RESILIENT WIDE AREA NETWORK ROUTING ALGORITHMS FOR MEETING SERVICE LEVEL OBJECTIVES

by

Chuan Jiang

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Electrical and Computer Engineering

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Sanjay G. Rao, Chair

School of Electrical and Computer Engineering

Dr. Mohit Tawarmalani

Krannert School of Management

Dr. Xiaokang Qiu

School of Electrical and Computer Engineering

Dr. Xiaojun Lin

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

To my family.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Sanjay Rao, for all the help, advice, support and opportunities I received during the last six years. Professor Sanjay Rao not only provided me guidance and resources in conducting research, but also taught me a lot of valuable lessons beyond research.

I would also like to thank my committee members, Professor Mohit Tawarmalani, Professor Xiaokang Qiu and Professor Xiaojun Lin for their valuable feedback and questions on my dissertation.

I am grateful to have worked with my research collaborators, Ashish Chandra, Ashiwan Sivakumar, Yanjun Wang, Yiyang Chang, Zixuan Li, Professor Mohit Tawarmalani and Professor Xiaokang Qiu. Their hard work greatly helped my research.

I am also fortunate to have worked with my colleagues at Internet Systems Lab: Ashiwan, Chandan, Ehab, Russ, Usman, Yiyang, Yun and Zixuan. Discussing research with them is always enlightening. They also offered me lots of support.

Lastly, I would like to thank Graduate School of Purdue University for awarding me Anthony T. C. Gaw Fellowship; and I would like to thank NSF for funding my research.

TABLE OF CONTENTS

LIST OF TABLES 9

LIST OF FIGURES 10

ABSTRACT 12

1 INTRODUCTION 13

1.1 Background 13

1.2 Recent works and their limitations 15

1.3 Contributions 15

1.3.1 PCF 16

1.3.2 Lancet 16

1.3.3 Flexile 17

1.4 Thesis Organization 18

2 PCF: RESILIENT FLEXIBLE ROUTING WITH ROBUST THROUGHPUT GUARANTEES 19

2.1 Motivation 20

2.2 PCF overview 23

2.2.1 Notation and preliminaries 24

2.2.2 Modeling network structure 25

2.2.3 Modeling more flexible response 29

2.2.4 Conditional Logical Sequences 33

2.2.5 PCF generalizations . . . 35
2.3 Realizing PCF's mechanisms . . . 37
2.3.1 Realizing general logical sequences . . . 38
2.3.2 Topologically sorted logical sequences . . . 43
2.3.3 Implementation and deployment pathways . . . 45
2.4 Evaluations . . . 46
2.4.1 Results . . . 49
2.4.2 Feasibility of local yet optimal routing . . . 52
2.4.3 Tractability of formulations . . . 53
2.5 Related work . . . 53
2.6 Conclusions . . . 56
3 LANCET: PROTECTION ROUTING DESIGNED FOR PERCENTAGE OF SCENARIOS . . . 57
3.1 Generalized protection routing model . . . 57
3.1.1 Protection routing definition . . . 57
3.1.2 Offline protection routing design . . . 59
3.1.3 Design with multiple traffic classes . . . 61
3.2 Efficient online adjustment of protection routing parameters . . . 62
3.3 Design percentage of scenarios . . . 66
3.3.1 Lancet system . . . 67

3.3.2 Lancet with multiple traffic classes model 67
3.4 Conclusions 68
4 FLEXILE: MINIMAL FLOW LOSS ALMOST ALWAYS 70
4.1 Motivation 71
4.1.1 Background 71
4.1.2 Example Motivating Flexile 73
4.1.3 Discussion 75
4.2 Flexile design 77
4.2.1 Optimizing flow loss percentiles 78
4.2.2 Efficiently finding critical scenarios 81
4.2.3 Critical flow-aware online allocation 87
4.2.4 Generalizations 89
4.3 Flexile Vs. Teavar 90
4.4 Evaluations 93
4.4.1 Comparisons on emulation testbed 96
4.4.2 Comparisons across topologies 97
4.4.3 Does Flexile increase loss in scenarios? 99
4.4.4 Evaluating other aspects of Flexile 101
4.5 Related work 103
4.6 Conclusions 104

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS 105

5.1 Conclusion 105

5.2 Future research directions 106

REFERENCES 108

VITA 113

LIST OF TABLES

2.1 Throughput of different schemes for the topology in Fig 2.4 under 2 simultaneous link failures. 33

2.2 Topologies used in evaluation (PCF). 48

3.1 Symbol table. 58

4.1 Notation. 80

4.2 Topologies used in evaluation (Flexile). 94

LIST OF FIGURES

2.1 (a) Example to illustrate FFC's coarse modeling of network structure (b) Example to illustrate how tunnel-based reservations can be inefficient 21

2.2 Throughput guarantee with FFC for different tunnel choices compared to the optimal. 22

2.3 A topology with m nodes generalized from the previous example. 29

2.4 Illustrating conditional logical sequences 33

2.5 Realizing PCF in practice. (a) Example abstract model; (b) a practical realization using only tunnels applicable for arbitrary LSs (§2.3.1); (c) an alternate realization when LSs can be topologically sorted (§2.3.2). 38

2.6 Reservation matrix associated with Fig. 2.5. 39

2.7 (a) Impact of adding tunnels to FFC's performance (b) Benefits of PCF across multiple demands for Deltacom 47

2.8 Performance of PCF-TF and FFC when more tunnels are added 48

2.9 (a) PCF vs FFC across topologies. (b) Performance under three simultaneous failures. (c) Reduction in throughput overhead compared to FFC 49

2.10 Solving time for PCF's schemes and optimal. 52

3.1 Illustrating protection routing. 59

3.2 Efficacy of Lancet aided designs with two traffic classes. 68

4.1 Illustrating Flexile's opportunity. Flow 1 can be fully sent over link A-B 99% of time. Flow 2 can be fully sent over link A-C 99% of time. 71

4.2 Bandwidth objectives cannot be met for topology in Fig.4.1 by existing TE schemes. 71

4.3 Critical scenarios for Fig. 4.1. 72

4.4 CDF of 99.9%ile loss across flows for IBM topology. 72

4.5 Increase in *ScenLoss* relative to ScenBest (optimal). 72

4.6 In Teavar's design, f_1 and f_2 are both split equally among 2 paths. 74

4.7 ScenBest can meet objectives in above topology but not in Fig.4.1 which has an additional link. 76

4.8 Example topology to illustrate unfairness with max-min across scenarios. While the network can meet the 99% requirement for both flows, max-min meets the requirements for f_2 but not f_1. 76

4.9 Meeting bandwidth requirements requires computing the β^{th} percentile of flow losses. 79

4.10 Systematic decomposition approach. 82

4.11 (a) Emulation testbed results. (a) Flexile vs. SWAN. (b) Flexile vs. Teavar and SMORE. (c) Comparing flow losses across scenarios in emulations with model predicted losses. 93

4.12 Flexile Vs. SWAN. Flexile matched optimal whenever it was computable. Vertically aligned dots correspond to the same topology. 97

4.13 Flexile Vs. Teavar and our CVaR variants. 99

4.14 Flexile Vs. SMORE and Teavar in richly connected topologies. TLE indicates Time Limit Exceeded. 100

4.15 Flexile sees modest penalty in *ScenLoss* relative to the optimal. 101

4.16 Performance improvement with each iteration. 101

4.17 Flexile can achieve higher traffic scale. 102

4.18 Reduction in solving time with Flexile 102

ABSTRACT

It is challenging to meet stringent network performance requirements with growing traffic and increasing expectations on higher performance. Downtime caused by failures can cost billions of dollars and cause severe problems. In this thesis, I have explored the problem of how to design network for Service Level Objectives (SLOs) with an emphasis on provable performance guarantees under failures. This thesis not only considers worst-case guarantees, but also considers requirements that must be met a percentage of time given SLOs are typically expressed in this fashion. To tackle the problem, this thesis makes the following contributions: (i) PCF, a novel set of mechanisms which ensure the network is provably congestion-free under failures. PCF outperforms FFC, the state-of-the-art mechanism, by a factor of 1.5X on average across 21 topologies; (ii) key components of Lancet, the first system for designing protection routing schemes that can meet a performance target a desired percentage of time; (iii) Flexile, a system for designing routing that meets the bandwidth requirements of flows for a desired percentile of time. Flexile exploits a key unexplored opportunity that each flow's requirement could be met using a different set of failure states. Our experiments show that Flexile outperforms state-of-the-art schemes including SMORE and Teavar in reducing loss at desired percentiles by 46% or more in the median case.

1. INTRODUCTION

Online and cloud-based services have been more and more prevalent in recent years, and the network performance requirements are becoming increasingly stringent. Users are requiring the network infrastructure to ensure their critical applications to operate with acceptable performance and high availability. Network architects often express these requirements as service level objectives(SLO) [1]), which asks the network to meet a performance metric a desired percentage of the time such as "user i is guaranteed b_{i} network bandwidth at least $\beta\%$ of the time".

To meet SLOs, network architects have to cope with uncertainty such as failures in network operations. Failures have been shown to be common in the context of wide area networks [2]–[4]. The rapid pace of network evolution implies that failure is the norm, and the complexity of failures is on the rise [4]. These trends make the task of ensuring acceptable performance more challenging, and bring more attention to this task.

However, existing approaches to designing networks for failures (i) only focus on availability [5]–[9] resulting in poor performance on failures [4], [10]; (ii) only consider a small number of failure states [11]–[17], and do not scale as the number of possible failure states increase; or (iii) rely on ad-hoc simulation-based testing [18], [19]. In this thesis, we propose to design new routing schemes with tractable models to provide analysis of network performance.

The rest of this chapter is organized as follows: §1.1 illustrates the background in this research area. §1.2 introduces some recent works and their limitations. §1.3 presents the contributions and key results of this thesis. §1.4 provides a road map of the whole thesis.

1.1 Background

Researchers have long recognized the need to quickly adapt to network failures. Thus, several schemes have been developed to achieve this by moving traffic away from a failed network device or link. For instance, in MPLS settings, two classes of adaptation mechanisms have been explored [5]. One is called link-based protection scheme. In a link-based protection scheme, traffic previously on a failed link l is rerouted along pre-computed detour paths that

do not include l. The other is called path-based protection scheme, where backup paths are reserved in advance for traffic from each source s to every destination t, and when a particular path fails (e.g., an underlying link, or a node fails), the traffic is diverted to those back-up paths. Nevertheless, these adaptation schemes usually don't provide enough flexibility in routing and hence are insufficient to prevent congestion (and consequently, packet losses and delays) under failures, which can negatively impact the performance of demanding applications such as online retail, Web search, and video streaming.

There also has been much work on designing networks while considering a small number of possible failures (e.g., single link or node failures) [12]–[17]. Typically, these works involve formulating resilient network design as optimization problems (e.g., to determine the spare capacity to provision to handle failures), enumerating all failure states explicitly. While such an approach is tractable when the number of possible failure states is small, it does not scale well for the SLO requirements involving multiple failures simultaneously [10], [20]. The intractability arises from the fact that naively enumerating failure states leads to a formulation that simultaneously models exponentially many routing problems.

The state-of-practice in checking whether networks conform to SLOs involves simulation-based testing [18], [19]. Unfortunately, the number of scenarios to consider is prohibitively large even for moderate sized networks. For instance, verifying that a network with 200 links performs acceptably under all 3 simultaneous link failures [10], [14], [20] considering all traffic matrices collected at 10 minute intervals over a week's period requires testing over a billion scenarios. Many more tests are required if partial link failures are also considered.

Even if such arduous testing can provide assurance that a given network design complies with a specific SLO, it remains challenging to use such tests for designing networks that meet an SLO requirement [21], or for deciding what SLOs to offer [22]. Given the large space of possible designs and policies, and since exhaustively testing any one of them is prohibitive, architects today use ad-hoc design techniques that may lead to overly conservative solutions, or fall short of meeting performance requirements, and lack provable guarantees.

1.2 Recent works and their limitations

The research community has recently designed traffic engineering mechanisms that proactively ensure that the network is congestion-free (i.e., ensure that no link carries more traffic than its capacity) under typical failure scenarios [10], [20], [23]. Given a set of failures, which can be exponentially large, they provide tractable models which can guarantee a network throughput that can always be served and no link will be congested. These mechanisms typically involve light-weight online operations upon failures. For instance, FFC [10], a representative and state-of-the-art approach, allocates bandwidth to flows so that no congestion occurs when f or fewer links fail. To do so, FFC splits traffic from each ingress to egress along a set of pre-specified tunnels. R3 [20], another congestion-free routing scheme, uses link-based protection routing to ensure no link will be congested under f link failures. However, both schemes can provide conservative performance guarantee and are far off the network's intrinsic capability.

1.3 Contributions

This thesis addresses the problem of how to design network to meet performance requirements under failures. One unique focus of this thesis is to provide provable performance guarantees for network design. It is motivated by several key challenges of the state-of-the-art. First, a lot of the existing approaches focus on the worst-case performance. But these approaches are found to be very conservative. Do there exist ways to develop mechanisms that can provide provable worst-case guarantees yet are not that conservative? Second, in practice, network architects care more about whether the network performance can be met a percentage of time. Designing networks for percentile guarantees is a challenging problem that we seek to address.

To answer the above questions, this thesis (i) develops PCF, a novel routing mechanism which can provide better worst-case guarantee than the state-of-the-art, (ii) contributes to Lancet, one of the first systems for designing routing to meet percentile performance target, and (iii) develops Flexile, a system for designing routing to ensure that each flow can meet its bandwidth requirement a percentage of time. We expand these contributions below.

1.3.1 PCF

In this thesis, we show that the existing congestion-free schemes perform much worse than optimal, and present deeper insights into the underlying reasons. In particular, we show that (i) FFC, the state-of-the-art congestion-free mechanism, is not only conservative, but also its performance can degrade with an increase in the number of tunnels; (ii) the performance of FFC can be arbitrarily worse than optimal, even when exponentially many tunnels are used. We show that these results arise because (i) FFC models network structure in a coarse fashion; and (ii) reservations are tightly coupled to paths. We propose PCF (Provably Congestion-free and resilient Flexible routing), a set of novel mechanisms that ensure the network is provably congestion-free under failures, while performing closer to the network's intrinsic capability. PCF achieves these goals by better modeling network structure, and through more flexible response strategies. The key challenge that PCF addresses is how to enhance the flexibility of network response while ensuring that the performance under failures can be tractably modeled.

Results. We compare the performance guarantees provided by PCF's congestion-free mechanisms with FFC. We evaluate our models on multiple real topologies obtained from [24]. The experiments show that across different topologies and traffic matrices, our PCF schemes consistently outperform FFC. On average, our scheme can achieve improvement of more than $1.44X$ over FFC. For GEANT topology, our scheme perform $2.6X$ better. We observe similar results with different number of link failures and different network performance metrics. We also evaluate the tractability of our scheme. For most topologies, the solving times is under 10 seconds. For the two largest networks (Deltacom and Ion) with 302 and 270 sub-links (each original link comprises 2 sub-links), the solving time for is under 100 seconds. This is reasonable because PCF's models only need to be run at the granularity of several minutes.

1.3.2 Lancet

In joint work with another thesis [25], this thesis develops Lancet [26], one of the first works to design networks to meet a percentile target. Lancet is a system for designing link-

based protection routing schemes that can meet a performance target a desired percentage of time. Lancet uses a novel and efficient divide and conquer algorithm to classify scenarios based on performance, and designs routing for the set of scenarios which can be handled by the network. The specific contributions of this thesis to Lancet include (i) showing how to realize a distributed link-based protection routing scheme, (ii) extending the scheme to support multiple traffic classes, and (iii) evaluating the potential to design protection routing schemes with better guarantees.

1.3.3 Flexile

Existing TE schemes including Lancet and Teavar [27] (a scheme to design for percentile requirements developed concurrently with Lancet) are conservative when meeting the percentile bandwidth requirement of each flow. This is because they design for all flows in all scenarios and use the same set of scenarios to meet each flow's percentile requirement. To tackle this, we develop a new system, Flexile, which exploits a key opportunity that each flow could meet its bandwidth requirements over a different set of failure scenarios. Flexile consists of an offline phase and an online phase. In the offline phase, Flexile identifies critical scenarios for each flow to meet its requirement. In the online phase, Flexile uses the information from offline phase to prioritize flows that are critical in the current scenario. A key challenge in developing is that identifying critical scenarios in the offline phase require solving a hard optimization problem at large scale. Flexile tackles this issue by decomposing the original problem and using domain-specific accelerations.

Results. We compare Flexile on percentile metric with Teavar, one of the state-of-the-art work which designs for percentile performence. We also compare Flexile with SMORE, a scheme that optimizes MLU within each failure state, and SWAN, a scheme that can maximize throughput or approximate max-min fairness for multiple traffic classes. Our results over 20 topologies show that Flexile outperforms state-of-the-art TE schemes in reducing flow loss at desired percentiles by 46% or more in the median case while not degrading performance much within a scenario. Our experiments also show that our algorithm for accelerating Flexile can significantly reduce solving time and help Flexile achieve its results

within acceptable computation time. In the largest topology where it can take hours to design without accelerating, our algorithm manages to reduce the solving time to under 100 seconds.

1.4 Thesis Organization

The thesis is organized as follows. Chapter 2 shows that existing congestion-free mechanisms, notably FFC, achieve performance far short of the network's intrinsic capability, and presents PCF, a set of novel congestion-free mechanisms to bridge this gap. We show PCF's effectiveness through formal theoretical results and empirical experiments over multiple Internet topologies. Chapter 3 presents a generalized model of link-based protection routing and shows its extension and implementation. Chapter 4 presents Flexile, a mechanism which directly designs for percentile performance and ensures each flow can sustain as much traffic demand as possible for a given percentage of time. We show that Flexile greatly reduces percentile traffic loss both in simulation and emulation experiments. Finally, Chapter 5 summarizes this thesis and discusses potential future research directions.

2. PCF: RESILIENT FLEXIBLE ROUTING WITH ROBUST THROUGHPUT GUARANTEES

As mentioned, in order to ensure that the network can handle desired demand under certain failure scenarios, the research community has recently designed congestion-free mechanisms. And FFC [10] is considered the state-of-the-art approach. In this chapter, we explore the performance of such congestion-free mechanisms relative to the performance that a network could achieve by responding optimally to each failure. We refer to the performance achieved when the network responds optimally as the intrinsic network capability. We make two **contributions** in this work.

First, we show that congestion-free schemes perform much worse than optimal, and present deeper insights into the underlying reasons. In particular, we show that (i) FFC is not only conservative, but also its performance can *degrade* with an increase in the number of tunnels; and (ii) the performance of FFC can be *arbitrarily worse than optimal*, even when *exponentially many tunnels* are used. We show that these results arise because (i) FFC models network structure in a coarse fashion; and (ii) reservations are tightly coupled to paths, and the failure of a link leads to unutilized capacity on other links in the tunnel that contain the failed link.

Second, we propose PCF (Provably Congestion-free and resilient Flexible routing), a set of novel mechanisms that ensure the network is *provably congestion-free* under failures, while performing closer to the network's intrinsic capability. PCF achieves these goals by better modeling network structure, and through more flexible response strategies. The key challenge that PCF addresses is how to enhance the flexibility of network response while ensuring that the performance under failures can be tractably modeled.

We develop multiple mechanisms as part of PCF that allow the architect to trade-off the achievable performance guarantee with deployment complexity. First, we present an alternate approach for bandwidth allocation with the FFC response mechanism which (i) results in a better performance guarantee; and (ii) ensures the allocation does not degrade with additional tunnels. Second, we explore more flexible network response based on an abstraction that we term **logical sequence (LS)**. A LS from a source to a destination

traverses a series of logical segments (formally defined in §2.2.3). The reservation on any LS for a targeted failure set is guaranteed by the logical segments constituting the sequence. Each segment may recursively route traffic over other LSs or physical tunnels servicing that segment. This allows for significant flexibility in how traffic is routed over various segments, and which nodes respond to a given failure. LSs are loosely inspired by ideas such as segment routing [28], [29] though with significant differences (§4.5). We show that when LSs are used, the performance can be arbitrarily better than FFC. We develop several mechanisms based on LSs, including those that provably out-perform R3 [20], another congestion-free mechanism.

We show how PCF's mechanisms can be implemented in practice. For example, we show that when LSs are chosen with some restrictions, they can be realized by a simple generalization of the local proportional routing scheme used by FFC. When LSs are arbitrarily chosen (which allows for even better performance guarantees), our approach discovers a viable routing using techniques that are lighter weight than the the optimal network response strategy.

Empirical evaluations of PCF over 21 topologies from the Internet Topology Zoo show that PCF significantly out-performs FFC. PCF's schemes can sustain higher throughput than FFC by a factor of 1.11X to 1.5X on average across the topologies, while providing a benefit of 2.6X in some cases.

