F$$

In (2.7) and (2.8), T_u represents the number of new tails to place at router u, not counting the T'_u already placed. Similarly, with R'_u defined as the number of regens already placed at u and R_u as the new regens to place, Constraint (2.3) becomes

$$R_u + R'_u \geq \sum_{\substack{\alpha, \beta \in I \\ v \in O}} R_{\alpha \beta u v f} \quad \forall u \in O, \forall f \in F.$$

We always solve the no-failure scenario first, as a baseline. After that, we try a variety of orderings of the other failure scenarios and choose whichever produces the lowest cost solution.

With Greedy, we solve for the $\ell + n$ failure scenarios in sequence, but each problem has only $g(\ell, n, d)$ variables and constraints. The number of failure scenarios is now an additive factor, rather than a multiplicative one in Optimal or absent in Simple.

2.4.3 Roles of Simple, Greedy, and Optimal

As we will show in Section 2.5, Greedy finds nearly equivalent-cost solutions to Optimal in a fraction of the time. Simple universally performs worse than both. We introduce Simple for theoretical completeness, though due to its poor performance we don't recommend it in practice; Simple and Optimal represent the two extremes of the spectrum of joint optimization across failure scenarios, and Greedy falls in between.

We see both Optimal and Greedy as useful and complementary tools for network design, with each algorithm best suited to its own set of use cases. Optimal helps us understand exactly how our constraints regarding tails, regens, and demands interact and affect the final solution. It is best used on a scaled-down, simplified network to (a) answer questions such as *How do changes in the relative costs of tails and regens affect the final solution?*; and (b) serve as a baseline for Greedy. Without Optimal, we wouldn't know how close Greedy comes to finding the optimal solution. Hence, we might fruitlessly continue searching for a better heuristic. Once we demonstrate that Optimal and Greedy find comparable solutions on topologies that both can solve, we have some reason to believe that Greedy will do a good job on networks too large for Optimal.

In contrast, Greedy's time efficiency makes it ideally suited to place tails and regens for the full-sized network. In addition, Greedy directly models the process of incrementally upgrading an existing network. The foundation of Greedy is to take some tails and regens as fixed and to optimize the placement of additional equipment to meet the constraints. When we explained Greedy, we described these already placed tails and regens as resulting from previously considered failure scenarios. But, they can just as well have previously existed in the network.

2.5 Evaluation

First, we show that CD ROADMs indeed offer savings compared to the existing, fixed IP link technology by showing that all of Simple, Greedy, and Optimal outperform current best practices in network design. Then we compare these three algorithms in terms of quality of solutions and scalability. We show that Greedy achieves similar results to Optimal in less time. Finally, we demonstrate that our algorithms should allow ISPs to meet their SLAs even during the transition period following a failure before the network has had time to transition to the new optimal IP link configuration.

2.5.1 Experiment Setup

Topology and traffic matrix. Figure 2.5 shows the topology used for our experiments, which is a scaled-down version of the core of a backbone network of a large ISP. The network shown in Figure 2.5 has nine edge switches, which are the sources and destinations of all traffic demands. Each edge switch is connected to two IP routers, which are colocated within one PoP and share a single optical connection to the outside world. The network has an additional 16 optical-only nodes, which serve as possible regen locations.

To isolate the benefits of our approach to minimizing tails and regens, respectively, we create two versions of the topology in Figure 2.5. The first, which we call 9node-450, assigns a distance of 450 miles to each optical span. In this topology neighboring IP routers are only 900 miles apart, so an IP link between them doesn't need a regen. The second version, 9node-600, assigns a distance of 600 miles to each optical span. In this topology, regens are required for any IP link.

To evaluate our optimizations on networks of various sizes, we also look at a topology consisting of just the upper left corner of Figure 2.5 (above the horizontal thick dashed line and to the left of the vertical thick dashed line). We refer to the 450 mile version of this topology as 4node-450 and the 600 mile version as 4node-600. Second, we look at the upper two-thirds (above the thick dashed line) with optical spans of 450 miles (6node-450) and 600 miles (6node-600). Finally, we consider the entire topology (9node-450 and 9node-600).

For each topology, we use a traffic matrix in which each edge router sends 440 GB/sec to each other edge router. In our experiments we assume costs of 1 unit for each tail and 1 unit for each regen, while communication between colocated routers is free. We use Gurobi version 8 to solve our linear programs.

Alternative strategy. We compare Optimal, Greedy, and Simple to Legacy, the method currently used by ISPs to construct their networks. Once built, an IP link is fixed, and if any component fails, the link is down and all other components previously dedicated to it are unusable.

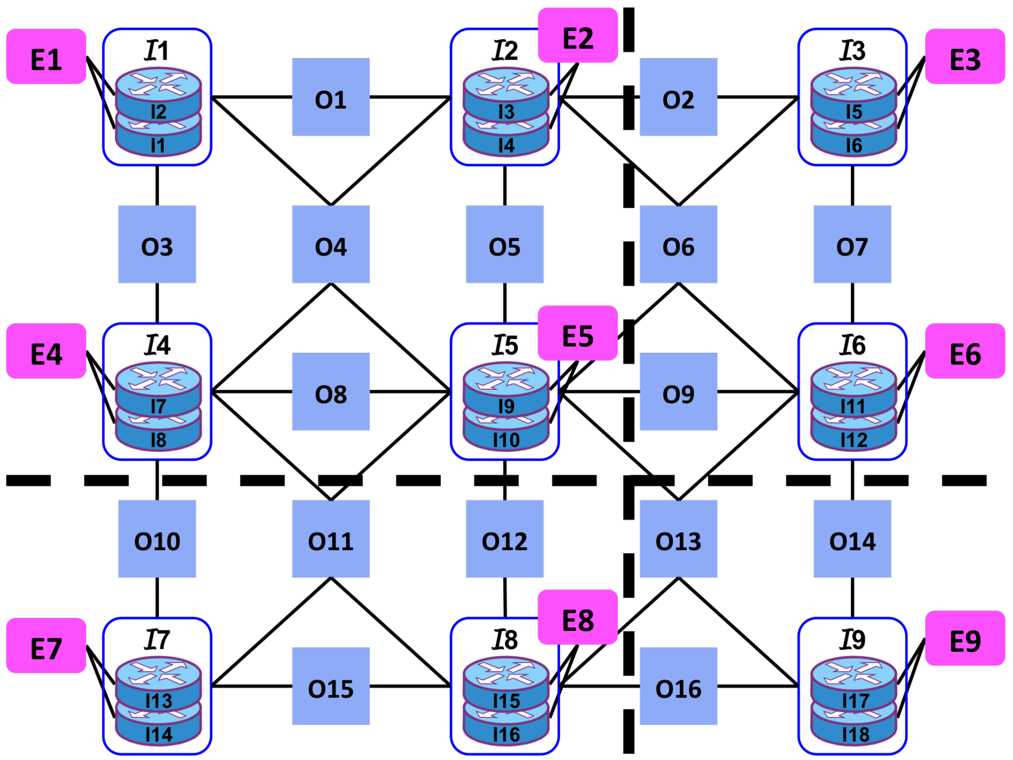
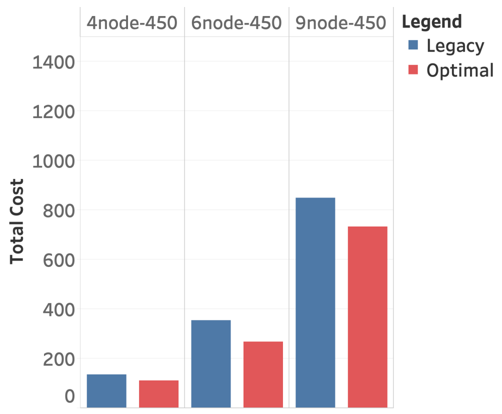


Figure 2.5: Topology used for experiments. We call the full network 9node-450/9node-600, the upper two-thirds (above the thick dashed line) 6node-450/6node-600, and the upper left corner 4node-450/4node-600.

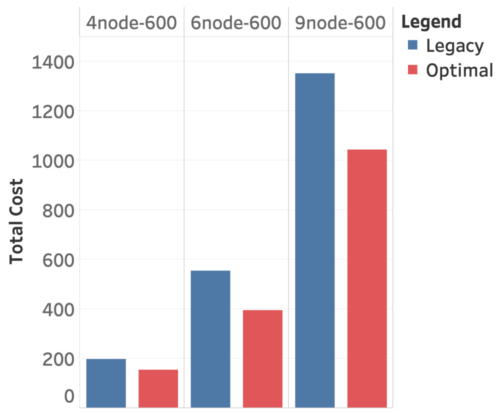
In our Legacy algorithm, we assume that IP links follow the shortest optical path. Similar to Greedy, we begin by computing the optimal IP topology for the no-failure case. We then designate those links as already paid for and solve the first failure case under the condition that reusing any of these links is "free." We add any additional links placed in this iteration to the already-placed collection and repeat this process for all failure scenarios.

Legacy is the pure IP layer optimization and failure restoration described in Section 2.2. As discussed previously, we need not compare our approaches to pure optical restoration, because pure optical restoration cannot recover from IP router failures. We need not compare against independent optical and IP restoration, because this technique generally performs worse than pure-IP or IP-along-disjoint-paths.

We compare against IP-along-shortest-paths, rather than IP-along-disjoint-paths, for two reasons. First, the main drawback of IP-along-shortest-paths is that, in general, it does



(a) Neighboring optical nodes 450 miles apart.



(b) Neighboring optical nodes 600 miles apart.

Figure 2.6: Total cost (tails + regens) by topology for Optimal and Legacy, unit cost for each tail/regen. Optimal outperforms Legacy on all topologies, and the absolute gap is greatest on the largest network (9node-600). We get the greatest percentage savings on 6node-600.

not guarantee recovery from optical span failure. However, on our example topologies, *as in most real ISP backbones*, Legacy *can* handle any optical failure, since the topologies are sufficiently richly connected. Second, the formulation of the rigorous IP-along-disjoint-paths optimization is nearly as complex as the formulation of Optimal; if we remove the restriction that IP links must follow shortest paths, then we need constraints like those described in Section 2.3.2 to place regens every 1000 miles along a link's path. For this reason, ISPs generally do not formulate and solve the rigorous IP-along-disjoint-paths optimization. Instead, they hand-place IP links according to heuristics and historical precedent. We don't use this approach because it is too subjective and not scientifically replicable. In summary, IP-along-shortest-paths strikes the appropriate balance among (a) effectiveness at finding close to the optimal solution possible with traditional technology; (b) realism; (c) simplicity for our implementation and explanation; and (d) simplicity for the reader's understanding and ability to replicate.

2.5.2 Benefits of CD ROADMs

To justify the utility of CD ROADM technology, we show that building an optimal CD ROADM network offers up to 29% savings compared to building a legacy network (6node-600 topology). Since neither approach requires any regens on the 450 mile networks, all savings in those settings (Figure 2.6a) come from tails. On 4node-600 Optimal requires 15% fewer tails, 38% fewer regens, and 23% fewer tails and regens together. On 6node-600 we achieve even greater savings, cutting total equipment costs by 29% by using 20% fewer tails and 44% fewer regens. On 9node-600 Optimal uses 16% *more* tails than Legacy but more than compensates by requiring 55% fewer regens, for an overall savings of 23%. The bars in Figures 2.6 illustrate the differences in total cost. Looking at Figures 2.6a and 2.6b, we see that Optimal offers greater savings compared to Legacy on the 600 mile networks. This is because regens, more so than tails, present opportunities for reuse across failure scenarios. Optimal capitalizes on this opportunity while Legacy doesn't; both algorithms find solutions with close to the theoretical lower bound in tails, but Legacy in general is inefficient with regen placement. Since no regens are necessary for the 450 mile topologies, this benefit of Optimal compared to Legacy only manifests itself on the 600 mile networks.

In these experiments we allow up to five minutes per failure scenario for Legacy and the equivalent total time for Optimal (i.e., 300 sec $\times$ 21 failure scenarios = 6300 sec for 4node-450 and 4node-600, 300 sec $\times$ 35 failures = 10,500 sec for 6node-450 and 6node-600 and $300 \times 59 = 17,700$ sec for 9node-450 and 9node-600). Recall that as "failure scenarios" we consider any single IP router or optical span failure. For example, the 21 failure scenarios for the small topologies come from eight IP routers, 12 optical spans, and one no-failure condition.