2.1 Motivation

A critical task for network architects is to ensure that their network designs can sustain desired traffic over a target set of failures [10], [20], [23]. This in turn depends on the mechanisms that the network uses to respond to failures.

To illustrate these issues, consider tunnel-based forwarding [10], [14], [30], where traffic from each ingress to egress is carried over a set of pre-selected tunnels. When a tunnel is no longer available (e.g., due to the failure of an underlying link), then, traffic is redistributed across the surviving tunnels. Redistributing traffic can potentially overload some links. A congestion-free routing mechanism guarantees that the network has been proactively designed so no link would be over-loaded over a desired set of failures [10], [20], [23].

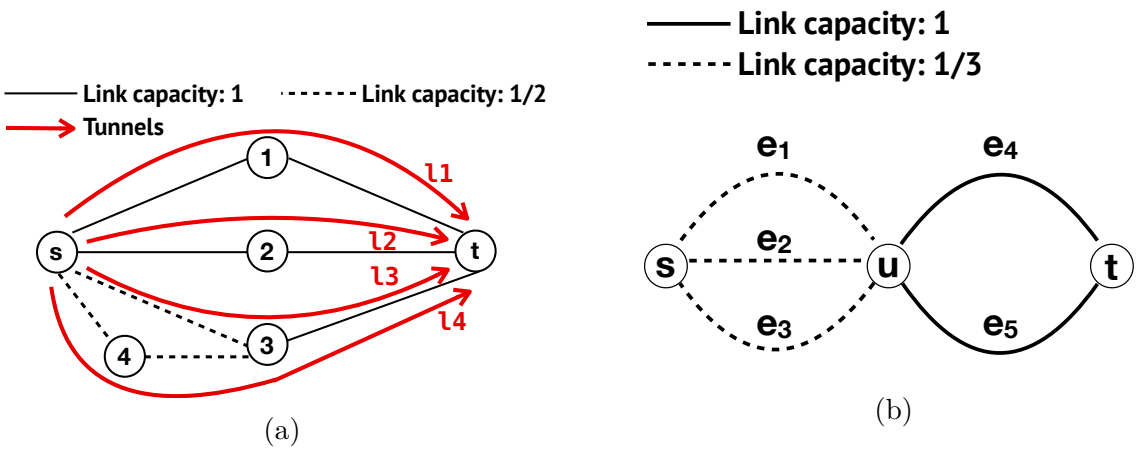


Figure 2.1. (a) Example to illustrate FFC's coarse modeling of network structure (b) Example to illustrate how tunnel-based reservations can be inefficient

FFC [10] is a recent and representative approach set in the context of tunnel-based forwarding. Consider a network where each pair of nodes (s, t) is associated with a traffic demand d_{st}, and a set of tunnels $T(s, t)$ to route the traffic. FFC seeks to assign a bandwidth bw_{st} to each node pair such that this bandwidth can be guaranteed under all possible f simultaneous link failures. To achieve this, FFC reserves bandwidth on each tunnel, and ensures that the total reservation on all tunnels in $T(s, t)$ exceeds bw_{st} under every failure scenario of interest. We present examples to illustrate why FFC is conservative.

Coarse modeling of network structure.

Consider Fig. 2.1a where the goal is to carry the maximum amount of traffic possible from s to t, while tolerating any possible single link failure. If the network could respond optimally for each failure scenario (by running an optimal multi-commodity flow for that scenario), it is easy to verify that the network is intrinsically capable of carrying 2 units of flow from s to t under all possible single link failures. When FFC is used, the results depend on the set of tunnels considered. We consider two schemes: (i) FFC-4 (all 4 tunnels $l1$ to $l4$ are used); and (ii) FFC-3 (only 3 tunnels $l1$ to $l3$ are used). Fig. 2.2 shows that both schemes perform worse than optimal, and surprisingly, FFC-4 performs worse.

We now explain why FFC is *conservative*, and why its performance may *degrade* with *more tunnels*. FFC uses a parameter p_{st} which denotes the maximum number of tunnels

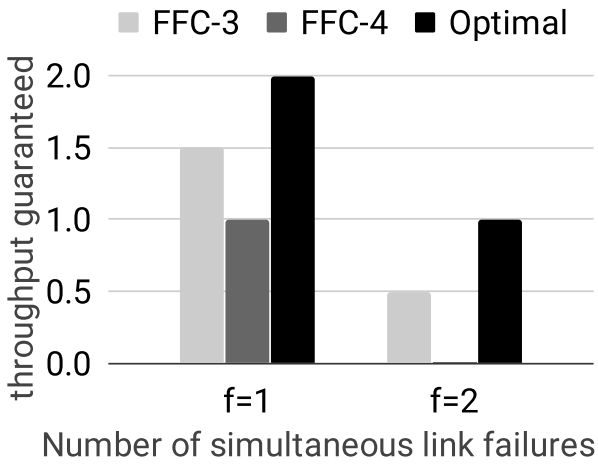


Figure 2.2. Throughput guarantee with FFC for different tunnel choices compared to the optimal.

between s and t that share a common link. When designing to tolerate f link failures, FFC conservatively assumes that upto fp_{st} tunnels may fail, and plans a reservation that can tolerate all possible failures of fp_{st} tunnels. In Fig. 2.1a, when FFC uses all 4 tunnels, p_{st} is 2. Hence, when designing for single link failures, FFC-4 plans for all possible combinations of two tunnel failures. This is conservative because tunnels $l1$ and $l2$ do not fail together under single link failures. With FFC-3, all tunnels are disjoint, and $p_{st} = 1$. Hence, FFC-3 only needs to be consider single tunnel failures. However, FFC-3 still cannot match the optimal since it cannot tap into the capacity of links $s - 4$ and $4 - 3$.

Fig. 2.2 also shows that if all two link failures must be tolerated, the throughput with the optimal, FFC-3, and FFC-4 are 1, 0.5, and 0 respectively. The reasons are similar – FFC-4 can only service traffic that can survive $p_{st}f = 2 \times 2 = 4$ tunnel failures, and hence cannot carry any traffic, while FFC-3 only needs to consider all 2 tunnel failure scenarios.

Limitations of tunnel reservations A second issue with FFC is that it is inherently limited by the fact that reservations are made at the granularity of entire tunnels. To illustrate this, consider Fig. 2.1b. It is easy to verify that if the network responds optimally, it can carry 2/3 units of traffic from s to t under any single link failure. Unfortunately, FFC can only achieve an optimal of 1/2. In §2.2.3, we will further generalize this example to show that FFC can see arbitrarily poor performance relative to optimal.

Tunnel-based allocation does not perform as well as optimal because reservations are made on all links of a tunnel, and when a link fails, the reservations on other links of that tunnel go unutilized. For example, consider a tunnel l that traverses links e_1 and e_4. When e_4 (and hence the tunnel l) fails, FFC only uses the reservations on the remaining tunnels, and the reservation on e_1 for the failed tunnel l goes unutilized. In contrast, the optimal approach is able to use all capacity on all the non-failed links.

In Fig. 2.1b, let T_4 and T_5 respectively denote the set of tunnels from s to t that use e_4 and e_5. Let r_4 and r_5 denote FFC's reservations on each of these sets of tunnels. FFC can carry at most r_5 units of traffic when e_4 fails, and at most r_4 units when e_5 fails. Thus, FFC can guarantee at most $\min(r_4, r_5)$ traffic from s to t over all single link failures. However, $\min\{r_4, r_5\} \times 2 \leq r_4 + r_5 \leq 1$, where the second inequality is because tunnels in $T4$ and $T5$ must reserve capacity in one of the links e1, e2, or e3, whose combined capacity is 1 unit. Hence, FFC can carry at most 0.5 units of traffic from s to t.

2.2 PCF overview

PCF's primary goal is to bridge the gap between existing congestion-free routing mechanisms, and intrinsic network capability. PCF tackles the issues raised in §4.1 by better modeling, and adopting more flexible response strategies.

Unfortunately, not all routing strategies are amenable to formal guarantees on worst-case performance under failures. For instance, when the network responds with an optimal multi-commodity flow (the most flexible response), the problem of determining the worst-case performance under failures is intractable [26]. Thus, a central challenge that PCF tackles is one of carefully crafting response strategies that are (i) amenable to formal worst-case guarantees; and yet (ii) perform closer to the network's intrinsic capability.

PCF achieves the above by (i) developing tractable optimization formulations that are inspired by practical response mechanisms in networking and conservatively estimate network capability; and (ii) providing explicit response mechanisms that achieve the estimated capability. Moreover, PCF allows the network architect to incrementally dial-in additional flexibility in response as desired.

Roadmap. We introduce notation (§2.2.1), and present PCF-TF (§2.2.2), which uses FFC's response mechanism but better models network structure. We show that PCF-TF outperforms FFC, and achieves better performance with more tunnels. Despite these benefits, we show that PCF-TF (like FFC) can perform arbitrarily worse than optimal. We introduce a more flexible approach based on the logical sequence abstraction, and formally show the performance benefits over PCF-TF (§2.2.3). We present further generalizations in §2.2.4, and show how to practically realize the schemes (§2.3).

2.2.1 Notation and preliminaries

Consider a network topology, represented as a graph $G = \langle V, E\rangle$. Each link $\mathrm{e} \in E$ is associated with a link capacity c_{e}. For each node pair (s, t) on the graph, we are given a traffic demand d_{st}, and a set of tunnels $T(s, t)$ to route the traffic. Each tunnel l consists of a set of links $\tau_l \subseteq E$. Below, we present a formulation for bandwidth allocation with tunnels,

$$
\begin{aligned}
(P1)\ \max_{z,a}\quad & \Theta(z) \\
\text{s.t.}\quad & \sum_{l\in T(s,t)} a_l(1-y_l) \geq z_{st}d_{st} \quad \forall s,t\in V, \forall y \in Y && (2.1)\\
& a_l \geq 0 \quad \forall s,t\in V, l\in T(s,t) && (2.2)\\
& \sum_{\forall s,t\in V, l\in T(s,t)} a_l\delta(\mathrm{e}\in\tau_l) \leq c_{\mathrm{e}} \quad \forall \mathrm{e}\in E. && (2.3)
\end{aligned}
$$

Here, $\delta(\mathrm{e} \in \tau_l) = 1$ if $\mathrm{e} \in \tau_l$ and 0 otherwise. The formulation determines a_l and z_{st}, where a_l represents the amount of reservation on tunnel l, and variable z_{st} represents the fraction of traffic from s to t that can be satisfied. Y stands for the set of tunnel failure scenarios of interest, and y_l indicates whether tunnel l fails or not in a failure scenario ($y_l = 1$ indicates tunnel l fails and $y_l = 0$ otherwise.) We later discuss how Y is modeled). $\Theta(z)$ is the metric function we want to optimize. For tractability, we assume $\Theta(z)$ is a concave function, and note that this model covers common metrics such as overall throughput and maximum link utilization. For example, $\Theta(z) = \sum_{s,t} \min\{1, z_{st}\}d_{st}$ models overall throughput. Alternately, when $\Theta(z) = \min_{s,t}\{z_{st}\}$, and the optimal value is Θ^*, the model guarantees that Θ^* fraction of

each flow can be sent in every failure scenario. This also means that using $1/\Theta^*$ of each link's capacity is sufficient to send all the flows. Hence, the inverse of this Θ^* is the utilization of the most congested link, also known as the Maximum Link Utilization (MLU). Thus, $\Theta(z) = \min_{s,t}\{z_{st}\}$ minimizes the MLU.

2.2.2 Modeling network structure

We now discuss how to model the set of failure scenarios Y. If at most p_{st} tunnels between s and t share a common link, FFC assumes that upto fp_{st} tunnels can fail under f link failures, and plans for *all possible combinations* of fp_{st} tunnel failures. As discussed in §4.1, this is conservative – e.g., for the network shown in Fig. 2.1a, FFC considers the simultaneous failure of $l1$ and $l2$ even though this is impossible under single link failure. To address this, PCF more accurately models Y by better relating link and tunnel failures. Let x_e indicate if link e fails ($x_\mathrm{e} = 1$ indicates link e fails and $x_\mathrm{e} = 0$ otherwise). Then, PCF models Y as:

$$\begin{aligned}
&\sum_{\mathrm{e}\in E} x_\mathrm{e} \le f \\
&x_\mathrm{e} - y_l \le 0 \quad \forall l, \mathrm{e} \in \tau_l \\
&y_l - \sum_{\mathrm{e}\in\tau_l} x_\mathrm{e} \le 0 \quad \forall l \\
&0 \le x_\mathrm{e} \le 1 \quad \forall \mathrm{e} \in E \\
&0 \le y_l \le 1 \quad \forall l.
\end{aligned} \tag{2.4}$$

The first constraint bounds the maximum number of simultaneous link failures. The second ensures that the failure of an underlying link will cause the tunnel to fail. The third ensures that a tunnel only fails when at least one underlying link fails. We denote (P1) with Y modeled by (2.4) as PCF-TF. Observe that we do not explicitly impose that $x_\mathrm{e} \in \{0, 1\}$ because, just as for FFC, the failure set Y may contain too many scenarios to enumerate. Instead, we conservatively relax this requirement to $x_\mathrm{e} \in [0, 1]$. Then, the model PCF-TF (and all other models presented in this thesis) can be solved using dualization to ensure the

number of constraints is polynomial in the size of the network, a technique that has been widely used in prior networking papers [10], [20], [31]. We rewrite (2.1) as

$$\min_{y \in Y} \sum_{l \in T(s,t)} a_l(1 - y_l) \geq z_{st} d_{st} \quad \forall s, t \in V. \tag{2.5}$$

These constraints are reformulated by relaxing the integrality of y variables, and expressing the LHS as a maximization problem leveraging LP duality shown below:

$$\begin{aligned}
(D1) \quad & \max_{\pi, \lambda, \sigma, \phi} -(f\lambda_{st} + \sum_{\mathrm{e} \in E} \sigma_{\mathrm{est}} + \sum_{l \in T(s,t)} \phi_l) \\
\text{s.t.} \quad & \boldsymbol{\pi}_l + \phi_l \geq a_l \quad \forall l \in T(s,t) \\
& -\sum_{l : \mathrm{e} \in \tau_l} \boldsymbol{\pi}_l + \lambda_{st} + \sigma_{\mathrm{est}} \geq 0 \quad \forall \mathrm{e} \\
& \boldsymbol{\pi}_l \geq 0 \quad \forall l \in T(s,t) \\
& \lambda_{st} \geq 0 \\
& \sigma_{\mathrm{est}} \geq 0 \quad \forall \mathrm{e} \in E \\
& \phi_l \geq 0 \quad \forall l \in T(x,y).
\end{aligned}$$

Now, we put $(D1)$ into (2.5) and combine it with the rest of constraints in $(P1)$ to obtain the final model below.

$$
\begin{aligned}
(D2) \quad & \max_{\boldsymbol{\pi},\lambda,\sigma,\phi,z,a} \quad \Theta(z) \\
\text{s.t.} \quad & \sum_{l \in T(s,t)} a_l - (f\lambda_{st} + \sum_{\mathrm{e} \in E} \sigma_{est} + \sum_{l \in T(s,t)} \phi_l) \geq z_{st} d_{st} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall s,t \in V \\
& a_l \geq 0 \quad \forall s,t \in V, l \in T(s,t) \\
& \sum_{l:\forall s,t \in V, l \in T(s,t)} a_l \delta(\mathrm{e} \in \tau_l) \leq c_{\mathrm{e}} \quad \forall \mathrm{e} \in E \\
& \boldsymbol{\pi}_l + \phi_l \geq a_l \quad \forall s,t \in V, \forall l \in T(s,t) \\
& -\sum_{l:\mathrm{e} \in \tau_l} \boldsymbol{\pi}_l + \lambda_{st} + \sigma_{est} \geq 0 \quad \forall \mathrm{e}, \forall s,t \in V \\
& \boldsymbol{\pi}_l \geq 0 \quad \forall s,t \in V, \forall l \in T(s,t) \\
& \lambda_{st} \geq 0 \quad \forall s,t \in V \\
& \sigma_{est} \geq 0 \quad \forall s,t \in V, \forall \mathrm{e} \in E \\
& \phi_l \geq 0 \quad \forall s,t \in V, \forall l \in T(s,t).
\end{aligned}
$$

Next, we prove that (i) PCF-TF performs at least as well as FFC; and (ii) unlike FFC, the performance of PCF-TF does not degrade as more tunnels are added.

Proposition 2.2.1. *The feasible region (the set of all possible values of the variables that satisfy the constraints) of FFC is contained in the feasible region of PCF-TF, so PCF-TF performs at least as well as FFC (i.e., achieves the same objective or higher) for any metric.*

Proof. FFC models Y as

$$
\begin{aligned}
& \sum_{l \in T(s,t)} y_l \leq f p_{st} \quad \forall s,t \in V \\
& 0 \leq y_l \leq 1 \quad \forall l.
\end{aligned}
\tag{2.6}
$$

Let Y_0 be the set of tunnel failure scenarios considered by FFC (constrained by (2.6)) and let Y_1 be the set of tunnel failure scenarios considered by PCF-TF (constrained by (2.4)). We show that $\mathrm{proj}_y Y_1 \subseteq Y_0$, where proj_y denotes projection of the set to y variables. For any $s,t \in$

V, we sum the third constraint in (2.4) over all $l \in T(s,t)$ to get $\sum_{l\in T(s,t)}(y_l - \sum_{\mathrm{e}\in\tau_l} x_\mathrm{e}) \leq 0$. Then,

$$\sum_{l\in T(s,t)} y_l \leq \sum_{l\in T(s,t)} \sum_{\mathrm{e}\in\tau_l} x_\mathrm{e} = \sum_{l\in T(s,t)} \sum_{\mathrm{e}\in E} x_\mathrm{e}\delta(\mathrm{e}\in\tau_l)$$
$$= \sum_{\mathrm{e}\in E} x_\mathrm{e} \sum_{l\in T(s,t)} \delta(\mathrm{e}\in\tau_l),$$

$\sum_{\mathrm{e}\in E} x_\mathrm{e}$ is the total number of link failures, which is no more than f. And $\sum_{l\in T(s,t)} \delta(\mathrm{e}\in\tau_l)$ is the number of tunnels from s to t traversing link e, which is no more than p_{st}. Hence, we have $\sum_{l\in T(s,t)} y_l \leq f p_{st}$, which shows that any scenario in Y_1 also satisfies (2.6). Since FFC imposes (1) for each $y \in Y_0$ while PCF-TF imposes (1) for each $(x, y) \in Y_1$, PCF-TF is less constrained than FFC. □

The above proof does not depend on the objective function in the optimization problem, which means that the proposition holds for any metric. We next show that unlike FFC, PCF-TF's performance does not degrade with more tunnels. The intuition behind this is that when more tunnels are added to PCF-TF, the set of constraints that need to be satisfied does not increase. Hence, any solution feasible when fewer tunnels are employed remains feasible when tunnels are added (though new and better solutions may be possible). Thus the performance cannot get worse.

Proposition 2.2.2. *As we provide more tunnels, PCF-TF's performance cannot decrease.*

Proof. Let $\{T_0(s,t) \mid \forall s,t \in V\}$ and $\{T_1(s,t) \mid \forall s,t \in V\}$ be two sets of tunnels, and $T_0(s,t) \subseteq T_1(s,t)$ for all $s,t \in V$. Then, we show that the optimal value for (P1) with $T = T_1$ will not be worse than the optimal solution to (P1) with $T = T_0$. Let (a^*, z^*) be the optimal solution to (P1) with $T = T_0$. We construct (a', z') in the following way,

$$\begin{aligned} a'_l &= a^*_l \quad \forall s,t \in V, l \in T_0(s,t) \\ a'_l &= 0 \quad \forall s,t \in V, l \in T_1(s,t) - T_0(s,t) \\ z'_{st} &= z^*_{st} \quad \forall s,t \in V. \end{aligned} \tag{2.7}$$

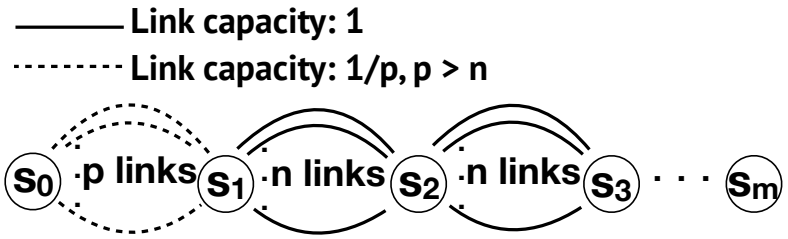


Figure 2.3. A topology with m nodes generalized from the previous example.

Let Y_0 denote (2.4) with $T = T_0$ and Y_1 denote (2.4) with $T = T_1$. It is easy to see that projection of Y_1 onto the space of variables $\{x_e\}_{e\in E}$ and $\{y_l\}_{l\in T_0}$ is contained in Y_0, since all the constraints in Y_0 are present in Y_1. Now for each $y \in Y_1$,

$$\sum_{l\in T_1(s,t)} a'_l(1-y_l) = \sum_{l\in T_0(s,t)} a^*_l(1-y_l) \leq z^*_{st}d_{st} = z_{st}d_{st},$$

where the first equality is because $a'_l = 0$ for $l \notin T_0(s,t)$, the first inequality is because (a^*, z^*) is feasible for $T = T_0$ and the projection of Y_1 is contained in Y_0 and the last equality is by construction. Since z is not altered, the objective value remains the same. □

We later show in §4.4 that PCF-TF performs much better than FFC for real networks.

2.2.3 Modeling more flexible response

While PCF-TF is guaranteed to out-perform FFC, we begin by presenting a theoretical result that shows the performance of PCF-TF can still be arbitrarily worse than optimal because of the inflexibility of tunnel-based reservations. We then discuss PCF's more flexible approach.

Proposition 2.2.3. *The throughput guaranteed by PCF-TF (and hence that guaranteed by FFC) can be arbitrarily worse than the optimal even with exponentially many tunnels.*

Proof. Consider the topology in Fig. 2.3 (the example in Fig. 2.1b is a special case where $p = 3$, $n = 2$ and $m = 2$). Under any failure involving $n-1$ links, the network can carry $1 - \frac{n-1}{p}$ units of traffic if it responded optimally. This is because under any such failure, the network can carry (i) at least 1 unit of traffic between s_i and $s_{i+1}, i > 0$; and (ii) at least

$1 - \frac{n-1}{p}$ units of traffic between s_0 and s_1. Moreover, if $n-1$ of the links between between s_0 and s_1 fail simultaneously, the traffic is no more than $1 - \frac{n-1}{p}$.

Next, consider PCF-TF, and assume that all possible tunnels between s and t are used. There are pn^{m-1} possible tunnels. We will show that PCF-TF can only guarantee traffic of $1/n$ units from s_0 to s_m under $n-1$ simultaneous link failures. To see this, observe that the reservation across all tunnels between s and t is at most 1 (constrained by the capacity of all links between s_0 and s_1). Let r_{i} denote the reservation on all tunnels that use the i^{th} link between s_1 and s_2. Then, $\sum_{\mathrm{i}=1}^{n} r_{\mathrm{i}} \leq 1$, and there must exist at least one link j between s_1 and s_2 such that $r_{\mathrm{j}} \leq 1/n$. Consider a failure scenario where all links between s_1 and s_2 except j fail. Under this scenario, PCF-TF can guarantee at most $1/n$ units of traffic from s_0 to s_m.

Note that $1 - \frac{n-1}{p} - \frac{1}{n} = \frac{(p-n)(n-1)}{pn} > 0$ whenever $p > n > 1$. Consider the case where $p = n^2$. Then, as n gets larger, the amount of traffic carried in the optimal solution converges to 1, while PCF-TF converges to 0. □

As discussed in §4.1, these issues with FFC and PCF-TF stem from the fact that reservations are made over entire tunnels, are tightly coupled to a particular network path, and are pre-allocated independent of any specific failure scenario. When a link in the tunnel fails, the corresponding capacity is unavailable in other links along the tunnel.

Logical sequences. PCF is motivated by the fact that more flexible methods of responding to failures can potentially address the limitations of FFC and PCF-TF highlighted by Proposition 2.2.3. However, even with more flexible response, PCF must proactively decide prior to any failure scenario how much traffic to admit so the network does not experience congestion over a given set of failure scenarios. Not all ways of making routing more flexible are amenable to provable congestion-free guarantees.

Instead, PCF considers a more carefully crafted flexible network response strategy, which we show is amenable to provable guarantees. Specifically, PCF introduces the notion of a *logical sequence (LS)*. A LS q from s to t consists of a series of routers $s, v_1, ..., v_m, t$ that we refer to as logical hops. Consecutive logical hops in a LS need not have a direct link between them, and in fact any pair of routers in the network could be consecutive logical hops. Traffic from s to t is required to traverse the logical hops $v_1, v_2, \ldots, v_m, t$, with significant flexibility

in terms of how traffic is carried between two consecutive logical hops. In particular, traffic may be carried over physical tunnels (like FFC), or other LSs. We refer to each of sv_1, v_1v_2, ..., v_mt as a logical segment of q. Each LS q is associated with a reservation b_q, which indicates that every segment of q (sv_1, v_1v_2, ..., v_mt) is guaranteed to carry b_q traffic under all failure scenarios that PCF is designed for.

We next illustrate the potential benefits of LSs using Fig. 2.3. Consider the LS q which traverses the logical hops $s_0, s_1, ..., s_m$. Let each link be a tunnel. Traffic between consecutive logical hops is carried by the tunnels (links) connecting those hops. For example, traffic between s_1 and s_2 is carried on the n tunnels (links) connecting the nodes. When any link fails, only the reservation in the relevant segment of q is impacted – e.g., if a link between s_1 and s_2 fails, there is no impact on the reservation on the segment between s_0 and s_1. This is unlike FFC and PCF-TF where such a failure would cause part of the capacity on other links to be unavailable. The corollary to Proposition 2.2.3 below captures the resulting benefits.

Corollary 1. *For the topology in Fig. 2.3, PCF's performance with a single LS and polynomially many tunnels can be arbitrarily better than PCF-TF and FFC with exponentially many tunnels.*

Proof. We have already shown that FFC and PCF-TF can be arbitrarily worse than optimal. Consider PCF where LS q corresponding to $s_0, s_1, ..., s_m$ is used, with each link being a tunnel. There are $p + n(m-1)$ tunnels in total. Under any scenario involving $n-1$ simultaneous link failures, the first segment (s_0s_1) has a capacity of at least $1 - \frac{n-1}{p}$ available. All other segments have at least capacity 1 available on any $n-1$ failure scenario. Thus, q can carry at least $1 - \frac{n-1}{p}$ traffic, which meets the optimal throughput. □

We note that using the LS has at least two sources of flexibility beyond classic tunneling. First, in classic tunneling, traffic on each tunnel only carries traffic corresponding to the end points of the tunnel. Second, when there is a failure, only the source node of a tunnel may respond. In contrast, with a LS, each segment may carry traffic corresponding to different sources and destinations - for instance, in Fig. 2.3, the segment (and hence tunnel) between s_1 and s_2 may carry traffic between s_0 and s_m. Further, if the link between s_1 and s_2 fails, s_1 may redistribute the traffic that arrives at s_1 onto the tunnels between s_1 and s_2.

Bandwidth allocation with LSs. We next show that bandwidth allocation with LSs can be tractably formulated. For each pair with source s and destination d, let $L(s,t)$ denote the set of LSs from s to t (with $T(s,t)$ denoting the set of tunnels as before). Note that each source destination pair is associated with zero or more LSs, and zero or more tunnels. Then, the model seeks to reserve b_q on each LS, and reserve a_l on each tunnel l as discussed below:

$$
\begin{aligned}
(P2)\ & \max_{z,a,b} \quad \Theta(z) \\
\text{s.t.}\ & \sum_{l \in T(s,t)} a_l(1-y_l) + \sum_{q \in L(s,t)} b_q \\
& \geq \sum_{q' \in Q(s,t)} b_{q'} + z_{st} d_{st} \quad \forall s,t \in V, \forall y \in Y \qquad (2.8) \\
& b_q \geq 0 \quad \forall s,t \in V, q \in L(s,t) \\
& \text{Constraints (2.2), (2.3).}
\end{aligned}
$$

The most significant change relative to (2.4) pertains to the capacity constraint (first constraint). The LHS of this constraint captures that traffic from s to t could use both the reservations (a_l) on the physical tunnels between s and t, and the reservations (b_q) on the LSs between s and t. While the capacity on tunnel l is only available when all links on the tunnel are alive ($y_l = 0$), the reservation on the LS q is always available (though we relax this requirement in §2.2.4). The RHS of this constraint corresponds to the total traffic that must be carried from s to t. With FFC, this corresponds entirely to the bandwidth allocated to traffic that originates at s, and terminates at t. However, in PCF, it is possible that st is a segment of a LS q' (between a source s' and destination t'). Let $Q(s,t)$ denote the set of all such LSs. Then, the RHS also accounts for reservations on all such $q' \in Q(s,t)$. We refer to (P2) as the **PCF-LS** model. Note that the reservation on a LS is supported by the reservations on physical tunnels and other LSs. The reservations on physical tunnels themselves are supported by the capacity of underlying physical links.

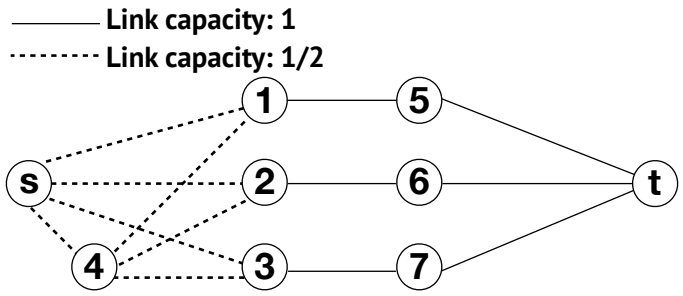


Figure 2.4. Illustrating conditional logical sequences

Table 2.1. Throughput of different schemes for the topology in Fig 2.4 under 2 simultaneous link failures.

Optimal	FFC	PCF-TF	PCF-LS	PCF-CLS	R3
1	0	2/3	4/5	1	0

2.2.4 Conditional Logical Sequences

As described in §2.2.3, each segment in a LS must guarantee the reservation associated with the LS over the entire set of failures. We next consider a generalization, that we call conditional LSs which only guarantee the reservation over a subset of failure scenarios. A conditional LS q is associated with a condition h_q, and a reservation b_q. The reservation b_q must be guaranteed over each segment of q for all scenarios where the condition h_q is met. An example condition is a given set of links being alive or dead.

Illustrating benefits of conditional LSs. We illustrate by considering Fig. 2.4. Table 2.1 shows the traffic guaranteed by different schemes for traffic from source s to destination t under single and two link failures. The table shows both FFC and PCF-TF (both schemes use all 6 tunnels from s to t) are sub-optimal (for the same reasons as (§2.2.2, §2.2.3)).

Consider now that a LS $(s, 4, t)$ is added with logical segments $s4$ (with the tunnel $s-4$), and $4t$ (with multiple tunnels from 4 to t including $4-1-5-t$, $4-2-6-t$, and $4-3-7-t$). Further, the LS is associated with a condition that the reservation is only needed when the link $s-4$ is alive. Table 2.1 shows the optimal is achieved with this conditional LS (PCF-

CLS). Consider two link failure case. When $s-4$ is dead, at most one of the tunnels $s-1-5-t$, $s-2-6-t$, $s-3-7-t$ are dead and the remaining can carry 1 unit of flow. When $s-4$ is alive, at most 2 of these tunnels are dead. Therefore, they can cary 0.5 units of flow. Finally, LS $(s,4,t)$ can carry 0.5 units of flow since $s4$ is alive and at most 2 of the tunnels $4-1-5-t$, $4-2-6-t$ and $4-3-7-t$ are dead.

Note that when the same LS is added but without the attached condition, the objective is not optimal. This is because, the logical segment $s4$ cannot guarantee any reservation over single link failures when only the tunnel $s-4$ is used. It is possible to add more tunnels between s and 4 (e.g., $s-1-4$, $s-2-4$, $s-3-4$), which allows the LS $(s,4,t)$ to be more resilient to failures (PCF-LS). However, this is at the cost of reservations on the tunnels from s to t, and consequently the objective is increased but still does not achieve the optimal.

Modeling conditional LSs. We next discuss how conditional LSs are modeled. Under any given failure scenario, let h_q indicate whether LS q is active or not. Like before, let y_l indicate whether tunnel l fails or not. Let (y,h) denote all y_l and h_q variables, and let YH denote all possible combinations of (y,h) under all scenarios involving the simultaneous failure of f links. To incorporate these conditions, we replace constraint (2.8) in (P2) with the constraint below, and refer to the resulting model as **PCF-CLS**.

$$\sum_{l\in T(s,t)} a_l(1-y_l) + \sum_{q\in L(s,t)} b_q h_q$$
$$\geq \sum_{q'\in Q(s,t)} b_{q'} h_{q'} + z_{st} d_{st} \quad \forall s,t\in V, \forall (y,h)\in YH.$$

In §4.4, we show that LSs activated under a simple condition (a single link being dead) is sufficient to get good performance. To handle this, we model YH by adding constraints $h_q = x_{\mathrm{e}_q}$ for each LS q to (2.4), where e_q is the link whose failure activates LS q. We can also model a more general condition to help generalize PCF to richer failures (e.g., node failures) (§2.2.5). Let h_q be a condition that requires all links in η_q to be alive and all links in ξ_q to be dead. Then we model h_q by linearizing the constraint:

$$h_q = \prod_{\mathrm{e}\in\xi_q} x_{\mathrm{e}} \prod_{\eta_q} (1-x_{\mathrm{e}})$$

as follows:

$$
\begin{aligned}
&(h_q - 1) + x_{\mathrm{e}} \leq 0 \quad \forall \mathrm{e} \in \eta_q \\
&h_q - x_{\mathrm{e}} \leq 0 \quad \forall \mathrm{e} \in \xi_q \\
&(1 - h_q) - \sum_{\mathrm{e} \in \eta_q} x_{\mathrm{e}} - \sum_{\mathrm{e} \in \xi_q} (1 - x_{\mathrm{e}}) \leq 0 \\
&0 \leq h_q \leq 1.
\end{aligned}
$$

2.2.5 PCF generalizations

In this section, we discuss generalizations of PCF, and its relationship with R3 [20], another congestion-free mechanism.

Heuristics for selecting LSs. We present a heuristic for selecting LSs that works well empirically (§4.4). Our approach involves considering a more general model based on flows and decomposing the results of that model into LSs.

We begin by introducing *logical flows*, which are a generalization of LSs in that traffic is no longer constrained to visiting a sequence of hops. A logical flow w from s to t is captured by the flow balance constraints below:

$$
\begin{aligned}
&b_w \geq 0 \quad \forall s, t \in V, \forall w \in W(s,t) \\
&\sum_{\mathrm{j}} p_w(\mathrm{ij}) - \sum_{\mathrm{j}} p_w(\mathrm{ji}) = \begin{cases} b_w & \forall s, t, \mathrm{i} = s, w \in W(s,t) \\ 0 & \forall s, t, \mathrm{i} \neq s, \mathrm{i} \neq t, w \in W(s,t) \\ -b_w & \forall s, t, \mathrm{i} = t, w \in W(s,t) \end{cases} .
\end{aligned} \tag{2.9}
$$

Here, b_w is the reservation associated with the logical flow, and $p_w(\mathrm{ij})$ is the amount of this reservation that must be supported on logical segment ij. Each logical flow w may itself be associated with a condition h_w, which indicates the reservation associated with w is only guaranteed when h_w is satisfied. Let $W(s,t)$ be the set of all logical flows for traffic from s

to t. Then, relative to (P2), the logical flow model involves adding (2.9), and changing (2.8) to

$$\sum_{l \in T(s,t)} a_l(1-y_l) + \sum_{w \in W(s,t)} b_w h_w$$
$$\geq \sum_{s',t' \in V, w' \in W(s',t')} p_{w'}(st) h_{w'} + z_{st} d_{st}$$
$$\forall s,t \in V, \forall (y,h) \in YH.$$

The first term on the RHS captures the reservation that must be supported on (s,t) for any logical flow w' from s' to t'.

To obtain LSs, we decompose the flow into sequences [30], [31]. For each flow $w \in W(s,t)$, this approach generates a derived graph with the same nodes as the original topology. For each node pair (i, j), if $p_w(\text{ij}) > 0$, we add an edge from i to j with the weight being $p_w(\text{ij})$. Then, we search for the widest path from s to t on this graph, and use the sequence of hops in this widest path as a LS with condition h_w.

Relationship to link bypass. While we have focused on tunnel based mechanisms so far [10], we next discuss the relationship of our work to R3 [20], another congestion-free routing scheme. Instead of tunnels, R3 [20] focuses on a link bypass mechanism, where traffic on a link $\text{e} = \langle \text{i}, \text{j} \rangle$ is re-routed upon its failure, along a pre-computed flow from i to j and this flow does not use e.

We first illustrate using Fig. 2.4 that our models can out-perform R3. As Table 2.1 shows, when R3 is applied to Fig. 2.4, no traffic can be carried from s to t if two link failures must be tolerated. To understand why, consider a scenario where links $1-5$, and $5-t$ fail. Since a link bypass for $1-5$ must start at 1 and end at 5, and a link bypass for $5-t$ must start at 5 and end at t, no viable bypass paths exist for either link. Instead, an obvious feasible strategy is to route the traffic along the path $s-2-6-t$, an option that is not considered by R3 because s is not an end point of either $1-5$ or $5-t$.

We now state a more formal result:

Proposition 2.2.4. *A special case of PCF's logical flow model where conditions are restricted to the no failure or single link failure scenarios, and links are tunnels, dominates (performs as well as or better than) R3.*

Proof. To see this, consider the logical flow model under the conditions above. More specifically, for each node pair (s, t), we have a flow w with the condition being no failure and we constrain the flow to exactly serve the demand, i.e., $b_w = z_{st} d_{st}$. For node pair (i, j) which has an edge, we have a flow w with the condition being the link (i, j) being dead. This model is exactly the Generalized-R3 model presented in [26] which has been shown to dominate R3. □

Shared risk links groups (SRLGs) and node failures While we have focused on link failures, a few modifications allow for the treatment of shared risk link groups (SRLGs), and node failures. An SRLG captures that a group of links may fail together (e.g., owing to failure of an underlying optical element) [32]. Each SRLG is modeled by a condition h_q which indicates all links in that SRLG fail. Observe that the first constraint in (2.4) is imposed on x variables that capture link failures. Instead, the constraint can be imposed on conditions dependent on the x variables. For example, a requirement that at most f SRLGs fail is modeled by requiring that $\sum_{q \in Q} h_q \leq f$, where Q is the set of SRLGs. Similarly, the failure of each node is modeled by a condition that all links incident on that router fail. Our discussion and results in §2.2.2 holds for node failures as well - i.e., relative to FFC, PCF-TF performs better, and PCF-TF's performance does not degrade with tunnels. Further, our models do not suffer from the weaknesses of link bypass mechanisms including R3 [20], that cannot deal with node failures (since no viable bypass paths for link $\langle$i, j$\rangle$ from i to j exist when node j itself fails).

2.3 Realizing PCF's mechanisms

In this section, we discuss how to realize PCF's network response mechanisms associated with the models in §2.2.

First, PCF-TF employs the same response mechanism as FFC, which we describe in the rest of this paragraph. Under any failure scenario, traffic across tunnels between a source

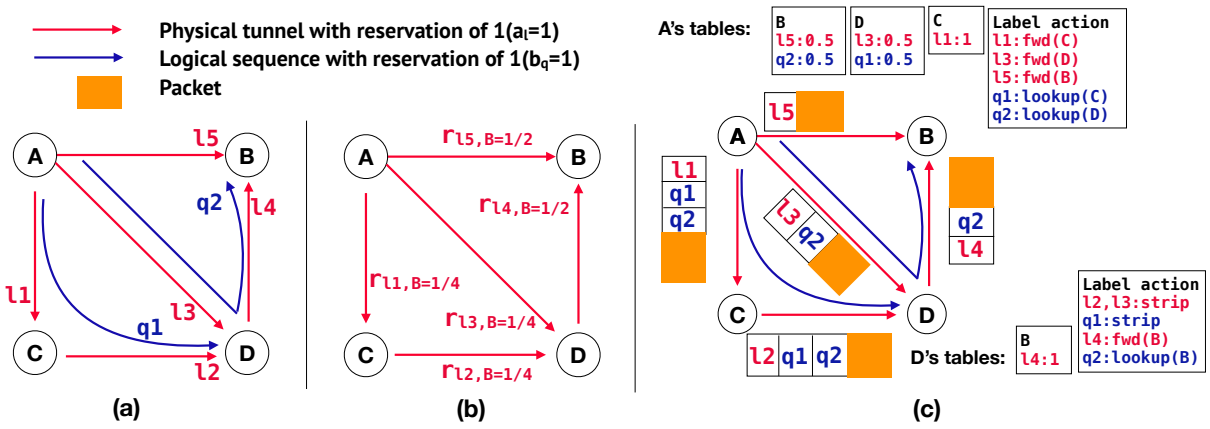


Figure 2.5. Realizing PCF in practice. (a) Example abstract model; (b) a practical realization using only tunnels applicable for arbitrary LSs (§2.3.1); (c) an alternate realization when LSs can be topologically sorted (§2.3.2).

s and destination t is carried on all live tunnels, and in proportion to the reservations on the tunnels. Consider three tunnels from s to t with reservations of $(2, 3, 5)$. When all the tunnels are alive, the (s, t) traffic is split across the tunnels in the ratio $(0.2, 0.3, 0.5)$. If the first tunnels fails, the traffic is sent across the tunnels in the ratio $(0, \frac{0.3}{0.8}, \frac{0.5}{0.8})$.