2.5.3 Scalability Benefits of Greedy

As Figure 2.7 shows, Greedy outperforms Optimal when both are limited to a short amount of time. "Short" here is relative to topology; Figure 2.7 illustrates that the crossover point is around 1200 seconds for 4node-600. In contrast, both Greedy and Optimal always outperform Simple, even at the shortest time limits. The design that Greedy produces costs at most 1.3% more than the design generated by Optimal, while Simple's design costs up to 12.4% more than that of Optimal and 11.0% more than that of Greedy. Reported times for these experiments do

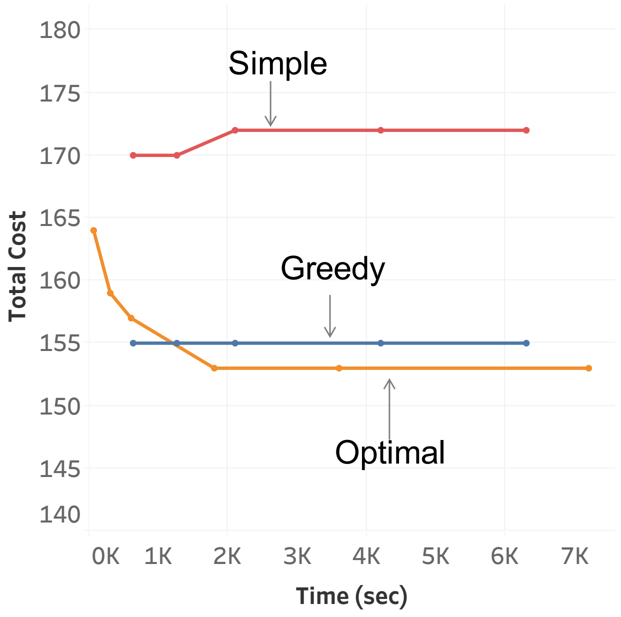


Figure 2.7: Total cost by computation time for Simple, Greedy, and Optimal on 4node-600. Lines do not start at $t = 0$ because Gurobi requires some amount of time to find any feasible solution.

not parallelize Simple's failure scenarios; we show the summed total time. In addition, the times for Greedy and Simple are an upper bound. We set a time limit of t seconds for each of $|F|$ failure scenarios, and we plot each algorithm's objective value at $t|F|$.

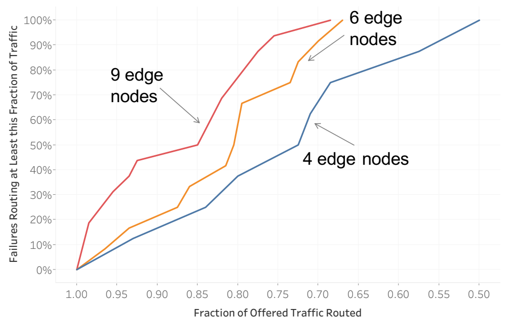
Interestingly, the objective values of Simple for this topology, and Greedy for some others, do not monotonically decrease with increasing time. We suspect this is because their solutions for failure scenario i depend on their solutions to all previous failures. Suppose that, on failure $i - j$, Gurobi finds a solution s of cost c after 60 seconds. If given 100 seconds per failure scenario, Gurobi might use the extra time to pivot from the particular solution s to an equivalent cost solution s', in an endeavor to find a configuration with an objective value less than c on this particular iteration. Since both s and s' give a cost of c for iteration $i - j$, Gurobi has no problem returning s'. But, it's possible that s' ultimately leads to a slightly worse overall solution than s. As Figure 2.7 shows, these differences are at most 10 tails and regens, and they occur only at the lowest time limits.

2.5.4 Behavior During IP Link Reconfiguration

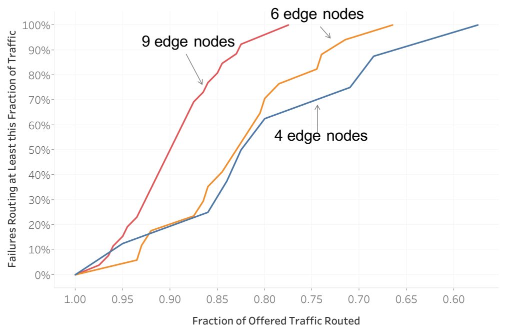
In the previous two subsections, we evaluate the steady-state performance of Optimal, along with Greedy, Simple, and Legacy, after the network has had time to transition both routing and the IP link configuration to their new optimal settings based on the current failure scenario. However, as we describe in Section 2.2.3, there exists a period of approximately two minutes during which routing has already adapted to the new network conditions but IP links have not yet finished reconfiguration. In this section we show that our approach gracefully handles this transient period, as well.

The fundamental difference between these experiments and those in Sections 2.5.2 and 2.5.3 is that here we disallow IP link reconfiguration. Whereas in Sections 2.5.2 and 2.5.3 we jointly optimize both IP link configuration and routing in response to each failure scenario, we now reoptimize only routing; for each failure scenario we restrict ourselves to the links that were both already established in the no-failure case and have not been brought down by said failure. Specifically, in these experiments we begin with the no-failure IP link configuration as determined by Optimal. Then, one-by-one we consider each failure scenario, noting the fraction of offered traffic we can carry on this topology simply by switching from Optimal's no-failure routing to whatever is now the best setup given the failure under consideration.

Figure 2.8 shows our results. The graphs are CDFs illustrating the fraction of failure scenarios indicated on the y-axis for which we can deliver at least the fraction of traffic denoted by the x-axis. For example, the red point at $(0.85, 50\%)$ in Figure 2.8a indicates that in 50% of the 59 failure scenarios under consideration for 9node-450, we can deliver at least 85% of offered traffic just by reoptimizing routing. The blue line in Figure 2.8a represents the results of taking the 21 failure scenarios of 4node-450 in turn, and for each recording the fraction of offered traffic routed. The blue line in Figure 2.8b shows the same for the 21 failure scenarios of 4node-600, while the orange lines show the 35 failure scenarios for 6node-450 and 6node-600, and the red lines show the 59 failure scenarios for the large topologies.



(a) Neighboring optical nodes 450 miles apart.



(b) Neighboring optical nodes 600 miles apart.

Figure 2.8: Percentage of failure scenarios for which rerouting over the existing IP links allows delivery of at least the indicated fraction of offered traffic.

We find two key takeaways from Figure 2.8. First, across all six topologies we always deliver at least 50% of traffic. Second, our results improve as the number of nodes in the network increases, and we do better on the topologies requiring regens than on those that don't. On 9node-600, we're always able to route at least 80% of traffic. Generally, ISPs' SLAs require them to always deliver all high priority traffic, which typically represents about 40-60% of total load. However, in the presence of failures or extreme congestion they're allowed to drop low priority traffic. These results are promising for translating to real ISP topologies, since most operational backbones are larger even than our 9node-600 topology. Note that we don't expect to be able to route 100% of offered traffic in all failure scenarios without reconfiguring IP links; if we could, there would be little reason to go through the reconfiguration process at all. But, we already saw in Section 2.5.2 that remapping the IP topology to the optical underlay adds significant value.

2.5.5 Evaluation on a Realistic Backbone Topology

As we have shown, this network design problem is both complicated to formulate and computationally complex to solve. Our Greedy heuristic serves as an important step toward making our solution scalable, but as described thus far, even Greedy cannot be used on a graph closely resembling AT&T's actual topology, which consists of 202 links, 68 edge nodes, and 40 optical backbone nodes, of which 23 are also IP nodes and 17 are optical-only. Each of the 23

IP nodes contains two IP routers. The primary bottleneck is CPU, as Gurobi cannot parallelize its process of finding an initial feasible solution [28], and after running our experiments for seven days on our university's Beowulf cluster [5], we don't even get an initial feasible solution for the first failure scenario.

To scale Greedy to this realistic setting, we make the following modifications. **We collapse the two routers located at each IP node into one.** This simplification reduces the computational complexity of the problem by cutting the number of IP routers in half, from 46 to 23. Once we do this, we can no longer consider router failures in our set of failure solutions, because when the single router at each IP node fails, the edge switches connected to that node become detached from the network. However, it is straightforward to adapt the solution on this simplified problem to get a good but not necessarily optimal solution to the real network and our target set of failure scenarios (any single link or router).

Suppose our solution calls for placing T_{u0} tails at the single IP router in the simplified topology in the no-failure case and T_u tails per IP router in the simplified topology when considering all single optical link failures. Then we place $\max(T_{u0}, 0.5T_u)$ per router in the actual network. This solution must be feasible, because when solving for the real network we allow for *either* one router failure *or* one optical link failure. Placing T_{u0} tails *per router* covers the situations in which a router failure is worse for a given node than any optical link failure. Placing T_u tails *per node* covers the situations in which an optical link failure requires more tails at a given router than any router failure. Since we aren't allowing for a router failure and optical link failure simultaneously, we can divide these T_u tails across the two routers at each node. Hence, by choosing the maximum of T_{u0} and $0.5T_u$ for each router, we ensure that we can always carry the necessary traffic.

We prevent IP links from taking particularly circuitous optical paths. Specifically, we prevent an IP link from using a regen at a given optical node if doing so would require the IP link to be more than some constant `path_stretch` times longer than its shortest path. This restriction simplifies the problem by limiting the number of $R_{\alpha\beta uvf}$ we need to solve for.

Also, it does little to worsen the final solution, because indirect, long routes are unlikely to require fewer regens than shorter routes. Further, even if this restriction marginally increases the monetary cost of the network design, the ISP might prefer a solution that avoids the increased latency inherent in choosing these longer paths.

We simplify the traffic matrix. The full traffic matrix for AT&T's backbone network contains 2648 demands. We reduce the size of the problem by randomly choosing 1500 or 1000 of these. To translate our results from this smaller traffic matrix to the real network, we can aggregate the demands of nearby edge switches. As long as the routing between the group of aggregated edge switches is straightforward, we can choose one to serve as the representative of the group for the purposes of the optimization. Then, in postprocessing we can deal with routing packets to specific edge switches within each group.

We relax Gurobi's error tolerance. Gurobi solves an ILP by computing a lower bound on the best possible solution and finding some feasible solution as an upper bound. It then works to raise the lower bound and lower the upper bound until the two are within some configurable fraction of each other. By slightly increasing this number to $\sim 10\%$, we allow the solver to terminate with an optimal solution more quickly than if it had to continue the process until its best feasible solution was within $\sim 1\%$ or 0.1% of its lower bound.

We provide Gurobi with a partial initial feasible solution. The vast majority of the variables in this problem relate to regen placement. We set initial feasible values for these variables by assuming that a dedicated IP link connects each (s, t) pair. We set the $R_{\alpha\beta uvf}$ variables such that, in each failure scenario, this link follows its shortest possible path and has enough capacity to carry all (s, t) traffic.

With all these optimizations, we are confident that Gurobi will be able to solve the network design problem on a realistic backbone topology. Whereas we don't find an initial feasible solution to the first failure scenario after seven days when we try to solve the full problem, with these simplifications we find an optimal solution to the first two failure scenarios within several days. However, the full computation may take weeks or months to run, since the ILP for each failure scenario remains complex, and we have 202 failure scenarios. In practice, this is a reasonable runtime for the network design computation, because this problem by definition operates on large timescales and only needs to be solved every few months or years. We leave to future work a full evaluation of the results of running this optimized version of Greedy on the realistic topology and how this setting compares to the smaller networks we discuss above (Sections 2.5.1 - 2.5.4).

2.6 Related Work

Perhaps most similar to our work is that by Papanikolaou et al. [46–48], who present an ILP for finding a minimal cost network design for IP-over-elastic-optical-networks. Our work goes beyond theirs in that we avoid precalculating optical paths to determine regen placement. On the other hand, their work is more detailed than ours in that they choose each link's transmission rate and spectrum.

Another class of related work addresses *either* IP link reconfiguration and routing *or* tail and regen placement, but not both, as we do. For example, the Owan work by Jin et al. [37] optimizes IP link reconfiguration and routing to minimize completion time for bulk transfers, but they assume that tails and regens are fixed. Like our work, Owan is a centralized system to jointly optimize the IP and optical topologies and configure network devices, including CD ROADMs, according to this global strategy. However, there are three key differences between Owan and our project. First and foremost, Jin et al. take the locations of optical equipment as an input constraint, while we solve for the optimal places to put tails and regens. This distinction is crucial, as a main source of complexity in our model is the need to make decisions on two separate timescales. Second, our objective differs from that of Jin et al. We aim to minimize the cost of tails and regens, while they aim to minimize the transfer completion time or maximize the number of transfers that meet their deadlines. Third, our work applies in a different setting. Owan is designed for bulk transfers and depends on the network operator being able to control sending rates, possibly delaying traffic for several hours. We target all ISP traffic; we can't rate control any traffic, and we must route all demands, even in the case of failures, except during a brief transient period during IP link reconfiguration.