We next discuss the response mechanisms associated with our models based on LSs. First, we discuss a mechanism that works for arbitrary LSs (§2.3.1). We then show that when LSs are topologically sorted (more formally explained later), a response mechanism similar to FFC may be used (§2.3.2).

2.3.1 Realizing general logical sequences

Consider Fig. 2.5(a) which shows the physical tunnels and the LSs used with our offline PCF-LS, and PCF-CLS models for an example setting (e.g., $l1$ is a physical tunnel between A and C, while $q1$ is a LS between A and D) where traffic is carried from A to B. These models determine the reservations associated with each tunnel, and each LS (e.g., a_{l1} and b_{q1} are respectively the reservation on $l1$ and $q1$).

We discuss an approach to realize this abstract model only using tunnels (in §2.3.2, we discuss an alternate implementation). While in FFC, a tunnel l from i to j may carry traffic

$$M_1 = \begin{array}{c} \begin{matrix} AC & CD & AD & DB & AB \end{matrix} \\ \begin{pmatrix} a_{l_1} & 0 & -b_{q_1} & 0 & 0 \\ 0 & a_{l_2} & -b_{q_1} & 0 & 0 \\ 0 & 0 & a_{l_3} + b_{q_1} & 0 & -b_{q_2} \\ 0 & 0 & 0 & a_{l_4} & -b_{q_2} \\ 0 & 0 & 0 & 0 & a_{l_5} + b_{q_2} \end{pmatrix} \end{array} \begin{matrix} AC \\ CD \\ AD \\ DB \\ AB \end{matrix}$$

Figure 2.6. Reservation matrix associated with Fig. 2.5.

only from i to j, PCF permits some flexibility – e.g., l may carry traffic from s to t if in the abstract model, (i, j) is a segment in a LS from s to t.

Like FFC, our models are run at the granularity of several minutes to periodically recompute reservations (e.g., to handle significant shifts in traffic demands). Once computed for a given traffic matrix, we show that the traffic carried on tunnel l to destination t for any failure scenario may be computed online by solving a system of linear equations, which is much faster than solving linear programs (LPs) such as the multi-commodity flow problem (e.g., a popular approach to solving LPs involves solving many linear systems).

In describing our approach, it is helpful to consider a matrix M that summarizes the reservations. For instance, for the topology in Fig. 2.5, the reservation matrix M is summarized in Fig. 2.6. Each row and column corresponds to a node pair. The diagonal entries indicate the total reservation across all live tunnels and active logical sequences associated with that node pair. A non-diagonal entry in column i and row j indicates that the node pair j must carry traffic corresponding to column i. For instance, in the third row corresponding to the node pair (A, D), the diagonal entry $a_{l3} + b_{q1}$ is the total reservation associated with that node pair (over tunnel $l3$ and LS $q1$). Further, the entry $-b_{q2}$ reflects that (A, D) is a segment of the LS $q2$ from A to B and must be able to carry the reservation b_{q2} associated with $q2$.

A node pair (s, t) is considered to be of interest if it carries positive demand, or if it carries traffic for another node pair of interest. Let P be the set of node pairs of interest. Constraint (2.8) in our LS model can be equivalently expressed in matrix notation as $M \times \vec{1} \geq_v \vec{D}$. Here,

$\vec{1}$ and $\vec{D}$ are $P \times 1$ column vectors. All entries of $\vec{1}$ are 1, while the p^{th} row of $\vec{D}$ has an entry $z_p d_p$ indicating the total traffic associated with pair p that can be carried. Let $\vec{U}$ be a $P \times 1$ column vector. Then, we have:

Proposition 2.3.1. *M is an invertible M-matrix[1], and there is a unique solution $\vec{U^*}$ to the linear system $M \times \vec{U} = \vec{D}$, where $\forall(\mathrm{i},\mathrm{j}) \in P$, $\vec{U^*}(\mathrm{i},\mathrm{j}) \in [0,1]$.*

Proof. We will show $M \in \mathbb{R}^{P \times P}$ is a weakly-chained diagonally dominant matrix, where $M_{\mathrm{ij},\mathrm{i_1j_1}} \leq 0$ for $(\mathrm{i},\mathrm{j}) \neq (\mathrm{i_1},\mathrm{j_1})$. Then, it follows from Theorems 2.1 and 2.2 in [33] that M is an invertible M-Matrix.

For a particular failure scenario x, let $T_x(s,t)$ denote the set of alive tunnels from s to t, $L_x(s,t)$ denote the set of active LSs from s to t and $Q_x(s,t)$ denote the active LSs which go through segment (s,t). We first give the formal definition of P, the set of node pairs of interest. A node pair $(\mathrm{i_1},\mathrm{j_1}) \in P$ if and only if there is a sequence of node pairs $(\mathrm{i_1},\mathrm{j_1})$, $(\mathrm{i_2},\mathrm{j_2})$,...,$(\mathrm{i_k},\mathrm{j_k})$, such that $z_{\mathrm{i_k j_k}} d_{\mathrm{i_k j_k}} > 0$ and $\forall m : 1 \leq m \leq k-1$, $\exists q \in L_x(\mathrm{i}_{m+1},\mathrm{j}_{m+1}) \cap Q_x(\mathrm{i}_m,\mathrm{j}_m)$ such that $b_q > 0$. There is a chain of LSs, such that a preceding LS serves a segment in the subsequent LS, where the last LS serves a pair with non-zero allocation and the first LS contains (i,j). For the node pairs which are not included in P, we set $U(\mathrm{i},\mathrm{j}) = 0$.

Next, we formally define each entry in M. The diagonal of M is the sum of available reservations on the pair, i.e. $\forall(\mathrm{i},\mathrm{j}) \in P, M_{\mathrm{ij,ij}} = \sum_{l \in T_x(\mathrm{i,j})} a_l + \sum_{q \in L_x(\mathrm{i,j})} b_q$. Other entries of M denote how much a node pair needs to carry for other node pairs, i.e. for $(\mathrm{i},\mathrm{j}) \neq (m,n)$ we set $M_{\mathrm{ij},mn} = -\sum_{q \in Q_x(\mathrm{i,j}) \cap L_x(m,n)} b_q$.

It is easy to see that M is weakly diagonally dominated because $M \times \vec{1} \geq \vec{D} \geq 0$, where the first inequality is the capacity constraint and second because $z_p d_p \geq 0$ for all p.

[1]↑A matrix T is an invertible M-matrix if $T_{\mathrm{ij}} \leq 0$ when $\mathrm{i} \neq \mathrm{j}$ and $Tx \geq 0$ implies that $x \geq 0$.

From our definition of P, we know that $\forall(\mathrm{i}_1,\mathrm{j}_1) \in P$, there is a sequence $(\mathrm{i}_1,\mathrm{j}_1)$, $(\mathrm{i}_2,\mathrm{j}_2)$,...,$(\mathrm{i}_k,\mathrm{j}_k)$, such that $z_{\mathrm{i}_k\mathrm{j}_k} d_{\mathrm{i}_k\mathrm{j}_k} > 0$ and $\forall m : 1 \leq m \leq k-1$, $\exists q \in L_x(\mathrm{i}_{m+1},\mathrm{j}_{m+1}) \cap Q_x(\mathrm{i}_m,\mathrm{j}_m) : b_q > 0$. Thus, for each row $(\mathrm{i},\mathrm{j}) \in P$, there is a sequence $(\mathrm{i}_1,\mathrm{j}_1)$, $(\mathrm{i}_2,\mathrm{j}_2)$,...,$(\mathrm{i}_k,\mathrm{j}_k)$, such that

$$\begin{aligned}
&\sum_{(m,n)\in P} M_{\mathrm{i}_k\mathrm{j}_k,mn} \\
&= \sum_{l\in T_x(\mathrm{i}_k,\mathrm{j}_k)} a_l + \sum_{q\in L_x(\mathrm{i}_k,\mathrm{j}_k)} b_q - \sum_{q\in Q_x(\mathrm{i}_k,\mathrm{j}_k)} b_q \\
&\geq z_{\mathrm{i}_k\mathrm{j}_k} d_{\mathrm{i}_k\mathrm{j}_k} > 0,
\end{aligned}$$

and $\forall m : 1 \leq m \leq k-1$, $M_{\mathrm{i}_k\mathrm{j}_k,\mathrm{i}_{k+1}\mathrm{j}_{k+1}} \neq 0$. Therefore, M is a weakly-chained diagonally dominant matrix. Since, for $(\mathrm{i},\mathrm{j}) \neq (\mathrm{i}_1,\mathrm{j}_1)$, $M_{\mathrm{ij},\mathrm{i}_1\mathrm{j}_1} = -\sum_{q\in Q_x(\mathrm{i},\mathrm{j})\cap L_x(\mathrm{i}_1,\mathrm{j}_1)} b_q \leq 0$, it follows that M is an invertible M-matrix and there is a unique solution $\vec{U}^*$ to the linear system $M \times \vec{U} = \vec{D}$.

Next, we use Brouwer fixed-point theorem [34] to prove that all entries of the solution are in $[0,1]$. Let $f(\vec{U})$ be a function mapping from $[0,1]^P$ to $\mathbb{R}^P$. We define $f(\vec{U})$ as

$$f(\vec{U})_{\mathrm{i},\mathrm{j}} = \frac{\vec{D}(\mathrm{i},\mathrm{j}) + \sum_{(m,n)\in P, q\in Q_x(\mathrm{i},\mathrm{j})\cap L_x(m,n)} \vec{U}(m,n) b_q}{\sum_{l\in T_x(\mathrm{i},\mathrm{j})} a_l + \sum_{q\in L_x(\mathrm{i},\mathrm{j})} b_q}. \tag{2.10}$$

Observe that the denominator is larger than zero. If not, it follows from weak diagonal dominance that $M_{\mathrm{ij},\mathrm{i}'\mathrm{j}'} = 0$ for all $(\mathrm{i}',\mathrm{j}') \neq (\mathrm{i},\mathrm{j})$, which contradicts $(\mathrm{i},\mathrm{j}) \in P$. It is easy to see that $\vec{U_0}$ is a solution to $M \times \vec{U} = \vec{D}$ if $f(\vec{U_0}) = \vec{U_0}$. With $\vec{U} \in [0,1]^P$, we have

$$f(\vec{U})_{\mathrm{i},\mathrm{j}} \geq \frac{\vec{D}(\mathrm{i},\mathrm{j})}{\sum_{l\in T_x(\mathrm{i},\mathrm{j})} a_l + \sum_{q\in L_x(\mathrm{i},\mathrm{j})} b_q} \geq 0. \tag{2.11}$$

Moreover,

$$f(\vec{U})_{\mathrm{i},\mathrm{j}} \leq \frac{\vec{D}(\mathrm{i},\mathrm{j}) + \sum_{(m,n)\in P, q\in Q_x(\mathrm{i},\mathrm{j})\cap L_x(m,n)} b_q}{\sum_{l\in T_x(\mathrm{i},\mathrm{j})} a_l + \sum_{q\in L_x(\mathrm{i},\mathrm{j})} b_q} \leq 1, \tag{2.12}$$

where the first inequality is because $\vec{U}(m,n) \leq 1$, $b_q \geq 0$, and the denominator is positive. The second inequality is from the capacity constraint. Since f is a continuous function mapping from $[0,1]^P$ to $[0,1]^P$, and $[0,1]^P$ is a compact convex set, it follows from Brouwer

fixed-point theorem [34] that there is at least one point $U_0 \in [0,1]^P$ so that $f(U_0) = U_0$, which we have already argued is the unique solution to $M \times \vec{U} = \vec{D}$. $\square$

We discuss the implications of Proposition 2.3.1 here. While PCF's models determine the reservations, realizing them in practice requires determining the fraction of the reservation that is actually used in any given failure scenario. The above result indicates that such a fraction exists and may be obtained as a solution to a linear system of equations. While linear systems are already much faster to solve than LPs, the result also indicates that the matrix M is of a type for which simple and memory-efficient iterative algorithms for solving linear systems can be used [35].

For $t \in V$, let $\vec{D}_t$ be a $P \times 1$ column vector where the p^{th} row of $\vec{D}_t$ has an entry $z_p d_p$ if t is an end point of p, and 0 otherwise. Using the same argument as for Proposition 2.3.1, there is a unique solution $\vec{U}_t^*$ to the linear system $M \times \vec{U}_t = \vec{D}_t$. Then, the following holds:

Proposition 2.3.2. *For any live tunnel l from* i *to* j *and any destination t, let $r_{lt} = \vec{U}_t^*(\mathrm{i},\mathrm{j})a_l$ be the total traffic carried to destination t on tunnel l. Then r_{lt} represents a valid routing which carries all the traffic with the destination of t.*

Proof We consider (s,t) column of M^{-1}, which exists by Proposition 2.3.1, and denote it as λ.

By definition, $M = A + B$ where A is a diagonal matrix with $A_{st,st} = \sum_{l \in T_x(s,t)} a_l$, $B_{st,st} = \sum_{q \in L_x(s,t)} b_q$, and for $(s,t) \neq (m,n)$, $B_{st,mn} = -\sum_{(m,n) \in P, q \in Q_x(s,t) \cap L_x(m,n)} b_q$. Then it follows that

$$\sum_{(m,n) \in P} \lambda_{mn} M_{mn,\cdot} = \mathrm{e}_{st}, \tag{2.13}$$

where e_{st} is $(s,t)^{th}$ unit vector in $\mathbb{R}^P$ and $M_{mn,\cdot}$ denotes the column of M corresponding to the pair (m,n). It follows that

$$\sum_{(m,n) \in P} \lambda_{mn} A_{mn,mn} \mathrm{e}_{mn} = \mathrm{e}_{st} - \sum_{(m,n) \in P} \lambda_{mn} B_{mn,\cdot} \tag{2.14}$$

Now, e_{st} can be interpreted as a directed path carrying a unit flow from s to t. Moreover, we show that $B_{mn,\cdot}$ is a circulation since it can be written as an addition of cycles, one for each logical sequence servicing (m, n). To show this, we only need to show that for any $q \in L_x(\mathrm{i}, \mathrm{j})$ with $b_q > 0$, if (k, l) is a logical segment in q, that is if $q \in Q_x(k, l)$, then $(k, l) \in P$. Since $(\mathrm{i}, \mathrm{j}) \in P$, there is a weak chain from (i, j) to a strictly dominated pair. The existence of q shows that (k, l) is connected to (i, j) since $M_{kl,\mathrm{ij}} \leq b_q < 0$. Therefore, $(k, l) \in P$. Thus, the RHS of (2.14) represents a directed path flow e_{st} and some circulations $\sum_{(m,n)\in P} \lambda_{mn} B_{mn,\cdot}$. By the flow decomposition theorem (Theorem 3.5 in [36]), this flow yields a unique arc flow on the tunnel network shipping the same traffic as the directed path e_{st}. The resulting flow is then the LHS of (2.14). In other words, each a_l tunnel connecting (m, n) can use λ_{mn} fraction of its capacity to transmit a unit flow from s to t. Observe that the resulting flow may have loops that could be extracted in post-analysis.

Finally, since we have shown that $M^{-1}\vec{D} = \vec{U}^* \in [0, 1]^P$, it follows that

$$0 \leq A_{mn,mn} \sum_{(s,t)\in P} M^{-1}_{mn,st} z_{st} d_{st} \leq A_{mn,mn}, \tag{2.15}$$

which shows that the accumulated traffic on the tunnels between (m, n) will not exceed the reservation. Thus the traffic is feasible. Observe that all (s, t) pairs with $z_{st}d_{st} > 0$ are included in P. Therefore, $\vec{D}$ contains all the serviced demands and the proof is complete. □ We compute U^*_t for every node t by solving the linear system $M \times [\vec{U^*_{t_1}}, \vec{U^*_{t_2}}, ...\vec{U^*_{t_{|V|}}}] = [\vec{D_{t_1}}, \vec{D_{t_2}}, ...\vec{D_{t_{|V|}}}]$, which in turn allows r_{lt} to be computed. As computed, r_{lt} may have cycles that can be eliminated by subtracting flow associated with the cycle. Fig. 2.5(b) shows a concrete realization of PCF's routing on tunnels for the abstract model shown in Fig. 2.5(a). Each tunnel is annotated with the fraction of the traffic to destination B carried on that tunnel – e.g., $r_{l5,B} = 1/4$ indicates that $l5$ carries $1/4$ of the traffic to B.

2.3.2 Topologically sorted logical sequences

While the approach in §2.3.1 works for *arbitrary* LSs, we next describe an alternate approach that works when LSs are chosen with some restrictions. Given two node pairs

(i, j), and (i′, j′), we say that (i, j) > (i, j′) if (i′, j′) is a segment of any active LS q in L(i, j). Our approach below is applicable if all the node pairs under every failure scenario can be topologically sorted with respect to relation '>'. For example, in Fig. 2.5, the LSs satisfy a topological ordering with $(A, B) > (A, D)$ since $q2 \in L(A, B)$ uses the segment (A, D) (but not vice versa). Note that, essentially, we only require a strict partial order over the node pairs. The topological sort refers to any total order that extends this strict partial order and, it is well-known that such a total order exists and can be derived easily from the partial order[37].

When a topological ordering is possible, PCF implements LSs more directly (Fig. 2.5(c)). When A sends packets to B, traffic is split across the tunnel $l5$ and LS $q2$. Traffic to $q2$ involves pushing a label, and looking up the table entry for host D. This entry indicates traffic is split across tunnel $l3$ and LS $q1$. Traffic to $q1$ involves pushing another label and looking up the entry for host C, which indicates the traffic is to be forwarded on tunnel $l1$. When a router receives a packet, it pops labels as needed, and if it is an intermediate point of a LS takes the appropriate action. For example, when D receives a packet on tunnel $l3$ it pops the outer label $l3$, and based on the inner label $q2$, looks up the entry for B, and forwards to B along tunnel $l4$.

A key question is to decide how to split the traffic at each hop – e.g., for traffic from A to B, what fraction is sent on each of tunnels $l5$, and LS $q2$. We define *local proportional routing* as a scheme where the traffic associated with each node pair (i, j) is split across all tunnels and LSs from i to j in proportion to the reservations associated with these tunnels and LSs. This is a generalization of FFC which uses a locally proportional scheme but in a context where there are only tunnels. Then, the following holds:

Proposition 2.3.3. *The LS models can be realized by local proportional routing when the topological sort property is met.*

Proof. For a particular failure scenario x, let $T_x(s, t)$ denote the set of live tunnels from s to t, $L_x(s, t)$ denote the set of active LSs from s to t and $Q_x(s, t)$ denote the active LSs which go through segment (s, t). We show by induction along the topological sort order that

locally proportional routing services the demand. Our induction hypothesis is that the pair (i,j) needs to route $\tilde{D}_{\mathrm{ij}}$ where,

$$\tilde{D}_{\mathrm{ij}} = \vec{D}(\mathrm{i},\mathrm{j}) + \sum_{(m,n)\in P, q\in Q_x(\mathrm{i},\mathrm{j})\cap L_x(m,n)} u_{mn} b_q, \tag{2.16}$$

if every router distribute $\tilde{D}_{\mathrm{ij}}$ among the tunnels ($l \in T_x(\mathrm{i},\mathrm{j})$) and LSs ($q \in L_x(\mathrm{i},\mathrm{j})$) in the proportion of their reservations, i.e., there is a constant u_{ij} such that traffic along l is $u_{\mathrm{ij}} a_l$ and that along LS q is $u_{\mathrm{ij}} b_q$ where

$$u_{\mathrm{ij}} = \frac{\tilde{D}_{\mathrm{ij}}}{\sum_{l\in T_x(\mathrm{i},\mathrm{j})} a_l + \sum_{q\in L_x(\mathrm{i},\mathrm{j})} b_q}.$$

For the base case, observe that for the topologically largest pair $p_1 = (\mathrm{i}_1,\mathrm{j}_1)$, the demand received is $\vec{D}(\mathrm{i}_1,\mathrm{j}_1)$. And the hypothesis is trivially true because $Q(\mathrm{i}_1,\mathrm{j}_1) = \emptyset$. For the induction step, we assume that the hypothesis is true for pairs $p_1, p_2, ..., p_n$ in topological sort order and show it holds for $p_{n+1} = (\mathrm{i},\mathrm{j})$. Observe that for any $q \in Q_x(\mathrm{i},\mathrm{j}) \cap L_x(m,n)$, the traffic sent to b_q is, by the induction hypothesis, $u_{mn} b_q$ because $Q_x(\mathrm{i},\mathrm{j}) \cap L_x(m,n) = \emptyset$ implies $(m,n) > (\mathrm{i},\mathrm{j})$. Then, (2.16) holds for (i,j), and it follows easily that if (i,j) routes $u_{\mathrm{ij}} b_q$ along each $q \in L_x(\mathrm{i},\mathrm{j})$ and $u_{\mathrm{ij}} a_l$ along each $l \in T_x(\mathrm{i},\mathrm{j})$ that $\sum_{l\in T_x(\mathrm{i},\mathrm{j})} u_{\mathrm{ij}} a_l + \sum_{q\in L_x(\mathrm{i},\mathrm{j})} u_{\mathrm{ij}} b_q = \tilde{D}_{\mathrm{ij}}$. Since it follows easily from above that

$$u_{\mathrm{ij}}\Big(\sum_{l\in T_x(\mathrm{i},\mathrm{j})} a_l + \sum_{q\in L_x(\mathrm{i},\mathrm{j})} b_q\Big) =$$
$$\vec{D}(\mathrm{i},\mathrm{j}) + \sum_{(m,n)\in P, q\in Q_x(\mathrm{i},\mathrm{j})\cap L_x(m,n)} u_{mn} b_q,$$

it follows that $(u_{\mathrm{ij}})_{(\mathrm{i},\mathrm{j})\in P}$ solves $M \times \vec{U} = \vec{D}$. Therefore, by proposition 2.3.1, $0 \le u_{\mathrm{ij}} \le 1$. This implies that the routing is feasible since none of the reservations are exceeded. □

2.3.3 Implementation and deployment pathways

Now, we discuss how our scheme can be implemented and practically deployed. We start with the case when the logical sequences are topologically sorted. The offline computation

phase determines the reservation for each tunnel and LS, similar to how FFC determines tunnel reservations. The regular forwarding operation and the failure recovery is completely distributed. Traffic associated with each node pair (i, j) is split across all physical tunnels and LSs in proportion to the reservations associated with them. When a tunnel fails or an LS is inactive, the weights are rescaled in proportion to the reservations on live tunnels and active LSs. This is similar to the existing approach of rescaling on live tunnels. Recall that LSs may have conditions attached to them and may only be active when the condition is true. Thus, for any conditional LS q from i to j, we need a mechanism to propagate the condition (e.g., link failure event) to i. For concreteness, we focus our discussion on two cases (the only cases considered in our evaluations). The first case involves LSs that do not have any conditions attached. This case is trivial to implement - such LSs are always active, and no hint propagation is needed. The second case involves LSs q between i and j which are only active when the link $i - j$ fails. This can be implemented by having i locally detect the failure of the $i - j$ link, which then results in i activating q and following the standard proportional scheme.

More generally, when logical sequences cannot be sorted in topological order, one simple implementation approach is to use a centralized controller. On each failure, the centralized controller solves a linear system which determines the new routing as discussed in §2.3.1. Solving a linear system is much easier than solving a linear program, as discussed earlier. While we do not explore further, we believe that it is possible to perform the operations on failure in a completely distributed fashion because the linear system we solve is of a special type (see Proposition 2.3.1) for which iterative algorithms exist. We defer further investigation to future work.

2.4 Evaluations

We compare the performance guarantees provided by PCF's congestion-free mechanisms with FFC, the state-of-the-art congestion-free mechanism. When possible, we compare PCF with the performance achieved by the **optimal** network response which involves computing the optimal multi-commodity flow for each failure scenario. We implement all our optimiza-

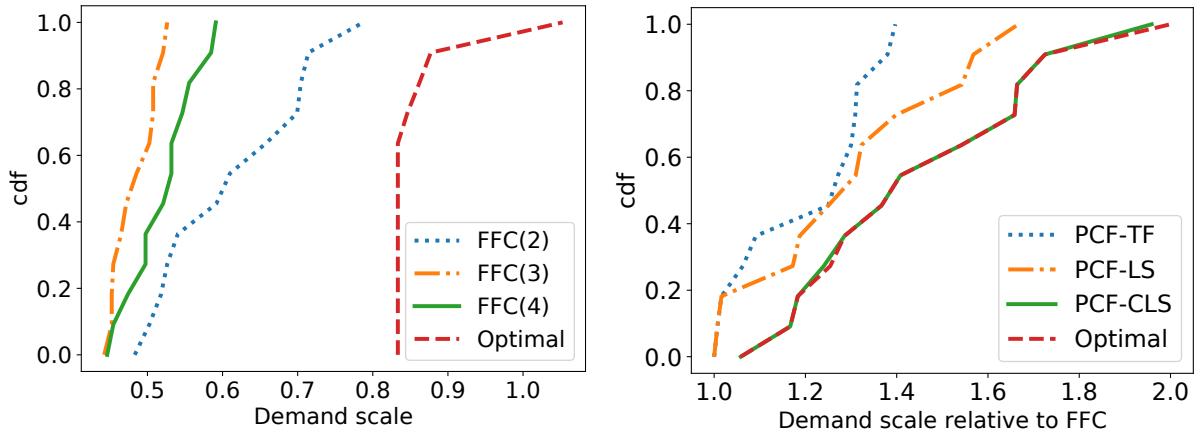


Figure 2.7. (a) Impact of adding tunnels to FFC's performance (b) Benefits of PCF across multiple demands for Deltacom

tion models in Python, and use Gurobi 8.0 [38] to solve them. We consider the following PCF schemes:

- *PCF-TF.* This uses FFC's mechanism to respond to failures, but models network structure more explicitly (§2.2.2).
- *PCF-LS.* Here, LSs are used but not associated with any condition (§2.2.3). For each node pair (s, t), we provide a single LS that includes the set of nodes in the shortest path from s to t. This guarantees that the topological sort assumption is met, which ensures the scheme can be implemented as a locally proportional routing scheme similar to FFC (§2.3.2).
- *PCF-CLS.* Here, the failure of each link $\langle i, j \rangle$ results in the activation of a LS from i to j. Further, each node pair is associated with a LS that is always active. We get these LSs by decomposing a restricted form of the logical flow model, where the only conditions are no link failures, or single link failures, with failure of link $\langle i, j \rangle$ resulting in the activation of a flow from i to j (§2.2.4). The LSs may not be topologically sorted. The scheme can be realized using relatively light-weight operations on each failure compared to the optimal network response (§2.3.1). In §2.4.2 we evaluate a heuristic that derives topologically sorted LSs from the above LSs, which allows for a proportional routing scheme similar to FFC.

Topologies. We evaluate our models on 21 topologies obtained from [24] and [30]. The number of nodes and the number of edges of each topology is shown in Table 2.2. Our

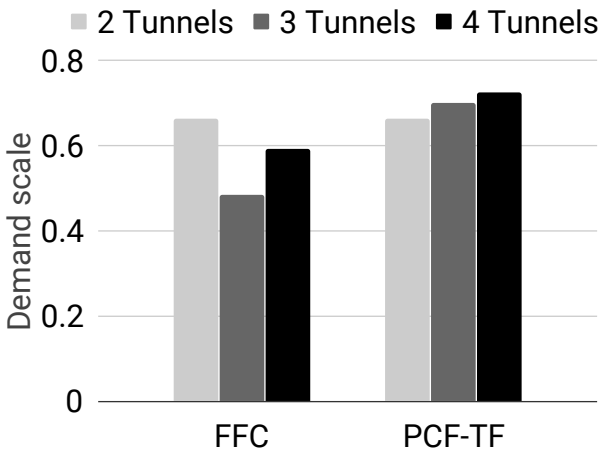


Figure 2.8. Performance of PCF-TF and FFC when more tunnels are added

Table 2.2. Topologies used in evaluation (PCF).

Topology	# nodes	# edges	Topology	# nodes	# edges
B4	12	19	Janet Backbone	29	45
IBM	17	23	Highwinds	16	29
ATT	25	56	BTNorthAmerica	36	76
Quest	19	30	CRLNetwork	32	37
Tinet	48	84	Darkstrand	28	31
Sprint	10	17	Integra	23	32
GEANT	32	50	Xspedius	33	47
Xeex	22	32	InternetMCI	18	32
CWIX	21	26	Deltacom	103	151
Digex	31	35	ION	114	135
IIJ	27	55			

two largest networks were Deltacom and Ion that contained 151 and 135 edges respectively and over a hundred nodes each. We remove one-degree nodes in the topologies recursively so that the networks are not disconnected with any single link failure. We use the gravity model [39] to generate traffic matrices with the utilization of the most congested link (MLU) in the range [0.6, 0.63] across the topologies.

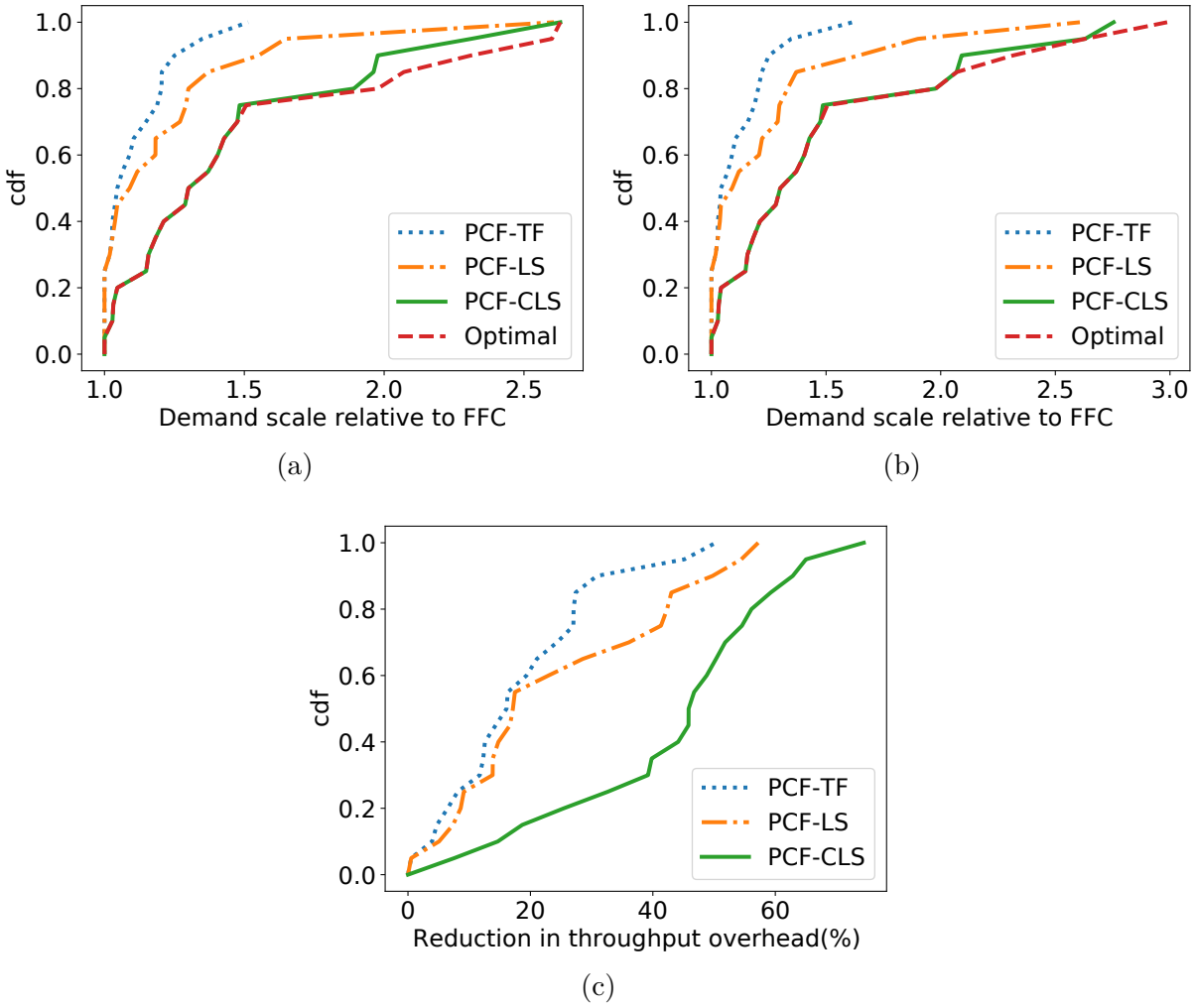


Figure 2.9. (a) PCF vs FFC across topologies. (b) Performance under three simultaneous failures. (c) Reduction in throughput overhead compared to FFC

2.4.1 Results

We start by reporting the demand scale (z) achieved by each scheme, which is the factor by which the traffic demand of all pairs can be scaled and yet supported by a given scheme. For example, $z = 0.5$ indicates that for all source destination pairs, half the demand can be served, while $z = 2$ indicates twice the demand can be handled. The MLU, or the utilization of the most congested link is the inverse of z. Later in this section, we report results with the throughput metric.

Benefits of modeling network structure. Fig. 2.7a shows the demand scale guaranteed by FFC when used to design for all single link failures for Deltacom (the topology with the most edges) for twelve different demands. Each curve corresponds to the number of tunnels used per node pair. We select physical tunnels so that they are as disjoint as possible, preferring shorter ones when there are multiple choices. With all our topologies, any node pair has at least two disjoint physical tunnels. When three or four tunnels are selected, it is not possible to guarantee that they are disjoint. Our strategy ensures that the failure of any link causes at most two tunnels to fail for all node pairs in the three tunnel case, and for most node pairs in the four tunnels case. The optimal is obtained by exhaustively enumerating all failure scenarios, and can take over 2 days in some settings. FFC performs significantly worse than optimal, and consistently better with two tunnels (additional tunnels hurt).

Fig. 2.8 shows the demand scale guaranteed by PCF-TF when designing for single link failures for Deltacom, and an example traffic matrix. Results for FFC are included for comparison. PCF-TF matches FFC's performance when 2 tunnels are used, and performs better as tunnels are added given that it better models network structure. We observed similar trends with all topologies, and across demands. Henceforth, in our experiments, all our schemes use three tunnels (this is conservative as adding more tunnels improves performance), while FFC uses two tunnels (this represents the best setting for FFC and choosing more tunnels leads to poorer performance).

Benefits of more flexible response. We next evaluate the performance of our various PCF schemes relative to FFC, and report the ratio of the demand scale for a given scheme to the demand scale with FFC. We generate 12 different demands for Deltacom to model a traffic matrix every 2 hours. Fig. 2.7b shows a CDF of the ratios across these demands. In the median case, PCF-TF and PCF-LS achieve an improvement of $1.25X$ over FFC, while PCF-CLS achieves a $1.37X$ improvement. Further, for 25% of the traffic matrices, PCF-TF, PCF-LS and PCF-CLS achieve improvements of more than $1.3X$, $1.4X$ and $1.66X$ over FFC respectively. Finally, PCF-CLS matches the optimal for most cases. While PCF-TF's improvements arise due to better modeling of network structure, the further benefits achieved by PCF-LS and PCF-CLS are due to additional flexibility provided by logical sequences.

Analysis across topologies. Fig. 2.9a presents a CDF of the ratios of the demand scale for each scheme relative to FFC across topologies when designing for single link failures. All our schemes provide significant benefits, with PCF-CLS matching the optimal for most topologies. On average, PCF-TF, PCF-LS and PCF-CLS achieve improvements of more than $1.11X$, $1.22X$ and $1.44X$ over FFC respectively. For GEANT (rightmost point), PCF-LS and PCF-CLS perform $2.6X$ better.

Multiple simultaneous failures. We next consider simultaneous link failures. To avoid disconnecting the topologies, we split the capacity of each link evenly across two sub-links that fail independently. We report the performance of all schemes when designing for all possible scenarios involving the simultaneous failure of three sub-links. For all PCF schemes we pick 6 tunnels, choosing them to be as disjoint as possible. For similar reasons as above, we found FFC achieved significantly better performance with 4 tunnels (FFC resulted in a demand scale factor of 0 with 6 tunnels).[2] Fig. 2.9b shows a CDF of the demand scale ratios for each scheme relative to FFC. On average, PCF-TF, PCF-LS and PCF-CLS achieve improvements of more than $1.11X$, $1.25X$ and $1.50X$ over FFC respectively. Note that while the trends are similar to single failures, the absolute values of demand scales are lower for all schemes – e.g., the optimal under 3 failures is 0.42 for Deltacom, while 0.85 under single failures).

Throughput metric. Instead of demand scale, we next consider performance when the schemes optimize the throughput metric (sum of bandwidth allocated to each pair). Given a demand d_{st} for source s and destination t, and an allocated bandwidth bw_{st} ($bw_{st} \leq d_{st}$), we compute the throughput overhead $1 - \frac{\sum bw_{st}}{\sum d_{st}}$. Fig.2.9c shows the % reduction in throughput overhead of each scheme relative to FFC when designing for three failures. PCF provides significant benefits. In the median case, PCF-TF and PCF-LS reduce the throughput overhead of FFC by more than 16%, and the reduction with PCF-CLS is 46%. For 25% of the topologies, PCF-TF, PCF-LS and PCF-CLS reduce the throughput overhead by 27%, 41% and 55% respectively. We do not report the optimal for this metric since it requires

[2]↑It was only feasible to select 6 tunnels, with 2 sharing a common link. Under three failures, FFC must provision for the case all tunnels failed.

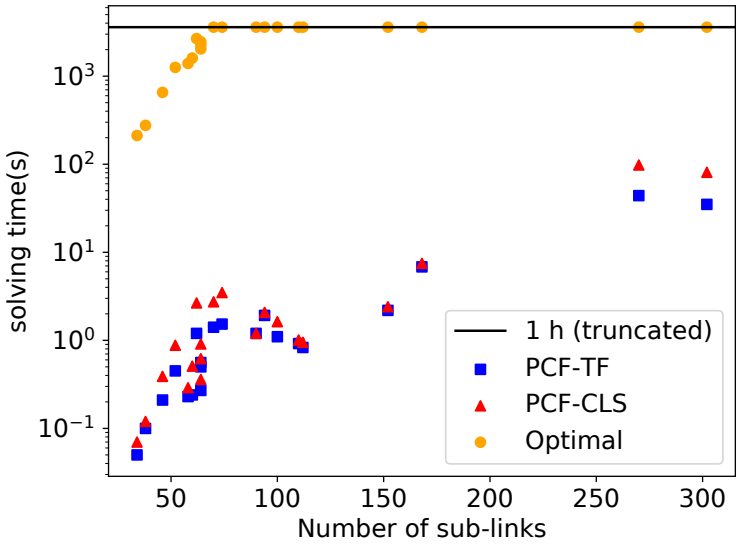


Figure 2.10. Solving time for PCF's schemes and optimal.

a prohibitively large optimization formulation that simultaneously models combinatorially many routing problems, one for each failure state.

2.4.2 Feasibility of local yet optimal routing

As discussed earlier, PCF-TF uses a routing mechanism identical to FFC, while PCF-LS uses topologically sorted logical sequences and can be realized using a locally proportional routing (§2.3.2) similar to FFC. However, the LSs chosen in PCF-CLS are not guaranteed to be topologically sorted.

Interestingly, under single link failures, the LSs generated by PCF-CLS are already topologically sorted by default for 16 of our 21 topologies. For the remaining ones, we consider a new scheme, which we refer to as PCF-CLS-TopSort, that starts with the LSs initially generated by PCF-CLS, and picks a subset which are topologically sorted. To achieve this, we use a greedy algorithm that adds LSs one by one from the original set, omitting any LS that violates the topological sort property. In all cases, less than 0.59% of the LSs were pruned. Further, for 4 of the 5 topologies, PCF-CLS-TopSort performs identically with PCF-CLS for the demand scale metric. For Ion alone there was some performance degradation, from a demand scale of 1.11 with PCF-CLS to 0.82 with PCF-CLS-TopSort, but still much better

than FFC which achieved a demand scale of 0.48. Overall, the results indicate that for single failure scenarios, a local proportional routing mechanism is sufficient to ensure near optimal performance.

For multiple simultaneous failures, PCF-CLS-TopSort does not match the performance of PCF-CLS. We note however that (i) PCF-CLS-TopSort and PCF-LS still significantly out-perform FFC and are realizable as local algorithms; and (ii) while PCF-CLS nearly matches optimal, it only requires a linear system of equations to be solved on each failure as opposed to a more expensive optimization problem.

2.4.3 Tractability of formulations

Fig. 2.10 presents the solving time (Y-Axis) against topology size (X-Axis) when PCF-TF, and PCF-CLS (the most complex scheme) are used to design for all simultaneous three link failure scenarios. Each point corresponds to a topology. PCF-LS takes less time than PCF-CLS and is not shown. For most topologies, the solving times is under 10 seconds. For the two largest networks (Deltacom and Ion) with 302 and 270 sub-links (each original link comprises 2 sub-links), the solving time for PCF-TF is under 50 seconds and for PCF-CLS under 100 seconds. This is reasonable because PCF's models only need to be run at the granularity of several minutes (on failure, lighter-weight online operations are used (§2.3.3)).

The figure also shows the solving time for the optimal scheme (truncating the Y-Axis at 1 hour). The solving time is much larger even for the smaller topologies and did not complete within an hour for most topologies. For one of the larger topologies, it did not complete even after two days.