Similarly to Jin et al., Gerstel et al. address the IP link configuration problem without placing tails and regens [24]. Like us, they take as input the end-to-end IP traffic matrix, the optical layer topology, and the set of possible failures their IP-optical mapping must withstand. Unlike us, they must start with an existing IP topology, which can be the ISP's current setup or any reasonable mapping. Our technique can modify an existing IP topology or start from scratch. In general, Gerstel et al. discuss similar ideas to ours about and reasons for multilayer optimization, but they don't present a complete formulation of an optimal algorithm for how to achieve it.

In contrast to the work by Jin et al. and Gerstel et al., which address IP link reconfiguration but not tail and regen placement, Bathula et al. minimize the number of regen *sites* without discussing how to reconfigure IP links in response to failures [11]. Further, their work differs from ours in that we aim to minimize the total number of regens.

Our work is also related, though less directly, to various projects addressing failure recovery [22, 54, 58, 61] and robust optimization [16, 29, 41]. Also relevant is the work by Brzezinski et al. [14], which demonstrates that, to minimize delay, it is best to set up direct IP links between endpoints exchanging significant amounts of traffic, while relying on packet switching through multiple hops to handle lower demands. Finally, some previous projects have attempted to solve our same joint tail/regen placement, IP link reconfiguration, and routing problem but present only heuristics without a formulation of the full optimization problem [21].

2.7 Conclusion

Advances in optical technology and SDN have decoupled IP links from their underlying infrastructure (tails and regens). We have precisely stated and solved the new network design problem deriving from these advances, and we have also presented a fast approximation algorithm that comes close to the optimal solution.

Chapter 3

Red: A Communist Approach to Network Operation

3.1 Introduction

As we mention in the Introduction to this thesis (Chapter 1), ISP backbone networks are infrastructure as crucial to modern life as roads and utilities. Therefore, even though these networks are largely invisible to us as consumers, we care a great deal about how well they serve their purpose of providing fast, reliable communication. The architecture of any given ISP backbone plays a key role in determining how well the network can meet these needs. Hence, we ought to think carefully about how we can improve upon the standard design and operation models of today's networks.

The vast majority of existing ISP backbones, along with most research proposals, fall into one of two categories:

(i) *smart-edge-dumb-interior*: all edge switches are "smart," and all interior nodes are "dumb"

(ii) *all-nodes-equal*: all switches are homogeneous.

When we refer to "smart" vs. "dumb" switches, we are specifically concerned with three axes along which switches differ:

Splitting flows over multiple next-hops. Every switch maintains a *forwarding table* in its memory, where the first column in the table *matches* on one or more of the packet's header fields and the second *action* column specifies matching packets' next-hop in the backbone network. Because switches must process a high volume of packets at rapid speeds, these tables are generally implemented in hardware. As such, they offer limited flexibiilty to stray from this basic match-action paradigm; many can do nothing more than send equal proportions across equal (minimal) cost paths. Researchers have explored how to work around these hardware restrictions and approximate splitting a given "match" across multiple next-hops in arbitrary proportions [38]. However, to do so, these proposals require multiple table entries for certain paths, thereby consuming additional memory and leaving less room for other destination prefixes. In contrast to these severely constrained legacy switches, most modern switches place no restrictions on which paths can be used for load balancing, but even some of these require equal splits [1].

Processing various header fields. SDN switches running OpenFlow 1.0 can match on 12 specific header fields, while those running OpenFlow 1.4 can match on 41 [13]. Even protocols as basic as multiprotocol label switching (MPLS) are not supported by all switches [4]. At the other extreme, programmable switches allow matching on arbitrary combinations of bits.

Responding to link failures. Some switches can participate in protocols to detect and recover from link failures (e.g., bidirectional forwarding detection (BFD)), while others cannot. And, some switches support only a subset of protocol features [3].

Even though many networks fit either the smart-edge-dumb-interior or all-nodes-equal models, some networks don't fall into either category. Because of factors such as incremental upgrades [12], business mergers [45], and efforts to avoid vendor lock in, *networks are heterogeneous*; edge switches generally are quite "smart," but some interior nodes are, too. Heterogeneous networks preclude routing and failure recovery architectures that rely on complex processing from interior switches. But, they're also not well-suited to approaches that assume all interior switches are "dumb," because these solutions waste the resources of the "smarter" interior nodes. In this chapter, we propose a "sweet spot" intermediate architecture that takes advantage of the resources of whichever interior nodes happen to be "smarter" without *requiring* complex functionality from any. Our solution also allows ISPs to purposefully and selectively add "smart" nodes to the network interior to derive the routing and failure recovery benefits that we describe below.

Table 3.1 summarizes the existing backbone network designs and the network operation models typically associated with each, and it also shows how our proposal fits in. As we mention above, the goal of any ISP backbone is to provide fast, reliable communication. To do so, it must limit congestion, even in the presence of failures. Three factors critical to meeting these goals are (i) the speed and ease with which the network can recover from switch or link failures; (ii) the cost of routers/switches; and (iii) the ability of the routing scheme to distribute packets over all links in the network, rather than overcrowding a few common paths. We care about the cost of routers and switches because the cheaper each individual box is, the more machines an ISP can afford, the more capacity it can handle, and the less congestion in the network.

Having outlined these goals, we now explain the basic design and operation models of the existing approaches and examine how well they fulfill their objectives.

In an *all-nodes-equal* model, routing is generally done through an interior gateway protocol (IGP) such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS). Each router is responsible for maintaining its own complete map of the network and calculating the best next-hop to use for any given destination IP address. These protocols require expensive equipment, because they require routers to perform complex logic to construct their forwarding tables, and they require significant memory for storing at least one table entry per destination IP prefix. In addition, they limit path diversity by

Design	**Operation Model**	**Inexpensive Equipment**	**Fast Failure Rec. w/ Ltd. Overhead**	**Diverse Paths**
all-nodes-equal	hop-by-hop	✗	✓✢	✗
smart-edge-dumb-interior	edge-to-edge tunneling	✓✢	✗	✗
heterogeneous nodes	**Red**	✓	✓	✓

Table 3.1: Properties of various TE/failure recovery architectures.

requiring packets to travel along shortest paths. On the other hand, they do allow for rapid recovery after a switch or link fails.

In contrast, in a *smart-edge-dumb-interior* model, routing is generally done via edge-to-edge tunneling [10,23,30,32,33,35,49,53,54]. When a packet enters one of these networks, the ingress edge determines the appropriate egress edge switch. It labels the packet accordingly, and the interior switches use this label, rather than the packet's destination IP address, to route the packet. In some cases, interior nodes are still responsible for calculating the appropriate next-hop to use for any given egress switch, while in others the label encodes the packet's full path. Either way, this model imposes less of a burden on interior node processing power and memory than does the all-nodes-equal model, since they only need space in memory proportional to the number of egress switches in the network, rather than space proportional to the number of destination IP prefixes in the Internet. However, failure recovery with edge-to-edge tunneling is either slow or imposes significant overhead on switch memory. One common edge-to-edge tunneling technique, MPLS, offers fast reroute (FRR), which promises to switch traffic from a failed path to a live path within tens of milliseconds. But, ISPs can only use FRR if their interior nodes support it, which somewhat negates the cost savings promised by the smart-edge-dumb-interior model by preventing them from using the least expensive "dumb" switches on the market. In addition, FRR is complex, requiring nodes to exchange multiple messages to set up the backup tunnels [8,60]. Another drawback of the smart-edge-dumb-interior model, which it shares with the all-nodes-equal architecture, is that it allows packets to make use of only a limited number of paths.

Therefore, as Table 3.1 shows, both existing network architectures offer positive qualities but also suffer from serious drawbacks. We propose a hybrid model that combines the strengths of each of the two existing approaches and avoids their weaknesses. More precisely, we consider a setting of an ISP backbone in which edge nodes are "smart," but interior nodes can be either "smart" (*big nodes*) or "dumb" (*little nodes*). Since interior big nodes are just as capable as edge nodes, we propose to tunnel big node-to-big node, rather than edge-to-edge.

The chief advantage of big node-to-big node tunneling over the all-nodes-equal model is reduced cost for the ISP. Whereas hop-by-hop forwarding in an all-nodes-equal network demands that *all* switches have enough memory to store forwarding table entries for all destination IP prefixes, big node-to-big node tunneling allows those little nodes with limited memories to maintain just one table entry per backbone egress node. But, big nodes with large memories can still make use of them as they would in an all-nodes-equal setting. Further, in an all-nodes-equal model, if the ISP wants to introduce new functionality or finds that its switches' routing tables are growing over time, it must upgrade *all* switches.

Given the massive size and geographic expanse of backbone networks and the high cost of switches and routers, upgrading an entire network at once is logistically infeasible and prohibitively expensive. Therefore, ISPs continuously engage in years-long upgrade cycles; they're constantly refreshing select portions of their networks such that every few years they turn over their full suite of equipment. However, by the time they reach the end of each cycle, the switches that were replaced in early phases are several years old, outdated and "dumb" compared to the ones replaced at the end.

With big node-to-big node tunneling, whichever switches have been replaced most recently can serve as big nodes. As the upgrade cycle proceeds and today's big nodes become more limited compared to switches that have been upgraded even more recently, the current generation of big nodes becomes little nodes, and the new generation becomes big nodes. Since Red places no restrictions on the locations of interior big nodes, this process need not impact how the

ISP chooses to go about updating its network. However, the ISP may choose to target its upgrades to maximize the benefits of Red.

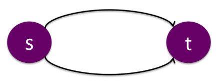
Having explained how big node-to-big node tunneling is an improvement over hop-by-hop forwarding in an all-nodes-equal architecture, we now address edge-to-edge tunneling and the smart-edge-dumb-interior design. The chief advantage of big node-to-big node tunneling over the edge-to-edge model is faster failure recovery. The problem with edge-to-edge tunneling is that it creates long paths, which increase the time necessary to recover from a failure. Generally, edge nodes monitor the liveness of their paths via some protocol based on exchanging heartbeat probes (e.g., BFD [39, 40]). The longer the path, the longer before the ingress realizes that probes are not getting through, and consequently the longer before it can take steps to route around the failure. And, all else being equal, a long path is more likely to fail, because it contains more individual links, each of which may fail at any time.

Another advantage of big node-to-big node tunneling over the edge-to-edge approach is greater path diversity. Figure 3.1a shows an example network in which the ingress switch s can split flows destined for egress t over two paths, but all forwarding is restricted to edge-to-edge tunneling. For simplicity, we omit from the figure any nodes internal to a tunnel, but Figures 3.1a, 3.1b, and 3.1c all depict the same physical network.

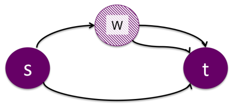
In contrast, consider Figure 3.1b. In this network, we assume that one node along the upper path happens to be as powerful as s. Ingress s can still split flows over only two paths, but now W can do the same. Hence, while each (s, t) flow in Figure 3.1a can make use of two total paths, an (s, t) flow in Figure 3.1b can use three. In addition, these path diversity gains are multiplicative. As Figure 3.1c shows, with three interior big nodes, each (s, t) flow can use nine paths.

3.1.1 Chapter Outline

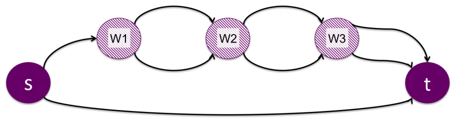
Even though the *potential* gains from big node-to-big node tunneling are immediately apparent, we must overcome two key research challenges to realize these benefits in practice. We must:



(a) Edge-to-edge tunneling allows only two paths.



(b) Taking advantage of W's splitting ability provides an additional path.



(c) Path diversity gains from interior big nodes are multiplicative; three inerior big nodes allow for nine (s, t) paths.

Figure 3.1: Example topology illustrating how edge-to-edge tunneling limits path diversity. We assume the underlying network is the same in both a and b; in both cases there exist more paths, but we only show those that (s, t) traffic is allowed to use.

1. formulate and solve the routing traffic engineering (TE) optimization problem, which requires simultaneously choosing the optimal sequence of tunnels a flow should take while also determining the optimal set of physical paths to implement each tunnel; and

2. devise a failure recovery protocol to take advantage of the shorter paths produced by our big node-to-big node approach.

In Section 3.2 we describe our overlay model that serves as the foundation to our solutions to both of these problems. We then describe our TE formulation (Section 3.3) and failure recovery protocol (Section 3.4). We evaluate our architecture in Section 3.5, and in Section 3.6 we explain the mathematics underlying some of our seemingly surprising results. In Section 3.7 we discuss related work, and we conclude in Section 3.8.