2.5 Related work

Reactive vs. congestion-free routing schemes. Many recent traffic engineering (TE) schemes [40], [41] have developed flexible ways of routing traffic motivated by the goal of efficiently utilizing network capacity. Typically, these schemes involve deciding how to optimally route traffic at a centralized controller leveraging network-wide views [30], [40], [41]. Failures are handled *reactively* by recomputing routes at the centralized controller, and up-

dating rules at switches, a process that can take a long time, and that could congestion links in the interim [10]. A more recent work [30] derives tunnels from an oblivious routing strategy, and determines how to split traffic across tunnels so link utilizations are minimized. The scheme does not guarantee that the network would remain congestion-free on failure.

A second class of schemes [10], [20], [23] *proactively* guarantee the network remains congestion-free over a large set of failure scenarios (e.g., all scenarios with f simultaneous link failures), while only allowing for the network to respond to failures using fast and light-weight response mechanisms. For instance, FFC [10] conservatively admits traffic so the network does not experience congestion when local proportional routing is used. With such schemes, an optimization problem is only solved *offline* (i.e., prior to any failure scenario). The optimization models guarantee the congestion-free property, and are tractable in that they do not require an explicit enumeration of the large space of failure scenarios.

PCF addresses both objectives at the same time, by developing provably congestion-free light-weight mechanisms that achieve close to the optimal performance sought by reactive TE mechanisms. PCF not only out-performs existing congestion-free mechanisms, but performs close to optimal (the best possible performance that can be achieved by a reactive centralized TE scheme). Further, like other congestion-free schemes, PCF does not solve an LP on failure but only involves light-weight operations. Finally, with topologically sorted LSs, PCF uses local proportional routing, similar to FFC. Finally, while we do not explore in this paper, the tractable failure models associated with congestion-free schemes in general and PCF in particular can aid in network design tasks such as provisioning networks with sufficient capacity to protect against failures.

Other congestion-free routing schemes. Among congestion-free schemes, we have extensively discussed FFC [10]. R3, another congestion-free mechanisms based on link bypass [20], is based on flows, and cannot handle node failures (§2.2.5). PCF uses tunnels which are easier to deploy [30], and can tackle node failures. When flows are allowed, PCF provably out-performs R3 even for link failures (§2.2.5). Another work [23] addresses link failures by adding edges to the network. The original excess capacity of the network is not used, and the number of edges added may be substantial.

Rather than tackle all f failures, recent works Teavar [27], and Lancet [26] design for scenarios that occur with sufficient probability so a desired availability target is met. The techniques in Teavar [27] and Lancet [26] are respectively demonstrated with FFC and a generalization of R3. Techniques for probabilistic design are orthogonal to the design of congestion-free routing schemes. In particular, the ideas in both Teavar and Lancet are complementary to PCF, and may be potentially combined with PCF in the future to achieve better performance bounds when designing for scenarios with given probability.

A framework for analyzing the worst-case performance of centralized TE approaches was presented in [31]. The framework provides conservative performance bounds when network response can be modeled as an optimization problem. The conservative bounds may be viewed loosely equivalent to the performance of a more restricted network routing scheme that does not re-optimize on each failure. However, the bounds are obtained using optimization-theoretic relaxation methods, and it is an open question whether these abstract relaxations relate to practically realizable network response mechanisms. In contrast, all of PCF's models are associated with realizable network response mechanisms as we have discussed.

Interestingly, while we do not explore in this paper, PCF's models may provide alternate and better ways to bounds the performance of centralized TE schemes – e.g., the performance of PCF-CLS under failures matched the optimal for most topologies (§4.4). These benefits arise because using LSs can improve the bounds for the relaxations proposed in [31]. Finally, PCF's formulations can be naturally used to augment capacities so as to meet a desired performance metric by simply making capacities variable.

Segment and pathlet routing. Logical sequences are similar to segment routing [28], [29] in that traffic is steered through a given series of hops. DEFO considers ISP carrier network settings where the traffic in each segment is carried using a (possibly legacy) mechanism such as shortest-path forwarding, and the segments may be chosen so as to optimize a traffic engineering goal [29]. In contrast, LSs are an abstraction to increase the flexibility of provably congestion-free resilient routing mechanisms. Each LS is associated with a reservation, and may only be active when some conditions are met. Our actual implemen-

tation (§2.3.1) may be entirely tunneling based, or use both LSs and tunnels with a local proportional routing scheme (§2.3.2).

In pathlet routing [42], sources concatenate fragments of paths (pathlets) into end-to-end routes in a bottom-up fashion. In contrast, with PCF, logical sequences and physical tunnels are predefined in a top-down manner. Moreover, pathlet routing is motivated by the challenges of multipath routing, while it does not provide any performance guarantee upon failures.

Other related work. While several works explore quick re-routing of traffic to restore connectivity on failures [5]–[9], PCF guarantees the network is congestion-free (not merely restore connectivity). Oblivious routing provides bounds on network performance over multiple demands, and when networks do not adapt [15], [43]–[45]. PCF carefully adds flexibility to network response to allow for tractable analysis of performance under failures. Robust network design under single link or node failures has received attention [11]–[17]. PCF scales to the large number of failure states arising from concurrent failures, and shows how networks with carefully chosen response can achieve near optimal performance.

2.6 Conclusions

In this chapter, we have made two contributions. First, we have shown that existing mechanisms which ensure the network is congestion-free on failures achieve performance far short of the network's intrinsic capability, and shed light on the underlying reasons. Second, we have proposed PCF, a set of novel congestion-free mechanisms that bridges this gap by better modeling network structure, and by carefully enhancing the flexibility of network response to ensure that the performance under failures can be tractably modeled. Through formal theoretical results, we show PCF's schemes provably out-perform FFC. Empirical experiments over 21 Internet topologies show that PCF's schemes can sustain higher throughput than FFC by a factor of 1.11X to 1.5X on average across the topologies, providing a benefit as high as 2.6X in some cases. PCF's schemes are practically realizable, and some of them can yet achieve near optimal performance.

3. LANCET: PROTECTION ROUTING DESIGNED FOR PERCENTAGE OF SCENARIOS

In last chapter, we discuss how PCF designs a congestion-free routing and improves the worst-case performance under a given set of failures. But how to ensure traffic demand is handled a percentage of time still remains a question. In this chapter, we will target at this question and use protection routing to design for percentage of scenarios.

Two mechanisms are used for quick recovery from network failures today: (i) link-based protection; and (ii) path-based protection [11]. In link-based protection [20], traffic on a link $l = \langle \mathrm{i}, \mathrm{j} \rangle$ is re-routed upon its failure, along pre-computed detour paths from i to j. To achieve this, an offline procedure is used, which for each link $l = \langle \mathrm{i}, \mathrm{j} \rangle$, makes a bypass reservation not used until l fails, along paths that are disjoint from l and can carry flow from i to j. When l fails, the network uses this reservation to re-route the traffic on l and executes an efficient online procedure to compute the changes required should another failure occur. In contrast, in a path-based protection scheme [10], failures are handled by diverting traffic to pre-computed backup paths between each source and destination.

In this chapter, we focus on link-based protection. The primary advantage of link-based protection is that repair is faster since it happens locally (i.e., at node i if link $\langle \mathrm{i}, \mathrm{j} \rangle$ fails) - in contrast, with path-based schemes, the failure information must be propagated to the source. In the rest of this chapter, we discuss a model for link-based protection, which generalizes the state-of-the-art scheme [20]. Then we extend this model to design routing with multiple traffic classes. We also present how this mechanism can be implemented in terms of online operation. And finally, we show how to apply it in a framework to design for a certain percentage of scenarios instead of all failure scenarios.

3.1 Generalized protection routing model

3.1.1 Protection routing definition

Consider a network with nodes V and edges E for a traffic matrix d with each link e having a capacity c_{e}. Determining a protection routing, requires computing three sets of

Table 3.1. Symbol table.

	Symbol	Definition
Parameters	c_{e}	Capacity of link e
	d_{st}	Traffic from s to t
Variables	a_l	Bypass reservation for link l
	x_{e}	Failure status of link e
	$r_{st}(\mathrm{e})$	Fraction of s to t traffic on link e (no failure)
	$p_l(\mathrm{e})$	Reservation on link e to handle link l failure

variables, r, p, and a, where r denotes routing under normal condition, p denotes protection routing, and a denotes bypass reservations. (Table 3.1). r is represented by a set of values $r_{st}(\mathrm{e}), s, t \in V, \mathrm{e} \in E$, which denotes the fraction of traffic from source s to destination t that traverses link $\mathrm{e} = \langle \mathrm{i}', \mathrm{j}' \rangle$ under normal conditions. We use the notation $r_{st}(\mathrm{e})$ and $r_{st}(\mathrm{i}', \mathrm{j}')$ interchangeably. For each s, t pair, r represents a flow, and must satisfy the constraints presented below which capture that one unit of traffic exits s and enters t, and traffic is conserved at intermediate nodes. In addition, r should also satisfy capacity constraints, and the full model will be presented in LP (H) below.

$$\begin{aligned} &\sum_{\mathrm{j}';\langle \mathrm{i}',\mathrm{j}' \rangle \in E} r_{st}(\mathrm{i}', \mathrm{j}') - \sum_{\mathrm{j}';\langle \mathrm{j}',\mathrm{i}' \rangle \in E} r_{st}(\mathrm{j}', \mathrm{i}') \\ &\quad = \begin{cases} 1 & \mathrm{i}' = s \\ 0 & \mathrm{i}' \neq s, \mathrm{i}' \neq t \quad \forall \mathrm{i}' \in V \\ -1 & \mathrm{i}' = t \end{cases} \\ &r_{st}(\mathrm{i}', \mathrm{j}') \geq 0 \quad \forall \mathrm{i}', \mathrm{j}'; \langle \mathrm{i}', \mathrm{j}' \rangle \in E \end{aligned} \tag{3.1}$$

The reservation a is represented by a set of values $a_l, l \in E$, which denotes the amount of reservation along bypass paths to protect against the failure of link l. The reservation a_l to protect against the failure of a link l may be less, equal, or larger than c_l but cannot be arbitrarily large since sufficient capacity must be available on the bypass paths. The variable $p_l(\mathrm{e})\ \forall l, \mathrm{e} \in E$ denotes the reservation on link e to handle the failure of $l = \langle \mathrm{i}, \mathrm{j} \rangle$. p_l represents a flow of a_l from i to j, and must satisfy constraints similar to (3.1) except replacing

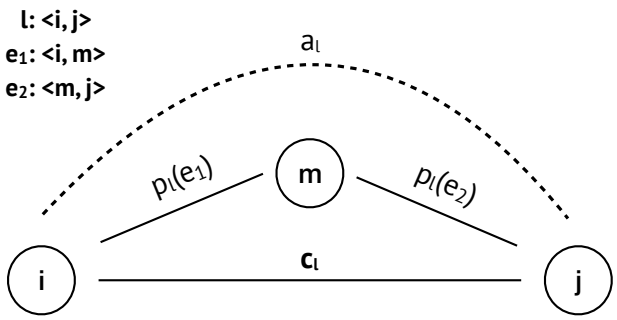


Figure 3.1. Illustrating protection routing.

(i) $r_{st}(\mathrm{i}', \mathrm{j}')$ with $p_l(\mathrm{i}', \mathrm{j}')$; (ii) the right-hand-side (RHS) of the flow balance equation with a_l, 0, and $-a_l$; and (iii) (s, t) with $l = \langle \mathrm{i}, \mathrm{j} \rangle$. Further, the routes protecting against e$'s$ failure should not traverse e, i.e., $p_{\mathrm{e}}(\mathrm{e}) = 0$.

Fig. 3.1 illustrates the notation related to bypass reservations. c_l denotes the link capacity of link l. In order to protect against the failure of link l, we reserve the amount a_l to bypass the traffic between i and j. a_l can only be used when l fails. In this example, we use e_1 and e_2 to construct the bypass. To support a flow of a_l, e_1 and e_2 need to carry $p_l(\mathrm{e}_1)$ and $p_l(\mathrm{e}_2)$ respectively. In this example, each of these variables are equal to a_l because of the flow balance constraints on the routing p_l.

3.1.2 Offline protection routing design

We next present an offline linear program (LP) that computes an optimal protection routing (r, p, a) so as to protect the network against all failure scenarios in a set X. For instance, if the architect wishes to protect against the set of all simultaneous f link failure scenarios, X consists of the set of $\binom{|E|}{f}$ possible f link failure scenarios. The LP minimizes the utilization of the most congested link (or Maximum Link Utilization, henceforth referred to as MLU) across all scenarios $x \in X$. We focus on this metric since it is widely used in the traffic engineering community [20], [31]. If the optimal MLU U is under 1, then it indicates the protection routing is *congestion free*, i.e., the network can handle any failure scenario

$x \in X$. An MLU higher than 1 indicates that the network cannot guarantee a congestion-free routing for all scenarios $x \in X$.

$$
\begin{aligned}
&\text{(H)} \quad \min_{r,p,a,U} \quad U \\
&\text{s.t. } r_{st} \text{ is a unit flow from } s \text{ to } t. \quad \forall s,t \in V \\
&\qquad p_l \text{ is a flow of } a_l \text{ from i to j.} \quad \forall l \in E, l = \langle \mathrm{i}, \mathrm{j} \rangle \\
&\qquad \forall x \in X, \mathrm{e} \in E, \\
&\qquad \sum_{s,t} d_{st} r_{st}(\mathrm{e}) + \sum_{l \in E} x_l p_l(\mathrm{e}) \leq U c_{\mathrm{e}}(1 - x_{\mathrm{e}}) + a_{\mathrm{e}} x_{\mathrm{e}} \qquad (3.2) \\
&\qquad a_{\mathrm{e}} \geq 0 \quad \forall \mathrm{e} \in E; \quad U \geq 0
\end{aligned}
$$

The LP is show above. The first two constraints capture the definition of a protection routing as elaborated above and summarized in (3.1). (3.2) captures the capacity and reservation constraints of all links must be met under all possible failures $x \in X$. The two terms on the left-hand-side (LHS) indicate the total traffic that link e must carry, which includes (i) the traffic under normal conditions ($\sum_{s,t} d_{st} r_{st}(\mathrm{e})$); and (ii) the bypass reservations made on link e to protect against other link failures ($\sum_{l \in E} x_l p_l(\mathrm{e})$). The RHS captures that (i) link e carries at most $U c_{\mathrm{e}}$ traffic when e is operational; and (ii) when e fails ($x_{\mathrm{e}} = 1$), a reservation of at most a_{e} is available along bypass paths.

A key difficulty in translating (H) into an LP is that the obvious strategy would create $|E|$ constraints per failure scenario, which is not scalable since X may need to be designed for potentially up to $\binom{|E|}{f}$ f-failure scenarios. We reformulate (H) into a more tractable LP by (i) relaxing the integrality requirements on x variables and (ii) expressing the LHS as a minimization problem leveraging LP duality. A similar approach was used by [20] when designing for all f failures.

Our treatment is a generalization of R3 [20], the state-of-the-art protection routing scheme. To protect against the failure of e, R3 reserves a fixed amount c_{e} that matches the capacity of link e. Instead, our formulation introduces the a_{e} variables, and thus determines the bypass reservation to be made. There are two advantages to our approach. First, our formulation is valid even when the utilization is higher than 1 while R3 is not.

This allows the formulation to be used in settings where we wish to determine how best to augment link capacity to handle failures [31]. Second, when utilization is under 1, our formulation achieves lower MLU than R3. For instance, for the Abilene network [24], and an example traffic matrix [46], the MLU with R3 when protecting against all 2-failure scenarios (each failure impacting 50% of link capacity) is 0.56. In contrast, the MLU with (H) is 0.12 (the optimal achievable if the network could respond ideally). We refer to this generalized approach as Gen-R3.

3.1.3 Design with multiple traffic classes

So far, we have focused on the MLU metric when all traffic must be handled. We next show how to extend to a context with multiple traffic classes where the architect wishes to meet all high priority, and as much low priority traffic as possible.

Designing a protection routing for multiple traffic classes involves minor changes to LP (H) and is presented below. Let d^h and d^l represent high and low priority traffic matrices, with d^h_{st} and d^l_{st} representing the relevant traffic from s to t. The formulation determines the largest Z such that the network can handle a traffic matrix D where D_{st} is $d^h_{st} + Zd^l_{st}$ (i.e., the network can carry all high priority traffic, and a fraction Z of low priority traffic). $Z \geq 1$ indicates all low priority traffic can be carried. Setting d^h to zero produces the special case where there is a single class of traffic, when Z would share an inverse relationship with the MLU metric. The protection routing has parameters (r^h, r^l, p, a), where r^h and r^l represent flows corresponding to high and low priority traffic from s to t. The formulation is similar

to (H) except that r^l_{st} only need carry a fraction Z of the d^l_{st} traffic, however link capacity constraints must be strictly met.

$$
\begin{aligned}
\text{(G)} \quad & \max_{r^h, r^l, p, a, Z} \quad Z \\
\text{s.t.} \quad & r^h_{st} \text{ is a flow of } d^h_{st} \text{ from } s \text{ to } t. \quad \forall s, t \in V \\
& r^l_{st} \text{ is a flow of } Z d^l_{st} \text{ from } s \text{ to } t. \quad \forall s, t \in V \\
& p_l \text{ is a flow of } a_l \text{ from i to j.} \quad \forall l \in E, l = \langle \mathrm{i}, \mathrm{j} \rangle \\
& \sum_{s,t} r^h_{st}(\mathrm{e}) + \sum_{s,t} r^l_{st}(\mathrm{e}) + \sum_{l \in E} x_l p_l(\mathrm{e}) \\
& \qquad \leq c_{\mathrm{e}}(1 - x_{\mathrm{e}}) + a_{\mathrm{e}} x_{\mathrm{e}} \quad \forall x \in X, \mathrm{e} \in E \\
& a_{\mathrm{e}} \geq 0 \quad \forall \mathrm{e} \in E \\
& Z \geq 0
\end{aligned}
$$

3.2 Efficient online adjustment of protection routing parameters

Consider a protection routing (r, p, a) initially computed offline using LP (H) (Table 3.1). If link $\epsilon = \langle \mathrm{i}, \mathrm{j} \rangle$ fails, the parameters for the routing must be recomputed since ϵ is unavailable for reservations to protect against future failures. To achieve this, failure information of $\langle \mathrm{i}, \mathrm{j} \rangle$ is propagated to the controller which efficiently adjusts the parameters as we discuss next. Since this update is only to protect against future failures, it is acceptable to involve the controller.

One approach to achieving this is for the controller to solve the design LP again to compute (r', p', a') but for the network where link ϵ does not exist. However, this can take time and is not suitable for online operation. Instead, we have been able to show that (r', p', a') can be efficiently computed online by making the following quickly computable adjustments to (r, p, a). Specifically, we have:

$$r'_{st}(\mathrm{e}) = \begin{cases} r_{st}(\mathrm{e}) + r_{st}(\epsilon)(p_\epsilon(\mathrm{e})/a_\epsilon) & \forall s,t,\mathrm{e} \in E\backslash\{\epsilon\} \\ 0 & \forall s,t,\mathrm{e} = \epsilon \end{cases}$$

$$\tilde{p}_l(\mathrm{e}) = \begin{cases} p_l(\mathrm{e}) + p_l(\epsilon)(p_\epsilon(\mathrm{e})/a_\epsilon) & \forall l \in E\backslash\{\epsilon\}, \mathrm{e} \in E\backslash\{\epsilon\} \\ 0 & \forall l \in E\backslash\{\epsilon\}, \mathrm{e} = \epsilon \end{cases}$$

$$a'_{\mathrm{e}} = a_{\mathrm{e}} - \tilde{p}_{\mathrm{e}}(\mathrm{e}) \quad \forall \mathrm{e} \in E\backslash\{\epsilon\}$$

$$p'_l(\mathrm{e}) = \begin{cases} \tilde{p}_l(\mathrm{e}) & \forall l \in E\backslash\{\epsilon\}, \mathrm{e} \in E, l \neq \mathrm{e} \\ 0 & \forall l \in E\backslash\{\epsilon\}, \mathrm{e} \in E, l = \mathrm{e} \end{cases}$$

These adjustments capture the increase in (s,t) traffic on link e when ϵ fails, for each (s,t) pair, and the increase in reservation on link e to protect against a subsequent failure of l, because reservations on ϵ have been invalidated. The final few steps ensure link e does not use itself when protecting against the failure of e. The update to r' is locally computed by each router since router i starts labeling packets to be sent along ϵ and forwards them using the protection routing p as discussed earlier. Thus, the controller only pushes p' and a' to switches. We next discuss the complexity of the update process. During the online adjustment, a node i will locally update r for each of its adjacent links, and for all source-destination pairs. As shown above, this takes $d_{\mathrm{i}}|V|^2$ operations where d_{i} is the degree of node i and V is the set of nodes. However, while we do not elaborate, it is possible to optimize this further by implementing r in a manner that only tracks the destination rather than both the source and destination, which would result in only $d_{\mathrm{i}}|V|$ operations. Further, p is updated in a centralized manner and the process takes $O(|E|^2)$ operations.

Proposition 3.2.1. *Let (r,p,a,U) be a feasible solution to LP (H) for a graph $G=(V,E)$ when protecting against all failure scenarios defined in the set X. Then, we can show that (r',p',a',U) is a feasible solution to (H) for the graph $G'=(V,E-\epsilon)$ and for failure scenarios in X with ϵ failing.*

Proof. Since (r, p, a, U) (parameters prior to failure) is feasible for LP (H), it satisfies every constraint of (H) for $G = (V, E)$ when protecting against all failure scenarios defined in X. The proof proceeds by showing that (r', p', a', U) obtained after the failure of ϵ and adjustments satisfies every constraint of LP (H) for the graph $G' = (V, E - \epsilon)$ and for failure scenarios in X with ϵ failing.

First, we show that r'_{st} is a unit flow from s to t for all s, t pairs. For convenience, let's define a function $g(\mathrm{i}, s, t)$ as:

$$g(\mathrm{i}, s, t) = \begin{cases} 1 & \mathrm{i} = s \\ 0 & \mathrm{i} \neq s, \mathrm{i} \neq t \\ -1 & \mathrm{i} = t \end{cases} \tag{3.3}$$

Then, the flow balance constraint on r_{st} can be written as:

$$\sum_{\mathrm{j}';\langle \mathrm{i}',\mathrm{j}'\rangle \in E} r_{st}(\mathrm{i}', \mathrm{j}') - \sum_{\mathrm{j}';\langle \mathrm{j}',\mathrm{i}'\rangle \in E} r_{st}(\mathrm{j}', \mathrm{i}') = g(\mathrm{i}', s, t) \tag{3.4}$$

We now verify that the flow balance constraint for r'_{st} is met:

$$\begin{aligned} &\forall \mathrm{i}', s, t, \\ &\sum_{\mathrm{j}';\langle \mathrm{i}',\mathrm{j}'\rangle \in E\setminus\{\epsilon\}} r'_{st}(\mathrm{i}', \mathrm{j}') - \sum_{\mathrm{j}';\langle \mathrm{j}',\mathrm{i}'\rangle \in E\setminus\{\epsilon\}} r'_{st}(\mathrm{j}', \mathrm{i}') \\ &= \Big(\sum_{\mathrm{j}';\langle \mathrm{i}',\mathrm{j}'\rangle \in E\setminus\{\epsilon\}} r_{st}(\mathrm{i}', \mathrm{j}') - \sum_{\mathrm{j}';\langle \mathrm{j}',\mathrm{i}'\rangle \in E\setminus\{\epsilon\}} r_{st}(\mathrm{j}', \mathrm{i}')\Big) + \\ &\frac{r_{st}(\epsilon)}{a_\epsilon}\Big(\sum_{\mathrm{j}';\langle \mathrm{i}',\mathrm{j}'\rangle \in E\setminus\{\epsilon\}} p_\epsilon(\mathrm{i}', \mathrm{j}') - \sum_{\mathrm{j}';\langle \mathrm{j}',\mathrm{i}'\rangle \in E\setminus\{\epsilon\}} p_\epsilon(\mathrm{j}', \mathrm{i}')\Big) \\ &= \begin{cases} g(\mathrm{i}', s, t) - r_{st}(\epsilon) + \frac{r_{st}(\epsilon)}{a_\epsilon} * a_\epsilon & \mathrm{i}' = u \\ g(\mathrm{i}', s, t) + \frac{r_{st}(\epsilon)}{a_\epsilon} * 0 & \mathrm{i}' \neq u, \mathrm{i}' \neq v \\ g(\mathrm{i}', s, t) + r_{st}(\epsilon) - \frac{r_{st}(\epsilon)}{a_\epsilon} * a_\epsilon & \mathrm{i}' = v \end{cases} \\ &= g(\mathrm{i}', s, t) \end{aligned} \tag{3.5}$$

Similarly, we can show that p'_l is a flow of a'_l from i to j for all $l = \langle \mathrm{i}, \mathrm{j}\rangle \in E$.

Second, we show that updated routing and future protection will not traverse ϵ. By definition, $\forall s,t : r'_{st}(\epsilon) = 0$ and $\forall \mathrm{e} \in E \backslash \{\epsilon\} : p'_{\mathrm{e}}(\epsilon) = 0$. So link ϵ will not be used in the updated routing and protection.

Third, we show that (r', p', a', U) still satisfies the capacity constraint for all failure scenarios x with $x_\epsilon = 1$ in the certified failure set X. From the capacity constraint on ϵ for any failure scenario $x \in X$ with $x_\epsilon = 1$, we have:

$$\sum_{s,t} d_{st} r_{st}(\epsilon) + \sum_{l \in E \backslash \{\epsilon\}} x_l p_l(\epsilon) \leq a_\epsilon \tag{3.6}$$

From the capacity constraint for $\mathrm{e} \neq \epsilon$ for any failure scenario $x \in X$ with $x_\epsilon = 1$, we have:

$$\sum_{s,t} d_{st} r_{st}(\mathrm{e}) + p_\epsilon(\mathrm{e}) + \sum_{l \in E \backslash \{\epsilon\}} x_l p_l(\mathrm{e}) \leq c_{\mathrm{e}}(1 - x_{\mathrm{e}}) + a_{\mathrm{e}} x_{\mathrm{e}} \tag{3.7}$$

Then the steps below show that the capacity constraints on other links after updating continue to be met, thereby proving the proposition.

$$
\begin{aligned}
&\forall \mathrm{e} \in E \backslash \{\epsilon\}, \\
&\sum_{s,t} d_{st} r'_{st}(\mathrm{e}) + \sum_{l \in E \backslash \{\epsilon\}} x_l p'_l(\mathrm{e}) \\
&= \sum_{s,t} (d_{st} r_{st}(\mathrm{e}) + d_{st} r_{st}(\epsilon) \frac{p_\epsilon(\mathrm{e})}{a_\epsilon}) + \\
&\sum_{l \in E \backslash \{\epsilon\}} (x_l p_l(\mathrm{e}) + x_l p_l(\epsilon) \frac{p_\epsilon(\mathrm{e})}{a_\epsilon}) - x_\mathrm{e} \tilde{p}_\mathrm{e}(\mathrm{e}) \\
&= \sum_{s,t} d_{st} r_{st}(\mathrm{e}) + \frac{p_\epsilon(\mathrm{e})}{a_\epsilon} (\sum_{s,t} d_{st} r_{st}(\epsilon) + \sum_{l \in E \backslash \{\epsilon\}} x_l p_l(\epsilon)) + \\
&\sum_{l \in E \backslash \{\epsilon\}} x_l p_l(\mathrm{e}) - x_\mathrm{e} \tilde{p}_\mathrm{e}(\mathrm{e}) \\
&\leq \sum_{s,t} d_{st} r_{st}(\mathrm{e}) + \frac{p_\epsilon(\mathrm{e})}{a_\epsilon} a_\epsilon + \sum_{l \in E \backslash \{\epsilon\}} x_l p_l(\mathrm{e}) - x_\mathrm{e} \tilde{p}_\mathrm{e}(\mathrm{e}) \\
&= (\sum_{s,t} d_{st} r_{st}(\mathrm{e}) + p_\epsilon(\mathrm{e}) + \sum_{l \in E \backslash \{\epsilon\}} x_l p_l(\mathrm{e})) - x_\mathrm{e} \tilde{p}_\mathrm{e}(\mathrm{e}) \\
&\leq c_\mathrm{e}(1 - x_\mathrm{e}) + a_\mathrm{e} x_\mathrm{e} - x_\mathrm{e} \tilde{p}_\mathrm{e}(\mathrm{e}) \\
&= c_\mathrm{e}(1 - x_\mathrm{e}) + a'_\mathrm{e} x_\mathrm{e}
\end{aligned}
\tag{3.8}
$$

3.3 Design percentage of scenarios

Up till now, our discussion focuses on how to design congestion-free routing under all possible f or fewer simultaneous link failures, where f specifies the desired resilience level. Unfortunately, even a few bad cases may make it infeasible to achieve this goal, forcing a design with much lower resilience. For example, if an architect wishes to protect against 3-failure scenarios, a small number of bad 2- and 3-failure cases imply that the architect can only design for single failures with the existing approach. Thus, it is important to analyze which failure scenarios a network is intrinsically capable of tackling, and to enable the design of a protection routing that can protect against most scenarios with f or fewer failures (all single and most 2- and 3-failure scenarios in the above example) when infeasible to protect

against all such scenarios. We also need to compare which scenarios a heuristically designed protection routing can tackle relative to intrinsic network capability.

3.3.1 Lancet system

Lancet [26] is developed to achieve these goals. Lancet employs a divide-and-conquer algorithm, which classifies an entire set of scenarios as acceptable or violating to the extent possible, and when necessary, partitions the set further. A key highlight of the algorithm is that it generates a compact representation of the large number of scenarios with acceptable performance as the union of m sets, where m is small.

Lancet can be used to design to meet probability requirements. Consider an architect goal of design a protection routing that is guaranteed to be congestion free for scenarios that occur occur $p\%$ of the time, given the probability of different failure scenarios. Lancet is easily adapted to this task. Specifically, the classification algorithm is modified to maintain a weighted count of scenarios for which performance is acceptable or violates requirements, with the weight indicating the probability that the network is in a given failure state. The algorithm terminates when either the weighted count of certifiable scenarios exceeds $p\%$, or that of violating scenarios exceeds $1-p\%$ (indicating the network is not intrinsically capable of meeting the goal).

3.3.2 Lancet with multiple traffic classes model

Now, we apply our multiple traffic classes model to Lancet to design routing for percentage of scenarios. The two-class LP in 3.1.3 determines a protection routing that handles all high-priority traffic and the low-priority traffic scaled by a factor Z. We refer to Gen-R3(f) as a protection routing derived from this two-class LP when protecting against all simultaneous failures involving f or fewer links. Similar to earlier, Lancet is obtained by (i) using Lancet to classify scenarios that obtain a $Z \geq 1$ with Centralized; and (ii) using the two-class LP to design with the set so obtained.

Fig. 3.2 shows the fraction of scenarios that achieve different Z thresholds for the schemes above for GEANT's 2-failure scenarios. We split the original traffic matrix into two classes

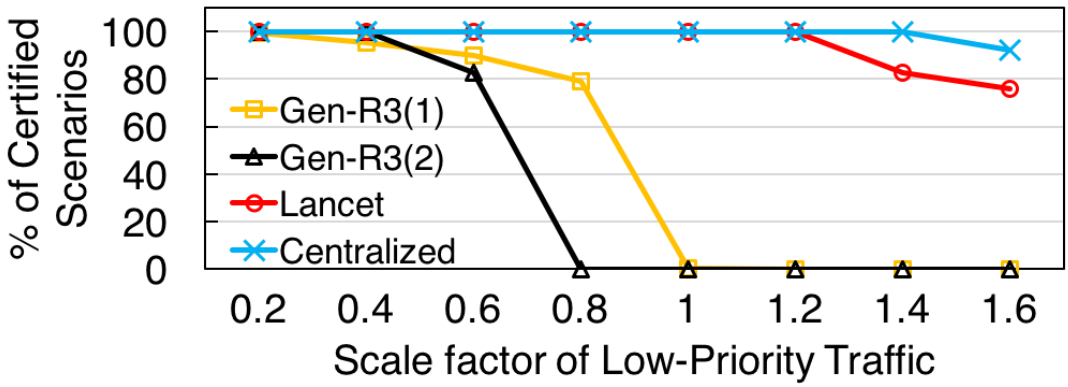


Figure 3.2. Efficacy of Lancet aided designs with two traffic classes.

with high and low priority. For each cell we assign a random fraction of it to the high-priority traffic class, and the rest to the low-priority one. Each curve corresponds to one scheme, and shows the fraction of 2-failure scenarios that can attain a particular Z. The top-most curve shows the ideal Centralized scheme, which attains Z of 1 for over 99% of the scenarios. Lancet performs nearly as well as Centralized. While it degrades moderately for the most stringent performance thresholds ($Z = 1.4$ and 1.6), we note that an architect could use Lancet to generate new protection routings optimized for these thresholds if this is desirable. The two Gen-R3 schemes perform poorly, with no scenario achieving a Z of 1 where all low priority traffic could be carried. Note that Gen-R3(2) matches Centralized and performs slightly better than Lancet for the worst-case (achieving a Z of 0.42) but this comes at the expense of performance for the vast majority of scenarios.

3.4 Conclusions

In this chapter, we present a generalized model of protection routing which designs routing for at most f simultaneous link failures. We extend this model to a context with multiple traffic classes where the architect wishes to send all high priority traffic and as much low priority traffic as possible. We present how the protection routing can be practically implemented through efficient online adjustment and prove the correctness of this adjustment scheme. Finally, we combine Lancet, a framework to design routing for a percentage of

scenarios, with our multiple traffic classes model, and show the effectiveness of our model through evaluation.

4. FLEXILE: MINIMAL FLOW LOSS ALMOST ALWAYS

As mentioned before, most state-of-the-art traffic engineering (TE) schemes do not explicitly provide ways to optimize performance at a desired percentile. Lancet took the first step toward solving this problem. In the meantime, Teavar [27] has also been developed to consider percentiles when designing routing. However, we find that Teavar provides extremely conservative guarantees. The poor performance stems partially from the fact that Teavar uses a common set of failure states to evaluate the percentile loss of all flows. Lancet will suffer the same issue. Moreover, Teavar is approximate since it minimizes an overestimate of percentile loss, and uses a less flexible routing strategy. Teavar's performance is improved by flexibly and optimally routing traffic in each scenario (an approach advocated by SMORE [30]). Nevertheless, the bandwidth is allocated such that some of the traffic continues to see significant loss at desired percentile. The key reason is that such a scenario-centric approach optimizes traffic unilaterally for each failure state, which leads to sub-optimal decisions across states. For example, the same flow may be penalized in many bandwidth constrained network states.

Contributions. To tackle these issues, we present ***Flexile*** (FLEXibily choose scenarios for each flow to evaluate loss percentILE). *Flexile* (i) ensures all flows see as low a loss as possible at a desired percentile; (ii) supports multiple traffic classes (e.g., minimize 99.9%ile loss for latency-sensitive traffic, and 99%ile for other traffic); and (iii) directly optimizes loss percentiles. *Flexile* does so by allowing flows to meet their bandwidth requirements in a possibly different subset of *critical states* that occur with sufficient probability. Although this couples bandwidth allocation decisions across failure states, *Flexile* decouples them by identifying critical states for each flow in an offline phase. Then, on failure, *Flexile* efficiently allocates bandwidth online, while paying more attention to critical flows.

We evaluate *Flexile* on 20 topologies, including many large topologies, and validate our results on an emulation testbed. We show that *Flexile* consistently outperforms existing TE strategies such as SWAN, SMORE and Teavar. Across topologies, the median reduction in flow loss at desired percentiles with *Flexile* is 46% for SMORE, and 63% for Teavar, and the benefits are even higher for SWAN. *Flexile* outperforms Teavar for various reasons that include evaluation of losses at flow level, more flexible rerouting, and optimization

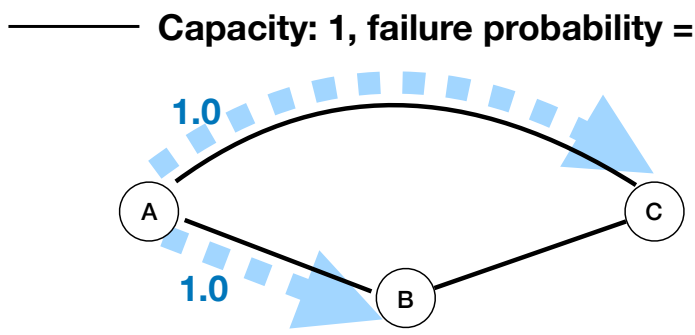
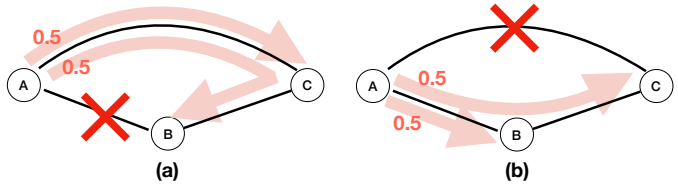


Figure 4.1. Illustrating Flexile's opportunity. Flow 1 can be fully sent over link A-B 99% of time. Flow 2 can be fully sent over link A-C 99% of time.



Scenario	AB and AC both alive (98.01%)	Only AB fails (0.98%)	Only AC fails (0.98%)	Others (0.03%)
Throughput of flow 1 (A->B)	1	0.5	0.5	-
Throughput of flow 2 (A->C)	1	0.5	0.5	-

(c)

Figure 4.2. Bandwidth objectives cannot be met for topology in Fig.4.1 by existing TE schemes.

of percentile instead of an overestimate. For comparison, we design and compare *Flexile* with generalizations of Teavar that route flexibly and evaluate losses at flow level. While these generalizations help, *Flexile* continues to out-perform. By exploiting various problem characteristics, we show that *Flexile*'s offline decomposition algorithm runs quite efficiently in practice, and is an order of magnitude faster than Teavar for the largest topology. Finally, *Flexile* maintains the online reaction times of existing TE schemes [30], [47].

4.1 Motivation

4.1.1 Background

Given traffic demand associated with a set of flows, TE schemes must decide the bandwidth to allocate to each flow, and how each flow must be routed so a desired performance metric is optimized. Many TE schemes minimize the maximum link utilization (MLU) of all links [30]). Equivalently, TE schemes may maximize the demand scale factor using a maximum concurrent flow formulation [10] or minimize the maximum loss across all flows,

Scenario	AB fails, AC alive (0.99%)	Both AB, AC alive (98.01%)	AB alive, AC fails(0.99%)	Others (0.01%)
Critical for flow 1 (A->B)		x	x	
Critical for flow 2 (A->C)	x	x		

Figure 4.3. Critical scenarios for Fig. 4.1.

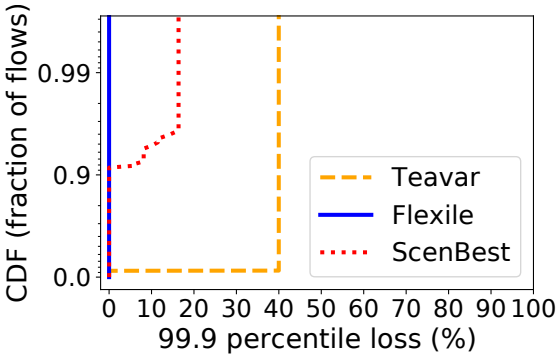


Figure 4.4. CDF of 99.9%ile loss across flows for IBM topology.

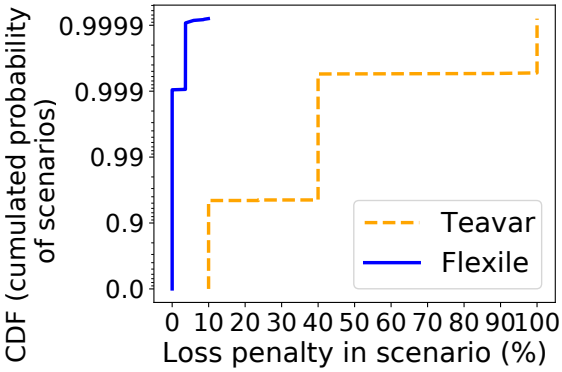


Figure 4.5. Increase in *ScenLoss* relative to ScenBest (optimal).

referred henceforth as *ScenLoss*. Many TE schemes [20], [30] optimize the utilization of the most congested link (Maximum Link Utilization or MLU), or alternately,solve the maximum concurrent flow, and maximize the fraction z (that we also refer to as scale factor) of demand the network can handle. Minimizing MLU, or maximizing z is equivalent to minimizing *ScenLoss* (the maximum loss across all pairs in a given scenario), since $ScenLoss = \max\{0, 1-z\}$, and $ScenLoss = \max\{0, 1-1/\text{MLU}\}$. We next consider a representative TE scheme **Teavar**.

Teavar [27] falls under a class of schemes [10], [20], [23], [27], [48] which explicitly guarantee the network remains congestion-free over a desired set of failure scenarios by conservatively allocating bandwidth. While most of these schemes [10], [20], [23], [48] ensure the congestion-free property over all scenarios with f simultaneous link failures, we focus on Teavar [27] since it is the only scheme that considers failure probabilities. Although the goal is to minimize the β^{th} percentile of *ScenLoss*, Teavar approximates – e.g., rather than minimize the 99^{th} percentile of *ScenLoss*, the actual formulation minimizes the average *ScenLoss* in the worst 1% of scenarios (see §4.3).