3.2 Overlay Model

The fundamental idea behind Red is that switch capabilities are not necessarily tied to location in the network. big nodes play the role of traditional edge switches and are responsible for both packet-processing logic and executing the Red failure recovery protocol. Importantly,

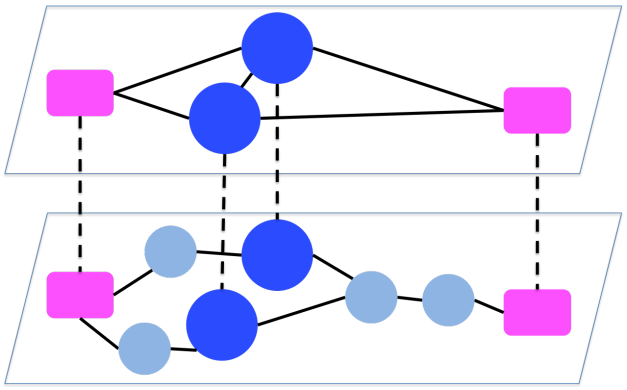


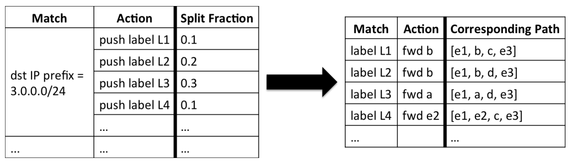
Figure 3.2: Example physical topology with its associated virtual overlay graph.

big nodes may be in the middle of the network; if an interior switch has the necessary capabilities (Section 3.2), then it is a big node. little nodes play the role of traditional interior switches and are responsible only for exact-match forwarding. We use the term waypoint to refer to a big node in the middle of the network.

To be able to compute the optimal routing and failure recovery for a big node/little node network, we extend the notion of edge switches and their tunnels as an *overlay* graph on the physical network. Our overlay contains all big nodes and the tunnels between them.

Formally, the Red network model consists of the physical network graph $\mathcal{G}$ and also an overlay virtual topology $\mathcal{V}$. The physical graph contains all switches and links actually present in the ISP's network. The virtual graph contains only the big nodes of the physical network; internal little nodes are omitted from the virtual graph. A virtual link exists between two big nodes if there exists a physical path in the physical graph between the two switches that doesn't traverse an intermediate big node. Figure 3.2 shows an example physical topology and its associated virtual overlay graph.

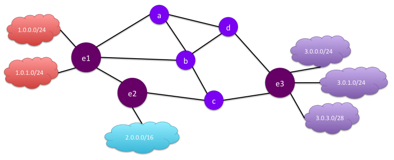
The Red forwarding model is to tunnel big node-to-big node, rather than edge-to-edge. Instead of having a packet's ingress switch determine its entire path through the ISP, Red has a packet's ingress choose (i) the next big node the packet should reach; and (ii) its path to this next-hop big node. The next-hop big node then chooses the next big node and a path to reach it,



Match	Action	Split Fraction
dst IP prefix = 3.0.0.0/24	push label L1	0.1
	push label L2	0.2
	push label L3	0.3
	push label L4	0.1
	...	...
...	...	...

Match	Action	Corresponding Path
label L1	fwd b	[e1, b, c, e3]
label L2	fwd b	[e1, b, d, e3]
label L3	fwd a	[e1, a, d, e3]
label L4	fwd e2	[e1, e2, c, e3]
...		...

(a) Table pipeline for big node $e1$ in b.



(b) Network corresponding to table pipeline shown in a.

Figure 3.3: big node table pipeline.

and so on until the packet reaches a big node that determines the packet should be forwarded outside the ISP.

Switch Capabilities

More specifically, we assume each big node has a two-stage table pipeline. When a packet p enters a big node W, the first table matches on an arbitrary set of header fields and assigns p to one of k_1 next-hop big nodes. These eligible next-hop big nodes are the k_1 neighbor big nodes of W most directly "en route" to p's destination. We describe in more detail how the set of eligible next-hops is chosen, and how any given next-hop is chosen from the set, in Section 3.3. The second table matches on this assigned next-hop big node and chooses one of k_2 physical paths. Finally, the switch pushes a label corresponding to this physical path and forwards p to the first switch on the path. A little node has just one table; when p enters a little node, the switch matches on the label and forwards p to the indicated next-hop. Figure 3.3a shows an example of big node $e1$'s table pipeline for the network in Figure 3.3b.

3.3 Traffic Engineering

At startup, Red's central controller directs traffic on optimal routes for the baseline, no-failure case. Ultimately, Red's TE algorithm must compute forwarding rules for the controller to send to

(a) big nodes indicating

Variable	Definition
$\mathcal{W}$	set of all waypoints
$\mathcal{E}$	set of all edge switches
$\mathcal{L}$	set of all internal little nodes
$\mathcal{G}$	physical graph; nodes of the physical graph are $\mathcal{E} \cup \mathcal{W} \cup \mathcal{L}$
$\mathcal{V}$	virtual overlay graph; nodes of the overlay graph are $\mathcal{E} \cup \mathcal{W}$ (big nodes)
$c_{\alpha\beta}$	capacity of link (α, β)
D_{st}	demand from edge switch s to edge switch t
Z_{stabp}	amount of traffic from src. edge s to dst. edge t that traverses the virtual link from src big node a to dst big node b on path p
B_{stabp}	indicator that $Z_{stabp} > 0$
$I_{\alpha\beta p}$	indicator that physical link (α, β) is on path p; $I_{\alpha\beta p} = 1$ if path p uses link (α, β) else 0
$\mathcal{P}_{ab}$	set of allowed paths implementing virtual link (a, b)
nhbn(W, t)	set of allowed next-hop big nodes at big node W for dst. edge t

Table 3.2: Notation.

(i) the fraction of each flow from source IP prefix s to destination IP prefix t to forward to each next-hop big node; and

(ii) the fraction of flow destined to each next-hop big node to forward to each physical next-hop; and

(b) little nodes indicating the single next-hop to forward all packets labeled as destined to a particular big node on a particular path.

To simplify the problem formulation, we solve for the *volume* of flow to be sent on each link, rather than the percentage splits. We convert to splitting fractions as a final, post-processing step before actually installing the rules. Hence, in mathematical notation, we solve for the values Z_{stabp}, the volume of (s, t) flow assigned to overlay link (a, b) that should traverse physical path p.

We formulate Red's optimization first by extending the multicommodity flow framework and then adding mechanisms to enforce switch flow splitting constraints and to reduce computation time. Table 3.2 summarizes our notation.

3.3.1 Extending MCF

At its core, our formulation of Red's optimization is an extension of MCF, a general paradigm used for routing various "commodities" from their respective sources to their respective destinations in a network. "Optimal" can be defined in various ways — in *max flow* applications the objective is to maximize the total volume of flows routed, in *max concurrent flow* applications the objective is to satisfy demands for each commodity while maximizing the fraction of links kept under their set capacities, and in *min cost* applications the objective is to route all demands at a minimal cost while respecting all capacity constraints [56]. The problem's constraints include *capacity constraints* on the maximum volume of traffic that can be routed on each edge and *flow conservation constraints* to ensure that all the traffic that is sent by each source node arrives at its appropriate destination.

Although a variety of flavors of MCF exist in the literature, we have not encountered an existing formulation that applies to our overlay/underlay setting in which nodes are limited in the number of next-hops across which they can transmit. We now describe our novel adaptation of MCF, highlighting the similarities and differences between our formulation and the standard setup.

As with traditional MCF, Red's initial inputs comprise the physical topology $\mathcal{G}$ and the demand matrix of bandwidth requirements for all edge-to-edge flows. Red's information about the physical topology includes the capacities of all links and the role of each switch (edge switch, waypoint, or little node). Red uses these inputs to create the overlay graph $\mathcal{V}$ (Section 3.2).

Assume for now that Red uses its knowledge of the physical network to compute the entire set $\mathcal{P}_{ab}$ of physical paths implementing each overlay link (a, b). We relax this requirement in Section 3.3.3. Incorporating $\mathcal{P}_{ab}$ into the flow conservation constraints is how we apply MCF to our overlay/underlay setting and a key way in which our formulation differs from traditional MCF.

Constraints

Flow conservation at all waypoints.

$$\sum_{\substack{a\in\mathcal{V}\\ p\in\mathcal{P}_{aw}}} Z_{stawp} = \sum_{\substack{b\in\mathcal{V}\\ p\in\mathcal{P}_{wb}}} Z_{stwbp} \quad \forall s,t\in\mathcal{E}, \forall w\in\mathcal{V}-\{s,t\}. \tag{3.1}$$

Source edge sends all demand; destination edge receives all demand.

$$\sum_{\substack{b\in\mathcal{V}\\ p\in\mathcal{P}_{sb}}} Z_{stsbp} = \sum_{\substack{a\in\mathcal{V}\\ p\in\mathcal{P}_{at}}} Z_{statp} = D_{st} \quad \forall s,t\in\mathcal{E}. \tag{3.2}$$

Source edge receives nothing; destination edge sends nothing.

$$\sum_{p\in\mathcal{P}_{as}} Z_{stasp} = \sum_{p\in\mathcal{P}_{tb}} Z_{sttbp} = 0 \quad \forall s,t\in\mathcal{E}, \forall a\in\mathcal{V}. \tag{3.3}$$

Objective

Minimize a convex function of link utilization [59]:

$$\min \sum_{\alpha\beta} \Phi(\alpha,\beta) \quad \forall \alpha,\beta\in\mathcal{G}, \tag{3.4}$$

where

$$u_{\alpha\beta} = \frac{f_{\alpha\beta}}{c_{\alpha\beta}} \tag{3.5}$$

and

$$\Phi(\alpha,\beta) = \begin{cases} f_{\alpha\beta} & u_{\alpha\beta} \le 1/3 \\ 3f_{\alpha\beta} - 2/3c_{\alpha\beta} & 1/3 \le u_{\alpha\beta} \le 2/3 \\ 10f_{\alpha\beta} - 16/3c_{\alpha\beta} & 2/3 \le u_{\alpha\beta} \le 9/10 \\ 70f_{\alpha\beta} - 178/3c_{\alpha\beta} & 9/10 \le u_{\alpha\beta} \le 1 \\ 500f_{\alpha\beta} - 1468/3c_{\alpha\beta} & 1 \le u_{\alpha\beta} \le 11/10 \\ 5000f_{\alpha\beta} - 16318/3c_{\alpha\beta} & 11/10 \le u_{\alpha\beta}. \end{cases} \tag{3.6}$$

Figure 3.4: Formulation of Red's LP.

Figure 3.4 shows our formulation. Constraints (3.1)-(3.3) are standard flow conservation constraints. Our objective is a convex function of link utilization designed to minimize queuing delay. Specifically, Equation (3.6) is a piecewise-linear approximation of the M/M/1 delay formula for packets on a particular link (α,β), and our objective (3.4) is to minimize the sum of delays across the entire network.

3.3.2 Enforcing Splitting Constraints

As described thus far, the formulation in Figure 3.4 distinguishes between big nodes and little nodes but does not capture switch flow splitting limitations. To fix this problem, we can add the following constraints, where M is a constant greater than the maximum volume that could possibly be sent on any path.

$$B_{stabp} \geq \frac{Z_{stabp}}{M} \qquad \forall s,t \in \mathcal{E}, \forall a,b \in \mathcal{V}, \forall p \in \mathcal{P}_{ab} \tag{3.7}$$

$$\sum_{p \in \mathcal{P}_{ab}} B_{stabp} \leq k_2 \qquad \forall s,t \in \mathcal{E}, \forall a,b \in \mathcal{V} \tag{3.8}$$

$$\sum_{\substack{b \in \mathcal{V} \\ p \in \mathcal{P}_{ab}}} B_{stabp} \leq k_1 \qquad \forall s,t \in \mathcal{E}, \forall a \in \mathcal{V} \tag{3.9}$$

$$B_{stabp} \in \{0,1\} \qquad \forall s,t \in \mathcal{E}, \forall a,b \in \mathcal{V}, \forall p \in \mathcal{P}_{ab} \tag{3.10}$$

Together, constraints (3.7) and (3.10) ensure that $B_{stabp} = 1$ if any (s,t) traffic is sent on path p of overlay link (a,b). Constraint (3.8) forces each overlay link (a,b) to use no more than k_2 physical paths per (s,t) flow. Constraint (3.9) forces big node a to use no more than k_1 next-hop big nodes b for each (s,t) flow, regardless of the physical paths.

Although the formulation in Figure 3.4 combined with constraints (3.7)-(3.10) correctly enforce splitting constraints, this problem is an integer linear program (ILP). ILPs are NP-hard, while continuous LPs can be solved in polynomial time. In both theory and practice, solving an ILP takes exponentially longer than solving a similar sized LP. Additionally, choosing a suitable value for M is difficult; M must be large enough such that $\frac{Z_{stabp}}{M}$ is never greater than 1 but small enough not to cause floating point errors.

3.3.3 Reducing Computation Time

Because formulating Red's TE problem as an ILP would make it intractable, we instead enforce splitting constraints via two pre-processing steps. First, we enforce k_1 by restricting each big node W to a set of up to k_1 eligible next-hop big nodes for each (s,t) flow. We use the heuristic of choosing the k_1 next-hops that minimize path stretch en route to t and denote this set as $\mathsf{nhbn}(W,t)$.

Second, we enumerate only k_2 paths for each overlay link, rather than all possible paths; each set $\mathcal{P}_{ab}$ has at most k_2 elements. This strategy simultaneously prevents Red from violating switch limitations and also drastically reduces the time to compute $\mathcal{P}_{ab}$.

Red's process for choosing paths allows for flexibly prioritizing path disjointness and path length. We iteratively apply a standard shortest paths algorithm [2], beginning with setting the weight $w_{\alpha\beta}$ of any physical link from switch α to switch β to be inversely proportional to its capacity:

$$w_{\alpha\beta} = \frac{\max_{(\gamma,\delta)\in\mathcal{G}} c_{\gamma\delta}}{c_{\alpha\beta}}. \tag{3.11}$$

The numerator in (3.11) is the highest capacity of any link in the network and is constant for all links. The denominator is the capacity of the specific link (α,β). At each iteration, we take the shortest not-yet-chosen path and then multiply the weights of all links in this path by a configurable parameter `disjoint_path_multiplier` (`dpm`). Hence, if all links have equal capacity, setting `dpm` $= 1$ is equivalent to choosing the k_2 paths with the fewest hops. Increasing `dpm` gives greater priority to path disjointness at the cost of path length.

With these limited sets of next-hop big nodes and physical paths, the LP in Figure 3.4 does respect switch limitations, even without the integer constraints (3.7)-(3.10).

3.3.4 Putting it All Together

Figure 3.5 illustrates Red's complete four-step process for computing optimal route assignments. First, Red uses its knowledge of the physical network to create the overlay graph.

Second, Red chooses nhbn(W, t) and enumerates k_2 physical paths for each overlay link. Finally, Red solves the LP shown in Figure 3.4 to compute the volume of (s, t) traffic assigned to each physical path p implementing each allowed overlay link (a, b). As a post-processing step, Red converts these flow volumes to splitting proportions and pushes the appropriate rules to switches. Red repeats this process every-so-often as traffic demands change but *does not* go through all these steps in response to link failures.

3.4 Failure Recovery

The existence of interior waypoints, combined with Red's failure recovery protocol, allow Red to respond more quickly to most link failures than is possible with edge-to-edge tunneling. Red's failure recovery protocol tries to localize the process as much as possible, which makes recovery faster and limits the number of live links asked to take on more load. Red's specific response depends on the location of the failed link within the physical and virtual topology in relation to the path of a given flow. In most cases, the closest big node preceding the failed link can renormalize its flow splitting ratios to avoid the failure; we say these are instances of *local* recovery, because the central controller isn't involved. In the few situations when local recovery isn't possible, the controller computes new paths only for the specific flows that can't be recovered locally.

In the following subsections, we define the precise situations in which local recovery is possible and which require a global response. We also detail exactly how Red reroutes any given flow f in each case. Assume that f's no-failure path includes $[W_1, \alpha, \beta, W_2]$, where W_1 and W_2 are big nodes and α and β are little nodes. Link (α, β) fails.

3.4.1 Local Recovery

Local recovery comprises two sub cases.

1: W_1 has at least one live path to W_2. In this case, W_1 renormalizes its splitting ratios across its remaining live paths to W_2. For example, suppose W_1 initially sends 100 units of

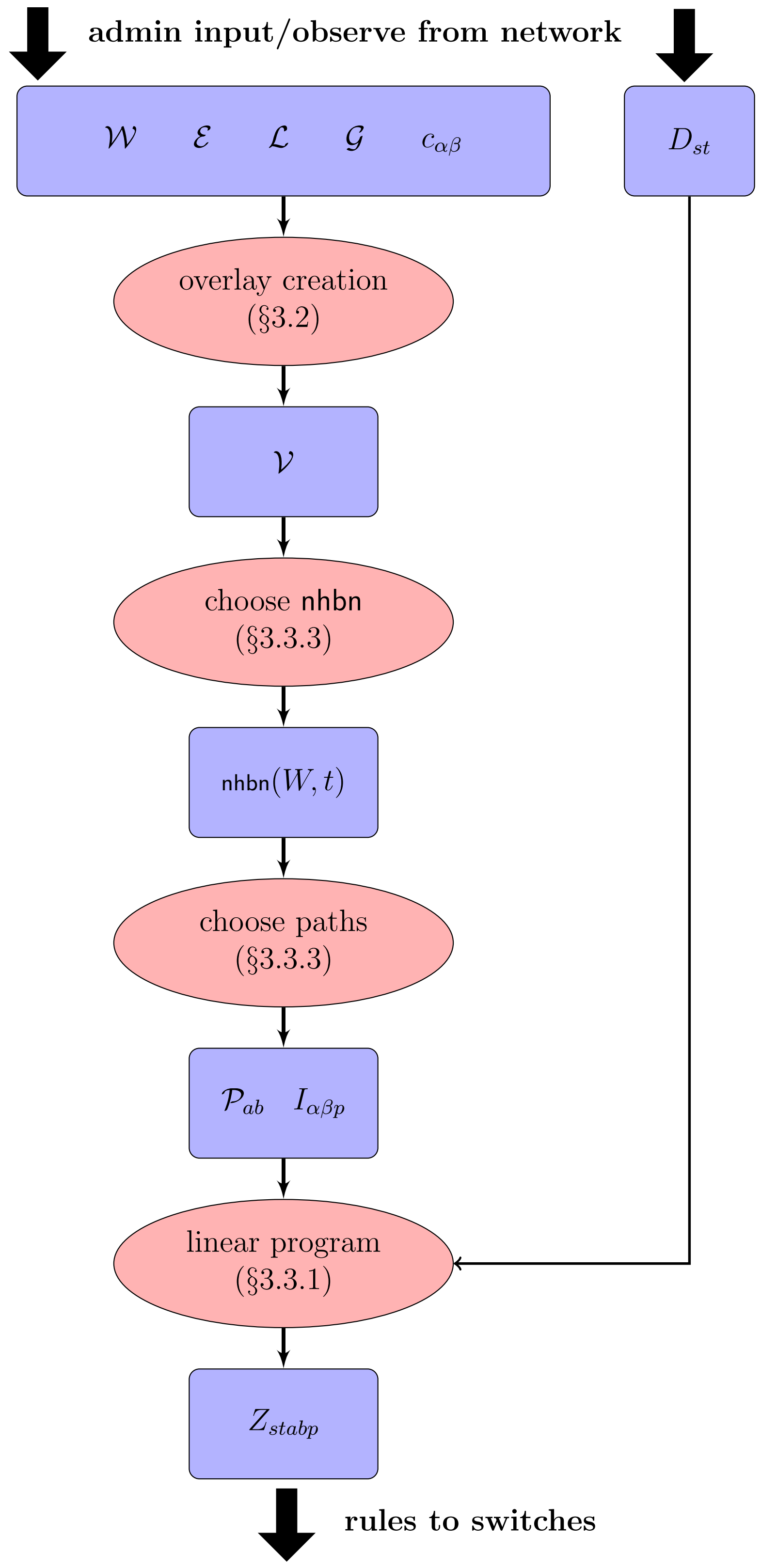


Figure 3.5: Red's TE algorithm. Blue rectangles represent inputs and outputs to the various phases of computation. Red ovals represent these computational processes. Table 3.2 summarizes the meanings the various symbols.

traffic to W_2 on paths (p_1, p_2, p_3) in volumes $(40, 10, 50)$. Then, p_1 fails. The remaining live paths together initially accounted for 60 units of traffic, with p_2 responsible for 10. Therefore, p_2 takes an additional $\frac{10}{60} \cdot \big(1 - (60/100)\big) = 0.067$ fraction of total demand. p_3 takes the other $\frac{50}{60} \cdot \big(1 - (60/100)\big) = 0.333$, giving new weights of $(p_1, p_2, p_3) = (0, 0.167, 0.833)$ and new volumes $(0, 17, 83)$.

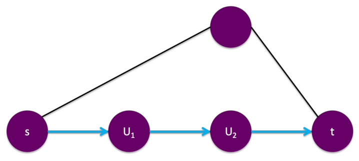
If W_1 has remaining live paths to W_2 but none of these paths was initially assigned any traffic, then W_1 splits its demand equally among them.

2: W_1 has no live paths to W_2 but still has an overlay link to at least one eligible next-hop big node for f. In this case, W_1 renormalizes its splitting ratios across its remaining next-hop big nodes for f; the computation is analogous to that described above for renormalizing across remaining paths to a given destination big node.

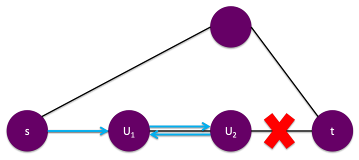
3.4.2 Global Recovery

As described thus far, local recovery should always be possible. However, as described thus far local recovery can also create forwarding loops, if we allow a big node to send to a previously unused neighbor big node, as illustrated in Figure 3.6. For ease of explanation, Figure 3.6 shows a physical network in which all switches are big nodes; the physical and overlay graphs are identical. Suppose Red's initial optimization routes all (s, t) flow along the route $[s, U_1, U_2, t]$ (Figure 3.6a). Then, link (U_2, t) fails. Further, assume $k_1 = 2$, so both t and U_1 are in U_2's initial $\mathsf{nhbn}(U_2, t)$. If U_2 is allowed to recover from the (U_2, t) failure by sending to any next-hop big node in its initial $\mathsf{nhbn}(U_2, t)$ set, then U_2 can forward the traffic it receives from U_1 right back to U_1. And, U_1 will continue to send to U_2, because (by design) U_1 doesn't know about the (U_2, t) failure. Hence, packets will loop between U_1 and U_2 if U_1 is allowed to remain in $\mathsf{nhbn}(U_2, t)$ (Figure 3.6b).

To avoid this problem, immediately after solving the LP and before the network gets up and running, we remove any big node U_1 from $\mathsf{nhbn}(U_2, t)$ if the initial solution to the LP does not route any traffic on the overlay link (U_1, U_2).



(a) Initial routing.



(b) Routing after the failure of (U_2, t) if U_1 is allowed to remain in $\mathsf{nhbn}(U_2, t)$.

Figure 3.6: Example topology illustrating why we must remove initially unused big nodes from nhbn. Black lines represent bidirectional links, and blue arrows show flow routes.

This measure is sufficient to ensure that our failure recovery protocol will never create loops. We know that the initial optimal routing won't include loops, and by removing unused next-hop big nodes, we confine ourselves to the assuredly safe set of overlay links chosen by the LP solution. However, the cost of pruning nhbn is that it is possible that W_1 may have no remaining virtual links to any *eligible* next-hop big nodes without being physically partitioned from t. Hence arises the possible need for global recovery.

3: W_1 can't reach an eligible next-hop big node for f. This is the only case in which Red's central controller must be involved in failure recovery. The controller always knows the current state of the network and tracks each big node's eligible next-hop big nodes for each (s, t) flow. Therefore, the controller realizes whenever a failure leaves W_1 with no eligible next-hops for f. At this point, the controller reoptimizes to choose a completely new route for f. This reoptimization takes into account how other big nodes have responded to the failure for all flows. The controller simulates all the big nodes' renormalizations and then, assuming these flows are fixed, chooses a new route for f. Finally, the controller sends rules to the switches on f's new route to implement the recovery.

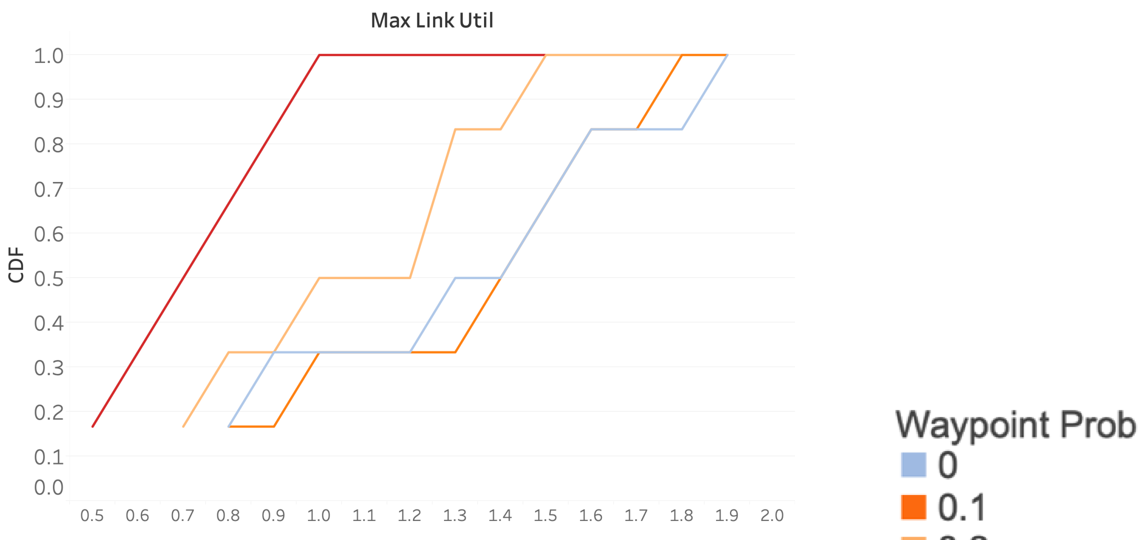
Although recovery in this scenario takes more time than in either of the above cases because it requires communication with the central controller, it still happens much more quickly than if the controller recomputed the optimization from scratch. The controller can solve this optimization quickly, because it its responsible only for rerouting the small number of flows that traverse some waypoint with no eligible next-hop big nodes.

3.5 Evaluation

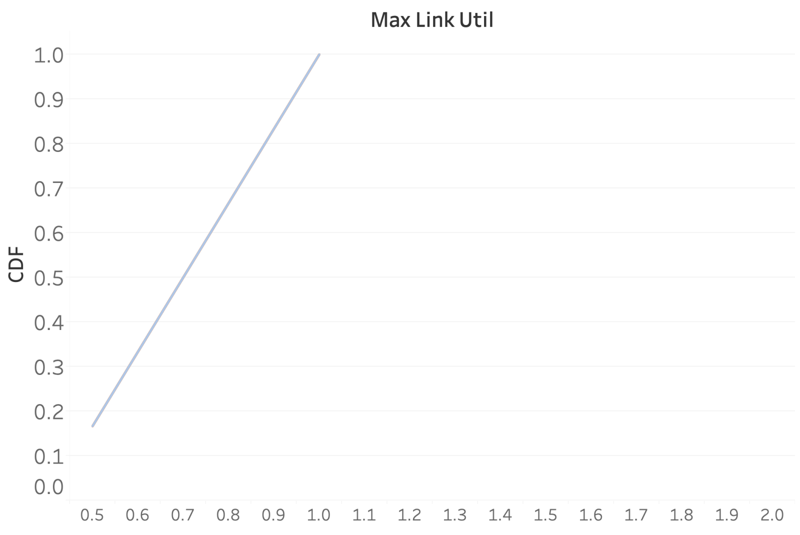
In this section we show experimental results evaluating Red's traffic engineering and failure recovery algorithms. We evaluate Red using a topology representative of AT&T's backbone and a subset of demands from the associated traffic matrix, both of which we obtained from the company itself. The network contains 209 bidirectional edges and 114 nodes. In our experiments we choose big nodes randomly to mimic a situation in which big nodes happen to exist at various points throughout the network. We also choose edge nodes randomly, though for this we restrict ourselves to choosing from the 68 nodes that AT&T designates as actual edge switches in its network. Our demands comprise 42 flows from a particular source edge switch to a particular destination edge switch.

3.5.1 Traffic Engineering

In the absence of failures, big node-to-big node tunneling does not dramatically reduce congestion compared to the traditional edge-to-edge approach. We evaluate the impact of interior waypoints on traffic engineering by randomly assigning nodes to be waypoints or little nodes according to varying probabilities and then computing the maximum link utilization resulting from routing traffic according to the algorithm described in Section 3.3. We refer to the parameter controlling the probability that any given interior node is a waypoint as `waypoint_prob`. Figures 3.7a and 3.7b, show CDFs of the fraction of trials for which the maximum link utilization is at most a certain value. The trials differ in the traffic demands they serve. We plot separate lines for `waypoint_prob` in $\{0.0, 0.1, 0.2, 0.5, 1.0\}$. Figure 3.7a assumes that each big node can split flows between two next-hop big nodes and use only one physical path to reach each next-hop big node $(k_1 = 2, k_2 = 1)$. Figure 3.7b assumes that each big node can split flows among three next-hop big nodes and use up to three physical paths to reach each



(a) $k_1 = 2, k_2 = 1$



(b) $k_1 = 3, k_2 = 3$

Figure 3.7: CDF showing fraction of trials for which maximum link utilization is at most a certain value, no link failures.

next-hop big node ($k_1 = 3, k_2 = 3$). The case in which `waypoint_prob` $= 0$ is equivalent to edge-to-edge tunneling.

If big node-to-big node tunneling has a significant impact on reducing congestion, the colored lines should appear far apart. In particular, the blue line for `waypoint_prob` $= 0$ should be far to the right, and then the lines for successively higher values of `waypoint_prob` should fall farther and farther left. We see this pattern to some extent when big nodes are severely limited in the number of next-hops across which they can split flows (Figure 3.7a), although the lines for `waypoint_prob` = 0 and `waypoint_prob` = 0.1 cross each other, and the red `waypoint_prob` = 0.5 line falls directly on top of the brown `waypoint_prob` = 1 line (which is why the brown line is not visible in the figure). By the time big nodes can split flows across three next-hop big nodes and use three physical paths to get to each next-hop big node, the case in which all nodes are big nodes is indistinguishable from the case in which no nodes are big nodes. The blue `waypoint_prob` = 0 line is the only one that appears in Figure 3.7b because it sits directly on top of the other four.

3.5.2 Failure Recovery

In this section we show that the presence of interior waypoints indeed reduces failure recovery latency by shortening tunnel lengths and also that these waypoints, in conjunction with Red's failure recovery protocol, reduce congestion compared to the no-waypoint case.

Reduced Path Length

Figure 3.8 shows that, as we would expect, the distance between waypoints decreases as we increase the number of waypoints in the network. The x-axis shows `waypoint_prob`, and the y-axis shows three different measures of big node-to-big node path length:

- *avg avg distance*: The average distance from a big node to each of its neighbor big nodes, averaged over all big nodes.
- *avg max distance*: The distance from a big node to its farthest immediate neighbor big node, averaged over all big nodes.
- *max max distance*: The maximum over all big nodes of each big node's distance to its farthest immediate neighbor big node.

All three metrics show a consistent dropoff as the number of waypoints increases. Our results are not particularly sensitive to the number of edge switches. As demonstrated by Figure

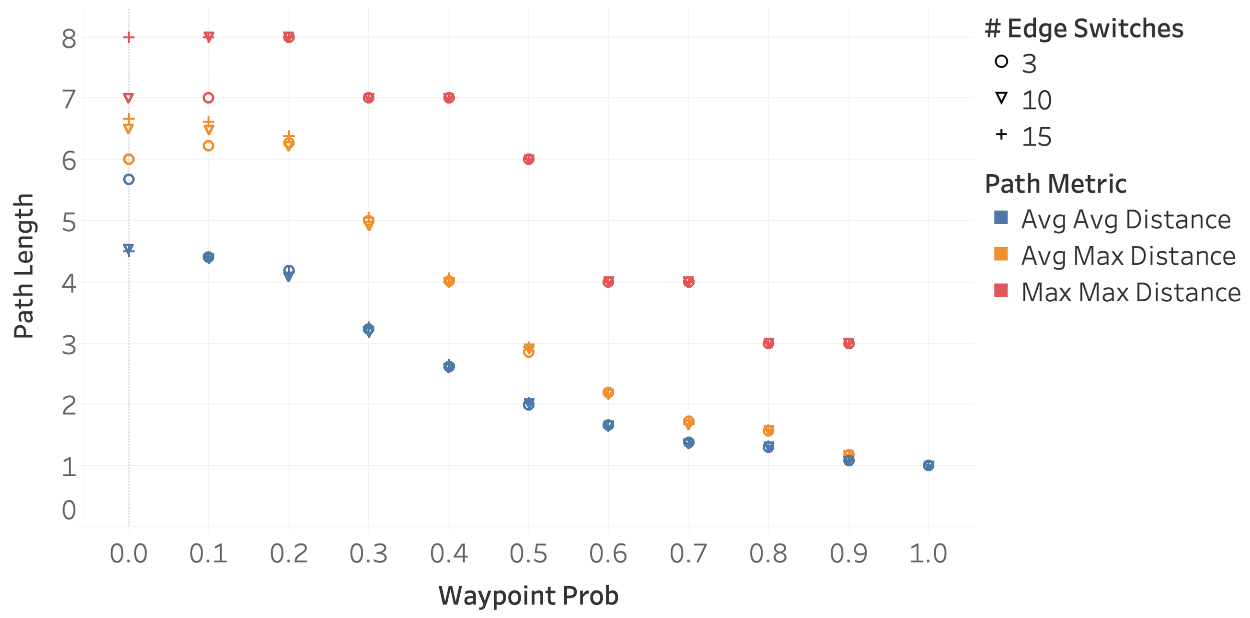


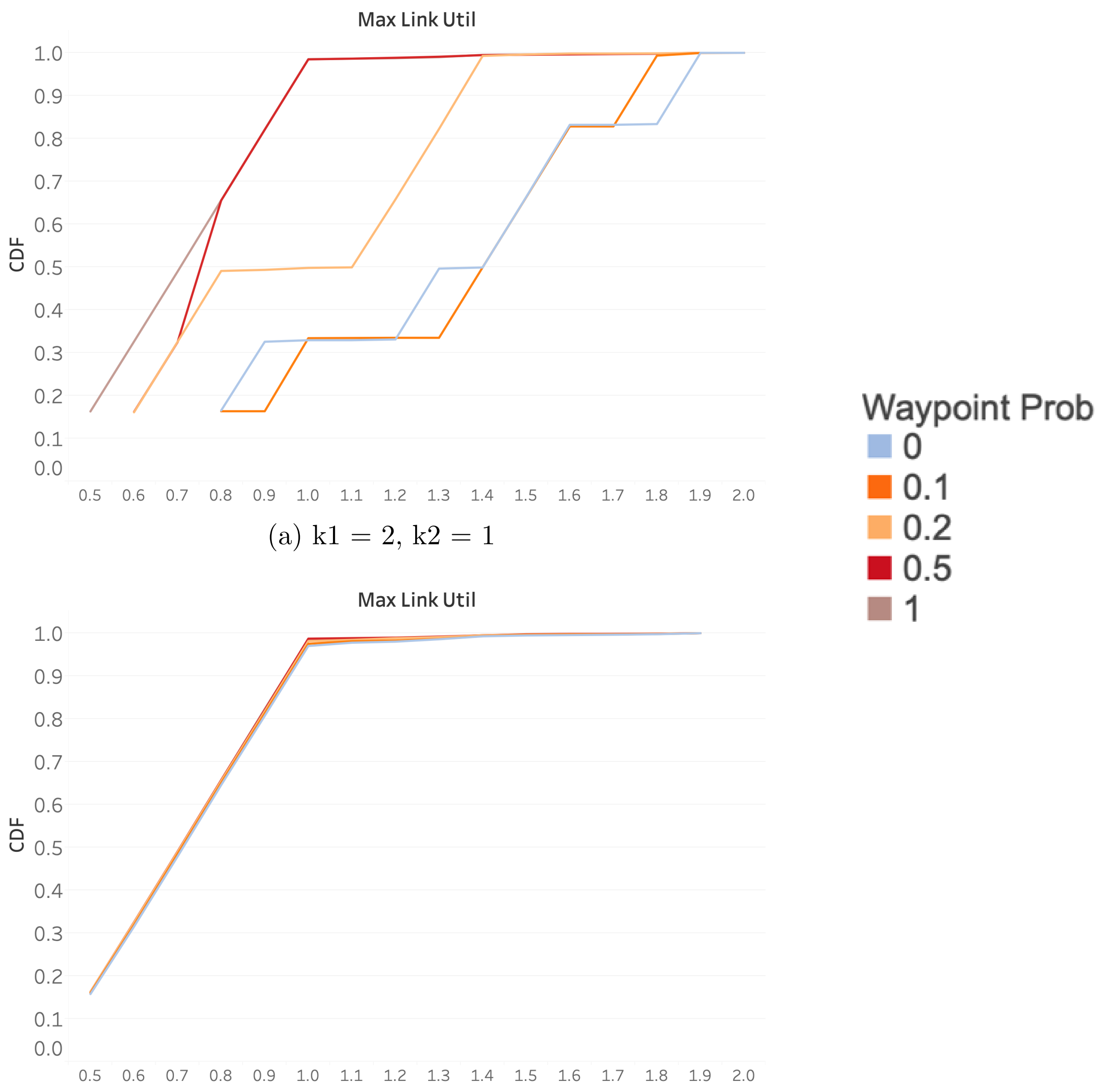
Figure 3.8: big node-to-big node path lengths for varying numbers of waypoints.

3.8, for any given waypoint probability beyond the very lowest, networks with three, 10, and 15 edge switches all produce big node-to-big node paths of similar lengths. This makes sense, because at medium to high values of `waypoint_prob` the number of waypoints is much greater than any of these values.

Reduced Congestion

To quantify the impact of big node-to-big node tunneling on limiting congestion in the presence of link failures, we repeat our TE experiments from Section 3.5.1, separately failing each of the 209 links in the network and calculating the maximum link utilization that results when Red applies its failure recovery protocol. Figures 3.9a and 3.9b show the results of these experiments. These figures are analogous to Figures 3.7a and 3.7b, respectively, except that Figures 3.9a and 3.9b aggregate results across trials that differ not only in the traffic demands they serve but also in which particular link was failed.

Comparing Figures 3.7a and 3.9a, we see that maximum link utilization is nearly identical with and without failures for the extreme values of `waypoint_prob`. The blue line representing the edge-to-edge tunneling case follows almost the same path in Figure 3.9a as



(a) k1 = 2, k2 = 1

(b) k1 = 3, k2 = 3

Figure 3.9: CDF showing fraction of trials for which maximum link utilization is at most a certain value with link failures.

it does in Figure 3.7a, as does the brown line representing the case in which every node is a big node. (Recall that the brown waypoint_prob = 1 line in Figure 3.7a sits directly under the red line.) However, as Figure 3.9a shows, in the presence of failures making half the nodes big nodes is not equivalent, from the perspective of minimizing maximum link utilization, to making all the nodes big nodes; in Figure 3.7a the red line falls to the right of the brown line.

Comparing Figures 3.7b and 3.9b, we see that, when links can fail, big node-to-big node tunneling is at least a little helpful for reducing congestion even when big nodes can split flows across more next-hops; the five lines in Figure 3.9b are not completely on top of each other, although they still overlap to a great extent.

3.6 Explaining the Limited TE Impact of big nodes

At first glance, the results we present in Sections 3.5.1 and 3.5.2 seem counterintuitive. As we explain in the chapter Introduction (Section 3.1), we would expect that the presence of interior waypoints should help limit congestion compared to edge-to-edge tunneling, because the ability of interior waypoints to split flows across multiple next-hops allows for greater path diversity. However, our results are in fact consistent with what other researchers have observed experimentally and proven mathematically ought to be the case.

Fundamentally, big node-to-big node tunneling does not reduce congestion because a minimal-congestion routing does not require a great deal of path diversity; interior waypoints do make multiplicatively more paths available, but these extra paths aren't very helpful for reducing congestion. Liu et al. [44] show that an optimal solution to an MCF routing problem with D demands and L links can have at most L demands using more than one path each; the other $D - L$ demands each require only a single path. Consequently, if D is much larger than L, most flows don't benefit at all from the ability of waypoints to split them across multiple next-hops. In their experiments they often observe that many more than $D - L$ flows use a single path [44]. Liu et al. also explain that, even in the presence of failures, multipath routing is not particularly beneficial. When a link fails the topology now has $L-1$ links, and the specific path each flow takes might change, but the $D+L$ property still holds [44].

Having said all this, we acknowledge that our setting differs slightly from that of Liu et al. Their proof applies to MCF routing, and our routing is a hybrid between edge-to-edge tunneling and MCF. They conclude that the vast majority of flows must each follow a single path *when the number of demands is much larger than the number of links*, but we have more links in our network than demands. Nevertheless, we believe that their mathematical and experimental results come from a setting similar enough to ours that the core ideas translate. Further, they are not alone in reaching these conclusions [57].

3.7 Related Work

Red draws on the well-established traffic engineering and failure recovery techniques of MPLS and segment routing along with a variety of research proposals in the same domain. Red also shares some similarities with prior work on partial deployments of new technologies within existing networks.

3.7.1 TE and Failure Recovery: Established Standards

Tunnel-Based

MPLS MPLS is an edge-to-edge tunneling architecture introduced and standardized in the late 1990s [8, 9, 50, 52]. At the time, longest prefix matching was difficult to implement efficiently, and the goal of MPLS was to remove the need for this complex functionality in the interior of the network [10]. It is still widely used today for routing, TE, priority control, service provisioning, and failure restoration [12]. However, MPLS suffers from the drawbacks of edge-to-edge tunneling outlined above.

Segment Routing On the surface, Red's network operation model seems to be nearly identical to a standardized routing protocol called *segment routing* [23], which is an implementation of a concept called *source routing*. In segment routing, the ingress node specifies all or part of a packet's path through the network by pushing onto it a stack of labels. Each label corresponds to an interior node that the packet must visit, but the packet may also visit some nodes not named in the stack. Each interior switch that the packet visits chooses the next hop based on whichever label is currently at the top of the stack. When the packet reaches this intermediate destination, that switch pops its own label off the stack and forwards the packet toward the node specified by the new top label.

Red is similar to segment routing in that its ingress switches assign each packet a particular interior node that the packet must visit as an intermediate destination on its path through the network. We can view Red's interior big nodes as analogous to the named nodes specified as intermediate destinations in segment routing's label stack. However, big node-to-big node tunneling differs from segment routing in that Red's ingress switches can specify at most one intermediate destination, and segment routing's ingress switches can specify any number. In addition, Red's interior big nodes keep per-flow state, while segment routing's interior nodes do not. Red's big nodes choose each packet's next-hop big node by matching directly on the packet's destination IP address. In contrast, even named nodes in segment routing forward based on labels in the stack pushed by the ingress edge switch.

Therefore, segment routing inherits some of the failure recovery drawbacks of edge-to-edge tunneling. For example, suppose the ingress switch pushes a label stack requiring a packet p to visit nodes (A, B, C, D) en route to its destination. While p is in flight, node D fails. None of A, B, or C is equipped to respond to the change in network state and choose an alternate path to the p's destination; p will be dropped. In contrast, with Red the ingress would specify A as p's the next-hop big node. If the network is still intact when the p reaches C, then C will push a label that the next-hop big node is D. However, if D has failed by this point, C will have been updated with an alternate next-hop big node for p. Hence, p will successfully reach its destination.

Segment routing suffers from several additional drawbacks, distinct from those of edge-to-edge tunneling. First, adding a label stack encoding a packet's entire edge-to-edge path consumes bandwidth and switch processing resources [26]. Second, segment routing requires routers to run an IGP, whereas Red does not.

Hop-by-Hop

The two most common classes of hop-by-hop routing protocols are *link state* and *distance vector*. In link state protocols such as Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS), each router broadcasts the identities and costs of all of its incident links to every other router. In this way, each router constructs its own complete map of the network and computes for itself the optimal next-hops to use for each destination.

In distance vector protocols such as Routing Information Protocol (RIP), each router sends to its neighbors a list of estimates of the costs of traveling from itself to all other nodes in the network. Each router decides which of its neighbors is the better next-hop for each destination by adding the cost of traveling from itself to said neighbor to the cost from that neighbor to the destination. Thus, no router gets a complete map of the network and instead makes its decisions based on local information.

OSPF and IS-IS are generally used in Tier-1 ISP backbones, while RIP is generally used in regional and local ISP backbones and in enterprise networks [42].

3.7.2 TE and Failure Recovery: Research Proposals

Tunnel-Based

Research proposals advocate for edge-to-edge tunneling in a variety of settings [15, 22, 31, 36, 43, 54], including data centers [7], enterprise networks, private backbones [25, 33, 35], and ISP backbones [61]. Our concept of local renormalizing across remaining live paths is based on Suchara et al.'s state independent splitting [54].

3.7.3 Incremental Deployment

Like Red, the works by Agarwal et al. [6], Chu et al. [22], and Hong et al. [34] allow network operators to realize the benefits of partially deploying new technology without going through the expensive and logistically difficult process of upgrading their entire networks. However, none of these projects includes the notion of big node-to-big node tunneling.

3.8 Conclusion

Red is a network architecture based on tunneling packets from big node-to-big node, rather than tunneling edge-to-edge or forwarding hop-by-hop. In this chapter we introduce the overlay model that serves as the basis for Red's TE and failure recovery schemes, and then we formulate the TE optimization problem and describe our novel failure recovery protocol. Together, these TE and failure recovery techniques allow us to realize the benefits of big node-to-big node tunneling: cost savings compared to hop-by-hop forwarding, faster failure recovery than edge-to-edge tunneling, and the ability to exercise a wider set of paths than is possible with either existing routing approach.

Chapter 4

Conclusion

Nearly all aspects of modern life, from entertainment to business to medical care, depend on the Internet providing reliable, rapid communication. For this to happen, ISPs must design and operate their backbone networks to minimize congestion, even in the presence of switch or link failures. If ISPs had unlimited budgets, they could achieve this goal by drastically overprovisioning their networks, installing many more switches than they anticipate needing. However, in reality ISPs have limited budgets for equipment, and therefore they strive to design networks that are both sufficiently robust and as inexpensive as possible. Hence, they must solve a complex constrained optimization problem.

In this thesis, we demonstrate that ISPs can improve upon their current approaches to solving this problem by *taking advantage of whatever resources the network has to offer.* In Sox: *S*trategic *O*ptical/IP *X*-Layer Network Design (Chapter 2), these resources are well-functioning tails and regens that have previously been tied to now-failed IP links. In Red: A Communist Approach to Network Operation (Chapter 3), these resources are the matching abilities and forwarding table capacities of interior big nodes. In both projects, we achieve this goal of maximizing the utility of available resources *by making decisions as late as possible.* In Sox, we shift IP link placement from network design to network operation. In Red, we avoid fixing a packet's path through the network at the ingress edge switch, instead allowing interior big nodes to use the best information available at the time to choose the later hops in its path.

4.1 Summary of Contributions

We present two formulations for this network design problem, each targeted at particular circumstances an ISP might encounter. Both are based on extensions to the multicommodity flow optimization framework.

4.1.1 Sox

First, we explain how ISPs can take advantage of new optical technology, specifically CD ROADMs, to reuse tails and regens across failure scenarios (Chapter 2). In this way, the ISP avoids permanently tying these components to a particular IP link and wasting these resources when the IP link goes down due to the failure of another tail, regen, or fiber segment. We show that CD ROADMs fundamentally change the nature of the network design problem by shifting the boundary between network design and network operation, and we present a formulation for the new optimization problem. We also describe two heuristics for solving this problem in a more time-efficient, scalable way, and we show that one of these, Greedy, is nearly as effective as Optimal at generating robust, inexpensive network designs.

4.1.2 Red

Second, we explain how ISPs can take advantage of a select few "smart" big nodes in the interior of their backbones to reduce the disruptions their networks suffer as a result of switch or link failures (Chapter 3). Most networks today employ one of two routing models, depending on their switches' abilities to match on various header fields and amount of memory available for storing forwarding tables. Specifically, networks whose nodes are all equally "smart" use hop-by-hop forwarding, and those with "smart" big nodes at the edges and "dumb" little nodes in the interior use edge-to-edge tunneling. Hop-by-hop forwarding allows for faster failure recovery than does edge-to-edge tunneling, but to use this model the ISP must pay to upgrade *all* of its switches at once to high-end models, which is both logistically and financially infeasible.

Thus, we propose big node-to-big node tunneling, a hybrid routing model that makes use of whichever interior nodes the ISP chooses to upgrade. big node-to-big node tunneling allows for faster failure recovery than is possible with the edge-to-edge approach, and it allows ISPs cost savings compared to hop-by-hop forwarding.

4.2 Future Work

Our formulations in both Sox and Red take into account a variety of real-world constraints such as the limited REGEN_DIST an optical signal can travel between regens and the varying abilities of switches to match on certain header fields and store forwarding tables of different sizes. However, we also elide many details, all of which we plan to add to our models in the future:

Changing traffic matrices. For the most part, our work assumes a single traffic matrix that "covers" all others. We have explained at a high level how to explicitly model changing traffic matrices, but we have yet to implement this version of the optimization and evaluate its benefits compared to our current approach.

Multiple classes of traffic. As we mention in 2.2.3, ISPs generally carry multiple classes of traffic, each of which is governed by a different SLA. We don't yet account for these differences in our mathematical formulations.

Specific latency requirements. SLAs sometimes include specific restrictions on the time it will take for packets to reach their destinations. Our optimizations both indirectly work toward this goal by limiting the disruption caused by failures, but neither incorporates latency as first class constraints.

More granular models of switch capabilities. In Red, we divide interior nodes into two categories, big nodes and little nodes, depending on their matching abilities and memory capacities. However, we don't account for the specific header fields each switch can match on, and we don't count a specific number of rules that each switch can store. In addition, these are not the only two axes along which switches differ. For example, switches support different sets of established protocols, and some allow the ISP to implement custom protocols by offering protocol-independent packet processing (P4) [13].

Adding these features to our models will be an important yet challenging direction for future work. The challenge will be to (i) figure out how to incorporate these considerations into our mathematical formulations; and (ii) then make the more complex optimization problems scale to networks of reasonable size. However, we believe these challenges are worth tackling, because these considerations *are* all important aspects of the real-world problem. We need to formulate and evaluate network designs that do explicitly model these features so that we can compare these results to those produced by our existing, simpler models. We may find that, in practice, there is little benefit to using the more complex formulations, or we may find that some of these details *are* crucial.

We also plan to explore how to further speed up the computation time required to solve our existing formulations. We have taken important steps toward this end by presenting some heuristics for both, but we plan to work on developing additional heuristics and methods for parallelizing the computation.

4.2.1 Red + Sox

Thus far, we have developed Sox and Red as independent network design strategies based around common themes yet not explicitly designed to coexist within the same network. They are not incompatible and *could* both be implemented in one backbone, but we have not yet investigated how they can be combined to produce a whole greater than the sum of their parts. We believe great opportunity exists in this direction, given their mathematical and conceptual similarities.

Bibliography

[1] http://www.brocade.com/content/html/en/configuration-guide/FI_08030_SDN/GUID-14B40208-34F5-4A20-B7AA-1131D9592897.html.

[2] https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.simple_paths.shortest_simple_paths.html.

[3] Bidirectional forwarding detection. http://www.cisco.com/c/en/us/td/docs/ios/12_0s/feature/guide/fs_bfd.html.

[4] MPLS limitations on QFX Series and EX4600 switches. http://www.juniper.net/techpubs/en_US/junos/topics/reference/general/mpls-limitations-qfx-series.html.

[5] CS cluster computing, 2019.

[6] Sugam Agarwal, Murali Kodialam, and T.V. Lakshman. Traffic engineering in software defined networks. In *IEEE INFOCOM 2013*, April 2013.

[7] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *USENIX Conference on Networked Systems Design and Implementation*, April 2010.

[8] Alia Atlas, George Swallow, and Ping Pan. Fast reroute extensions to RSVP-TE for LSP tunnels. IETF RFC 4090, May 2005.

[9] Daniel Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. IETF RFC 2702, September 1999.

[10] Daniel O. Awduche. MPLS and traffic engineering in IP networks. *IEEE Communications Magazine*, 37, 1999.

[11] Balagangadhar G. Bathula, Rakesh K. Sinha, Angela L. Chiu, Mark D. Feuer, Guangzhi Li, Sheryl L. Woodward, Weiyi Zhang, Robert Doverspike, Peter Magill, and Keren Bergman. Constraint routing and regenerator site concentration in roadm networks. *Journal of Optical Communications and Networking*, 5(11), November 2013.

[12] Martin Birk, Gagan Choudhury, Bruce Cortez, Alvin Goddard, Narayan Padi, Aswatnarayan Raghuram, Kathy Tse, Simon Tse, Andrew Wallace, and Kang Xi. Evolving to an SDN-enabled ISP backbone: Key technologies and applications. *IEEE Communications Magazine*, 54, 2016.

[13] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Computer Communication Review*, 44(3):87–95, July 2014.

[14] Andrew Brzezinski and Eytan Modiano. Dynamic reconfiguration and routing algorithms for IP-over-WDM networks with stochastic traffic. In *IEEE INFOCOM 2005*, March 2005.

[15] Martín Casado, Teemu Koponen, Scott Shenker, and Amin Tootoonchian. Fabric: A retrospective on evolving SDN. In *Workshop on Hot Topics in Software Defined Networking*, August 2012.

[16] Yiyang Chang, Sanjay Rao, and Mohit Tawarmalani. Robust validation of network designs under uncertain demands and failures. In *USENIX Conference on Networked Systems Design and Implementation*, March 2017.

[17] Angela Chiu, Gagan Choudhury, Robert Doverspike, and Guangzi Li. Restoration design in IP over reconfigurable all-optical networks. In *IFIP International Conference on Network and Parallel Computing*, NPC, September 2007.

[18] Angela Chiu and John Strand. Joint IP/optical layer restoration after a router failure. In *Optical Fiber Communication Conference and Exposition*, March 2001.

[19] Angela L. Chiu, Gagan Choudhury, George Clapp, Robert Doverspike, Mark Feuer, Joel W. Gannett, Janet Jackel, Gi Tae Kim, John G. Klincewicz, Taek Jin Kwon, Guangzhi Li, Peter Magill, Jane M. Simmons, Ronald A. Skoog, John Strand, Ann Von Lehmen, Brian J. Wilson, Sheryl L. Woodward, and Dahai Xu. Architectures and protocols for capacity efficient, highly dynamic and highly resilient core networks. *Journal of Optical Communications and Networking*, 4(1), January 2012.

[20] Gagan Choudhury, Martin Birk, Bruce Cortez, Alvin Goddard, Narayan Padi, Kathy Meier-Hellstern, John Paggi, Aswatnarayan Raghuram, Kathy Tse, Simon Tse, and Andrew Wallace. Software defined networks to greatly improve the efficiency and flexibility of packet IP and optical networks. In *International Conference on Computing, Networking, and Communications*, January 2017.

[21] Gagan Choudhury, David Lynch, Gaurav Thakur, and Simon Tse. Two use cases of machine learning for SDN-enabled IP/optical networks: Traffic matrix prediction and optical path performance prediction. *Journal of Optical Communications and Networking*, 10(10), October 2018.

[22] Cing-Yu Chu, Kang Xi, Min Luo, and H. Jonathan Chao. Congestion-aware single link failure recovery in hybrid SDN networks. In *IEEE INFOCOM 2015*, April 2015.

[23] Clarence Filsfils, Stefano Previdi, Les Ginsberg, Bruno Decraene, Stephane Litkowski, and Rob Shakir. Segment routing architecture. RFC 8402, July 2018.

[24] Ori Gerstel, Clarence Filsfils, Thomas Telkamp, Matthias Gunkel, Martin Horneffer, Victor Lopez, and Arturo Mayoral. Multi-layer capacity planning for IP-optical networks. *IEEE Communications Magazine*, 52, January 2014.

[25] Amitabha Ghosh, Sangtae Ha, Edward Crabbe, and Jennifer Rexford. Scalable multi-class traffic management in data center backbone networks. *IEEE Journal on Selected Areas in Communications*, 31(12), 2013.

[26] Alessio Giorgetti, Piero Castoldi, Filippo Cugini, Jeroen Nijhof, Francesco Lazzeri, and Gianmarco Bruno. Path encoding in segment routing. In *IEEE GLOBECOM 2015*, December 2015.

[27] Jennifer Gossels, Gagan Choudhury, and Jennifer Rexford. Robust network design for IP/optical backbones. *Journal of Optical Communications and Networking*, 11, August 2019.

[28] Inc. Gurobi Optimization. Mixed-integer programming (MIP) – a primer on the basics, 2019.

[29] Grani A. Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. k-adaptability in two-stage robust binary programming. *Operations Research*, 63(4), 2015.

[30] Renaud Hartert, Stefano Vissicchio, Pierre Schaus, Olivier Bonaventure, Clarence Filsfils, Thomas Telkamp, and Pierre Francois. A declarative and expressive approach to control forwarding paths in carrier-grade networks. In *ACM SIGCOMM*, August 2015.

[31] Victor Heorhiadi, Michael K. Reiter, and Vyas Sekar. Accelerating the development of software-defined network optimization applications using SOL. *Computing Research Repository*, abs/1504.07704, 2015.

[32] Victor Heorhiadi, Michael K. Reiter, and Vyas Sekar. Simplifying software-defined network optimization using SOL. In *USENIX Conference on Networked Systems Design and Implementation*, March 2016.

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

[34] David Ke Hong, Yadi Ma, Sujata Benerjee, and Z. Morley Mao. Incremental deployment of SDN in hybrid enterprise and ISP networks. In *ACM SIGCOMM Symposium on SDN Research*, March 2016.

[35] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a globally-deployed software defined WAN. In *ACM SIGCOMM*, August 2013.

[36] Umar Javed, Martin Suchara, Jiayue He, and Jennifer Rexford. Multipath protocol for delay-sensitive traffic. In *International Conference on COMmunication Systems And NETworks*, January 2009.

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

[38] Nanxi Kang, Monia Ghobadi, John Reumann, Alexander Shraer, and Jennifer Rexford. Efficient traffic splitting on commodity switches. In *ACM Conference on Emerging Networking Experiments and Technologies*, December 2015.

[39] D. Katz and D. Ward. Bidirectional forwarding detection (BFD). IETF RFC 5880, June 2010.

[40] D. Katz and D. Ward. Bidirectional forwarding detection (BFD) for IPv4 and IPv6 (single hop). IETF RFC 5881, June 2010.

[41] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert D. Kleinberg, and Robert Soulé. Kulfi: Robust traffic engineering using semi-oblivious routing. *Computing Research Repository*, abs/1603.01203, 2016.

[42] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Addison-Wesley Publishing Company, 6th edition, 2013.

[43] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. Traffic engineering with forward fault correction. In *ACM SIGCOMM*, August 2014.

[44] Xuan Liu, Sudhir Mohanraj, Michal Pióro, and Deep Medhi. Multipath routing from a traffic engineering perspective: How beneficial is it? In *22nd International Conference on Network Protocols*, October 2014.

[45] Scott Moritz. Verizon's CEO is open to possible mergers – maybe with Comcast or Disney, April 2017. `http://www.chicagotribune.com/bluesky/technology/ct-verizon-comcast-disney-deals-20170418-story.html`.

[46] Panos Papanikolaou, Konstantinos Christodoulopoulos, and Emmanouel Varvarigos. Incremental planning of multi-layer elastic optical networks. In *International Conference on Optical Network Design and Modeling*, May 2017.

[47] Panos Papanikolaou, Konstantinos Christodoulopoulos, and Emmanouel Varvarigos. Joint multi-layer survivability techniques for IP-over-elastic-optical-networks. *Journal of Optical Communications and Networking*, 9, January 2017.

[48] Panos Papanikolaou, Konstantinos Christodoulopoulos, and Manos Varvarigos. Optimization techniques for incremental planning of multilayer elastic optical networks. *Journal of Optical Communications and Networking*, 10, March 2018.

[49] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, 5th edition, 2011.

[50] Eric Rosen and Ross Callon. Multiprotocol label switching architecture. IETF RFC 3031, March 2013.

[51] M. Shand and S. Bryant. IP fast reroute framework. IETF RFC 5714, January 2010.

[52] Vishal Sharma and Fiffi Hellstrand. Framework for multi-protocol label switching (MPLS)-based recovery. IETF RFC 3469, February 2003.

[53] Robert Soulé, Shrutarshi Basu, Parisa Jalili Marandi, Fernando Pedone, Robert D. Kleinberg, Emin Gün Sirer, and Nate Foster. Merlin: A language for provisioning network resources. In *ACM Conference on Emerging Networking Experiments and Technologies*, December 2014.

[54] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. Network architecture for joint failure recovery and traffic engineering. In *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2011.

[55] Simon Tse and Gagan Choudhury. Real-time traffic management in AT&T's SDN-enabled core IP/optical network. In *Optical Fiber Communication Conference and Exposition*, March 2018.

[56] I-Lin Wang. Multicommodity network flows: A survey, part I: Applications and formulations. 15, December 2018.

[57] Meng Wang, Chee Wei Tan, Weiyu Xu, and Ao Tang. Cost of not splitting in routing: Characterization and estimation. *IEEE/ACM Transactions on Networking*, 19(6), December 2011.

[58] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. R3: Resilient routing reconfiguration. In *ACM SIGCOMM*, August 2010.

[59] Dahai Xu, Mung Chiang, and Jennifer Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Transactions on Networking*, 19(6), December 2011.

[60] Seisho Yasukawa, Adrian Farrel, and Olufemi Komolafe. An analysis of scaling issues in MPLS-TE core networks. IETF RFC 5439, February 2009.

[61] Jiaqi Zheng, Hong Xu, Xiaojun Zhu, Guihai Chen, and Yanhui Geng. We've got you covered: Failure recovery with backup tunnels in traffic engineering. In *IEEE International Conference on Network Protocols*, November 2016.