Teavar (like FFC [10]) uses a less flexible routing approach than the best possible scheme for each scenario. Specifically, it pre-determines how traffic is split across tunnels, and on

failure, proportionally rescales traffic on live tunnels. A more flexible yet still light-weight recovery mechanism (proposed by SMORE [30]) involves splitting traffic optimally among live tunnels (with the tunnels themselves not changing over scenarios). We term such a scheme **ScenBest** for generality, but note that when the metric optimized is MLU, ScenBest performs identically to SMORE[1]. Clearly, for any failure scenario, Teavar can perform no better than ScenBest. Hence, Teavar cannot achieve a percentile performance any better than ScenBest.

4.1.2 Example Motivating Flexile

Next, we illustrate the potential opportunity that Flexile exploits using concrete examples. Consider Fig.4.1, where a network must carry traffic corresponding to a flow f_1 from source A to destination B, and a flow f_2 from source A to destination C. Consider a requirement that each of f_1 and f_2 must support 1 unit of traffic 99% of the time. Each link has a capacity of 1 and a failure probability 0.01. We make the following observations:

The network can easily meet the bandwidth requirements. This clearly follows from the following simple routing strategy. The strategy sends f_1 on link $A{-}B$ and f_2 on link $A{-}C$. Clearly, the requirements of f_1 are met whenever link $A{-}B$ is alive, which occurs 99% of the time. Likewise, the requirements of f_2 are met in all scenarios where $A{-}C$ is alive, which occurs 99% of the time.

State-of-the-art TE schemes cannot meet the requirements of flows. Unfortunately, ScenBest and Teavar [27] can each only support 0.5 units for f_1 and f_2 99% of the time. To illustrate this, consider Fig 4.2(a) and Fig 4.2(b) which illustrate ScenBest's routing in each of two scenarios (one where link $A{-}B$ fails, and another where link $B{-}C$ fails). In both cases, ScenBest would send 0.5 units of each flow as shown. Fig.4.2(c) summarizes the throughput achieved by each flow under different failure scenarios. Since each of the scenarios shown in Fig 4.2(a) and Fig 4.2(b) occurs 0.98% of the time, to meet its bandwidth requirement, a flow must be able to send necessary traffic in at least one of the scenarios. Consequently, neither f_1 nor f_2 can support more than 0.5 units 99% of time.

[1]↑While SMORE [30] does not extensively discuss SMORE's failure recovery mechanisms, we clarified this from the authors and the source code [49]

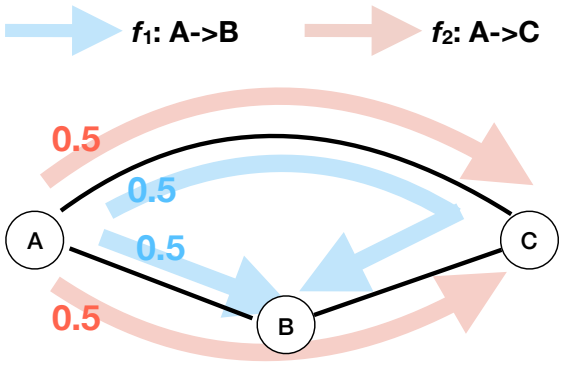


Figure 4.6. In Teavar's design, f_1 and f_2 are both split equally among 2 paths.

Fig. 4.6 shows Teavar's designed routing for the topology in Fig. 4.1 for 99% availability. We can see that in Teavar's design, f_1 is split equally across A-B and A-C-B, and f_2 is split equally across A-C and A-B-C. This way, for 99% of time, both flows will be able to send 0.5 units of traffic. It is also easy to see that when A-B link fails or A-C link fails, the remaining traffic is exactly the same as depicted in Fig. 4.2. Thus, like SMORE, Teavar too cannot support more than 0.5 units 99% of time for both flows.

Flexile's approach. Flexile routes traffic in a manner that exploits inherent network capability. Flexile determines the **critical scenarios** associated with each flow where its loss must be acceptable so the flow objectives can be met. Fig. 4.3 illustrates this for the topology in Fig. 4.1. The critical scenarios associated with f_1 (resp. f_2) are all those scenarios where $A{-}B$ (resp. $A{-}C$) is alive. Clearly, each flow may be associated with a different set of critical failure scenarios.

Unlike ScenBest which seeks to ensure all flows in a scenario see as low a loss as possible. Flexile prioritizes critical flows when allocating bandwidth in any given scenario. In individual scenarios, non-critical flows may see higher loss relative to ScenBest with Flexile. However, Flexile mitigates the penalty through many techniques:

• Our evaluations show that after assigning necessary bandwidth to critical flows, there is significant residual capacity available in each scenario. Flexile judiciously uses this residual capacity to ensure good performance even for non-critical flows in any failure state.

• Flexile supports flows of multiple traffic classes (interactive, and elastic), and ensures favorable treatment for higher priority interactive traffic in all failure states.

• Flexile allows architects to control the loss penalty that non-critical flows may incur in any scenario, trading off the bandwidth guaranteed at a desired percentile. For instance, in Fig. 4.3, if f_1 and f_2 could tolerate an additional loss l in their non-critical scenarios, Flexile can guarantee $0.5 + l$ for both flows 99% of the time.

Flexile on a real-world topology. Fig.4.4 shows a CDF of the 99.9%*ile* loss seen by flows across failure scenarios for Teavar, ScenBest and Flexile for the IBM topology (see §4.4 for evaluation details). There are many flows for which Teavar and ScenBest lead to a significant 99.9%ile loss. Most flows have 40% loss at 99.9%ile with Teavar for reasons outlined in §4.1.1. For 10% of flows ScenBest leads to a 99.9%ile loss of 16% or higher. In contrast, *Flexile ensures all flows see no loss 99.9% of the time.*

Even though Flexile does not explicitly minimize scenario loss, it does not increase this loss much. For concreteness, see Fig.4.5 which shows a CDF of the loss penalty paid by Flexile relative to ScenBest. ScenBest minimizes the loss of the worst performing flow in each scenario. We plot the increase in this loss with alternate schemes. For scenarios that occur 99.9% of time, Flexile incurs no loss penalty. The loss penalty at 99.99%ile is only 4%. In contrast, the loss penalty with Teavar is significant – at least 10% in every scenario, while the 99.9% (resp 99.99%) values are 40% (resp 100%).

4.1.3 Discussion

Flexile can achieve the same bandwidth guarantees with lower network capacity. The above example shows Flexile can meet bandwidth objectives when existing TE schemes cannot. Equivalently, Flexile requires less capacity to be provisioned to meet desired bandwidth objectives. In Fig.4.1, ScenBest and Teavar would require each link to be upgraded by 2X to meet the desired flow bandwidth objectives, while Flexile requires no additional capacity.

Unlike ScenBest, Flexile ensures bandwidth guarantees never degrade with additional links. The bandwidth guarantees provided by ScenBest is not *monotonic* in the

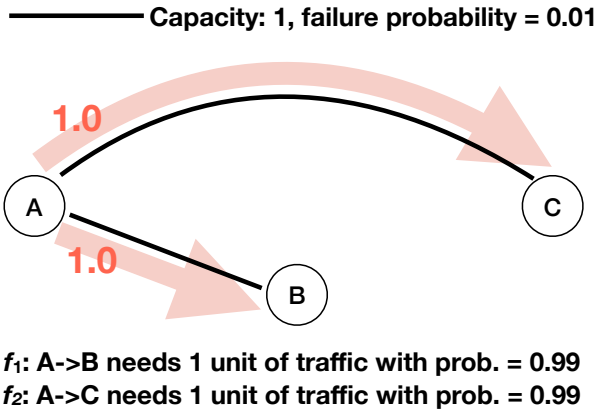


Figure 4.7. ScenBest can meet objectives in above topology but not in Fig.4.1 which has an additional link.

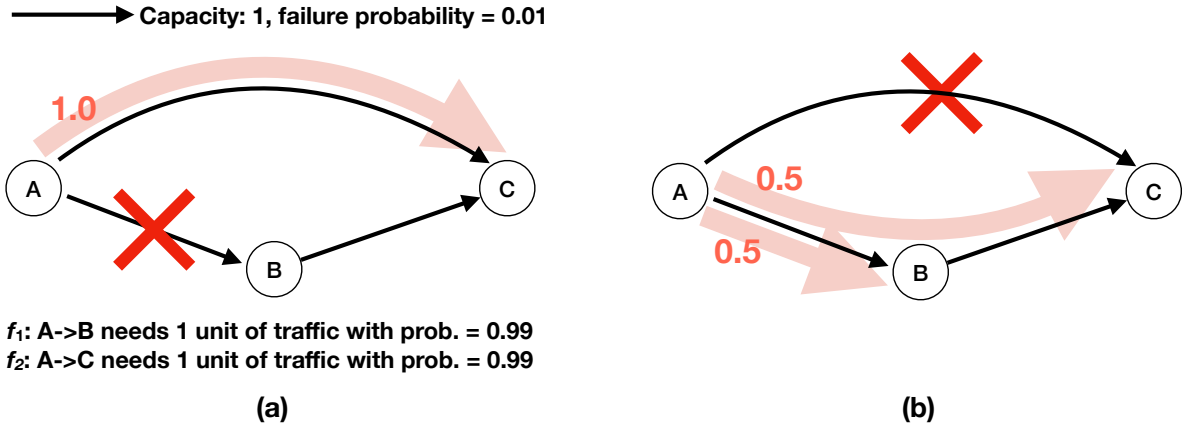


Figure 4.8. Example topology to illustrate unfairness with max-min across scenarios. While the network can meet the 99% requirement for both flows, max-min meets the requirements for f_2 but not f_1.

connectivity of the network, i.e., *adding links to a network topology may result in weaker flow loss guarantees*. Consider Fig. 4.7, which is similar to the topology in Fig. 4.1, except that link $B-C$ is removed. It is easy to verify that ScenBest always routes 1 unit of f_1's traffic on link $A-B$ whenever the link is alive, and likewise always routes 1 unit of f_2's traffic on link $A-C$ whenever that link is alive, thereby meeting the requirements of both flows. Thus, while ScenBest meets flow requirements in Fig. 4.7, it cannot meet requirements in Fig 4.1 which has an additional link. PCF prevents such anomalies since it ensures for any network that all flows see a loss percentile that is as small as possible.

Existing TE schemes can be unfair across scenarios though fair in each; Flexile mitigates this effect. To be fair for different flows in each scenario, one can use max-min

scheme to design the routing. Consider Fig. 4.8, which depicts a topology similar to Fig. 4.1, except that links are directional. Like Fig. 4.1, it is easy to verify that the network can meet the bandwidth objectives of both flows by always routing f_1 along $A{-}B$ and f_2 along $A{-}C$ respectively whenever the appropriate link is alive. However, while max-min ensures 1 unit of f_2's traffic 99% of the time, it cannot meet the requirements of f_1, and can only carry 0.5 units of $f_1's$ traffic 99% of the time. To see this, first consider that max-min always routes f_2's traffic along $A{-}C$ whenever that link is alive which occurs 99% of the time irrespective of failure status of other links. Fig 4.8 (a) illustrates this for an example scenario where $A{-}B$ fails. In contrast, observe that f_1's requirements can only be met when 1 unit of traffic is carried in all scenarios where link $A{-}B$ is alive. However, in the scenario shown in Fig 4.8 (b), max-min only carries 0.5 units of traffic for f_1 although link $A{-}B$ is alive. Further, f_2 is allocated traffic in this scenario although it already meets its requirements through other scenarios. Flexile, however, achieves 99.9%ile loss of 0 for all flows.

4.2 Flexile design

Flexile allocates bandwidth to each flow in every failure scenario so a desired (say β^{th}) percentile of bandwidth loss across flows is minimized. The problem is further complicated since requirements may vary across traffic classes (e.g., a 99.9% requirement for latency-sensitive, and a 99% requirement for other traffic). Minimizing loss at a given percentile (also referred to as Value at Risk or VaR) is a hard problem, and only recently received attention from the networking research community. While it is possible to approximate percentiles as done by Teavar [27], we show in S4.3 that the approximation is weak.

Flexile tackles these challenges through two components:

• **Efficient offline algorithm for determining critical scenarios.** Flexile tackles the hard problem of optimizing flow loss percentiles through a decomposition algorithm that decouples the failure states by identifying the critical states associated with each flow in an offline phase. For efficiency, we have developed several problem-specific accelerations. Our evaluations confirm the algorithmic strategy is efficient.

• **Light-weight online bandwidth allocation to critical and non-critical flows.** On failure, Flexile efficiently allocates bandwidth to all flows while taking particular care of critical flows in addition to favoring higher priority traffic classes. This step also ensures that after critical flows are handled, residual capacity can be appropriately allocated to non-critical flows. To achieve this, we have developed a light-weight adaptation of SWAN [47] that incorporates information about critical flows. However, it is also possible to easily extend other bandwidth allocation mechanisms such as SMORE with information regarding critical flows identified by Flexile.

We start by presenting Flexile's model for optimizing flow loss percentiles, and next discuss the two components above. We then discuss several generalizations related to Flexile.

4.2.1 Optimizing flow loss percentiles

Given bandwidth requirement for a set of flows and a set of failure scenarios with their associated probabilities, Flexile allocates bandwidth to each flow in each failure scenario so that the bandwidth loss at a given percentile is minimized. Further, Flexile models flows corresponding to different traffic classes with different percentile targets for these classes.

Consider a network topology, represented as a graph $G = \langle V, E \rangle$. K represents the set of traffic classes. Each traffic class $k \in K$ is associated with a target probability β_k for which the bandwidth requirement must be met for a set of flows F_k in this class. For instance, high priority traffic may have a 99.9% requirement, while lower priority traffic may have a 99% requirement, reflecting diverse service level objectives (SLOs) that the network supports. Each flow $f \in F_k$ is associated with traffic demand d_f that must be sent along the source-destination pair $pr(f)$. Q represents the set of failure scenarios with p_q denoting the probability of $q \in Q$.

Modeling desired percentile of flow loss. In each traffic class k, for each flow $f \in F_k$, we define $\mathit{FlowLoss}(f, \beta_k)$ to be the β_k^{th} percentile of loss for flow f. That is, there exist failure scenarios that together occur with probability β_k, where flow f encounters a loss less than $\mathit{FlowLoss}(f, \beta_k)$.

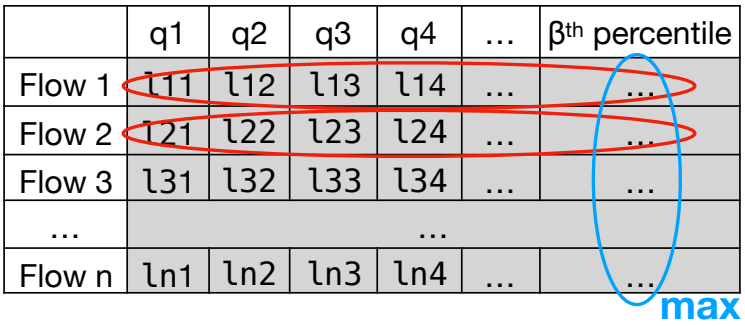


Figure 4.9. Meeting bandwidth requirements requires computing the β^{th} percentile of flow losses.

For each traffic class k, in order to make sure that every flow sees small loss, we are interested in reducing the maximum of the β_k^{th} percentile loss across all flows $f \in F_k$. Specifically, we consider the following metric that we refer to as *PercLoss* (and may abbreviate as α_k).

$$\alpha_k := PercLoss_k = \max_{f \in F_k} FlowLoss(f, \beta) \tag{4.1}$$

Fig. 4.9 depicts this pictorially. Each row corresponds to a flow (f), each column to a failure scenario (q), and each cell shows the bandwidth loss l_{fq} seen by flow f in scenario q. To meet flow level requirements, Flexile computes the β^{th} percentile of each row, computes the max across rows, and minimizes the result.

Considering different traffic classes. Each class $k \in K$ is associated with a weight w_k to compute its penalty for loss, which reflects the relative importance of this class. Thus, the penalty incurred for loss of traffic class k can be represented as $w_k\alpha_k$. We focus on a formulation where Flexile determines a bandwidth allocation such that the sum of penalty across all traffic classes, $\sum_{k \in K} w_k\alpha_k$ is minimized. For instance, a 2 class setting can be handled with a large weight for the higher priority class, and a small weight for the lower priority class. Other priority policies are easily modeled (§4.2.4).

Modeling critical scenarios. To ensure each flow's objectives, Flexile must for each flow $f \in F_k$ select scenarios that together occur with probability β_k such that f sees loss less than α_k in these scenarios. We denote these scenarios as *critical scenarios* for that flow. We

Table 4.1. Notation.

Notation	Meaning
Q	Set of all scenarios
K	Set of traffic classes
F_k	Set of flows in traffic class k
P	Set of source-destination pairs
R_k(i)	Pair i's tunnels for class k
E	Set of edges
α_k	The maximum of the β_k^{th} percentile loss across all flows $f \in F_k$
β_k	Target probability for which the bandwidth requirement must be met for class k
$pr(f)$	Pair along which flow f is sent
d_f	Traffic demand of flow f
p_q	Probability of scenario q
w_k	Weight of traffic class k
y_{tq}	1 if tunnel t is alive in scenario q, else 0
m_{eq}	1 if edge e is alive in scenario q, 0 otherwise
x_{ktq}	Allocated bandwidth on tunnel t for traffic class k in scenario q(routing variable)
l_{fq}	Loss of flow f in scenario q
z_{fq}	1 if scenario q is critical for flow f, else 0

use a binary variable z_{fq} to indicate whether scenario q is critical for flow f. If $z_{fq} = 1$, q is critical for f, and the loss of flow f cannot exceed $PercLoss_k$, i.e., $l_{fq} \leq PercLoss_k$.

$$\sum_{q \in Q} z_{fq} p_q \geq \beta_k \quad \forall k \in K, f \in F_k \tag{4.2}$$

$$\alpha_k \geq l_{fq} - 1 + z_{fq} \quad \forall k \in K, f \in F_k, q \in Q \tag{4.3}$$

Here, (4.2) ensures that for each flow in k, we select enough critical scenarios to cover the probability β_k. When $z_{fq} = 1$, (4.3) becomes $PercLoss_k \geq l_{fq}$ meaning we care about the loss l_{fq}. When $z_{fq} = 0$, (4.3) is satisfied no matter what $PercLoss$ and l_{fq} are, implying we don't care about the loss l_{fq}.

We next present the formulation below which determines the best routing and choice of critical scenarios that minimize the sum of penalty incurred by loss in different traffic

classes. Each link e $\in E$ is associated with a link capacity c_{e}. We use P to represent the set of source-destination pairs. Each pair i in traffic class k can use a set of tunnels $R_k(\mathrm{i})$ to route the traffic. This reflects that different traffic classes may have different routing options and requirements (e.g. background traffic classes can have more tunnel options than delay-sensitive traffic classes). Let y_{tq} represent whether a tunnel t is alive in scenario q. We use x_{ktq} to denote the bandwidth assigned to tunnel t in scenario q for traffic class k, i.e., our designed routing. Table 4.1 summarizes notation.

$$(I) \min_{z,x,l,\alpha} \quad \sum_{k\in K} w_k \alpha_k$$

$$\text{s.t.} \quad (4.2), (4.3)$$

$$\sum_{t\in R_k(\mathrm{i})} x_{ktq} y_{tq} \geq \sum_{pr(f)=\mathrm{i}, f\in F_k} (1-l_{fq}) d_f \quad \forall k \in K, \mathrm{i} \in P, q \in Q \tag{4.4}$$

$$\sum_{k\in K, \mathrm{e}\in t} x_{ktq} \leq c_{\mathrm{e}} \quad \forall \mathrm{e} \in E, q \in Q \tag{4.5}$$

$$x_{ktq} \geq 0 \quad \forall k \in K, \mathrm{i} \in P, t \in R_k(\mathrm{i}), q \in Q \tag{4.6}$$

$$z_{fq} \in \{0,1\} \quad \forall k \in K, f \in F_k, q \in Q \tag{4.7}$$

$$0 \leq l_{fq} \leq 1 \quad \forall k \in K, f \in F_k, q \in Q. \tag{4.8}$$

(4.4) ensures that there is enough bandwidth allocated to each pair. The LHS of (4.4) is the total amount of traffic required to be sent on pair i, and the RHS is the total allocated bandwidth on tunnels connecting pair i. This constraint was modeled like [27]. (4.5) and (4.6) ensure the allocated bandwidth on tunnels will not exceed any link's capacity, and the allocation is non-negative. The final two constraints indicate the z variables are binary, and ensure the loss fractions are between 0 and 1.

4.2.2 Efficiently finding critical scenarios

The above problem is a Mixed Integer Program (MIP), which can be challenging to solve. To tackle this, we tailor a systematic decomposition strategy [50], to our domain with many

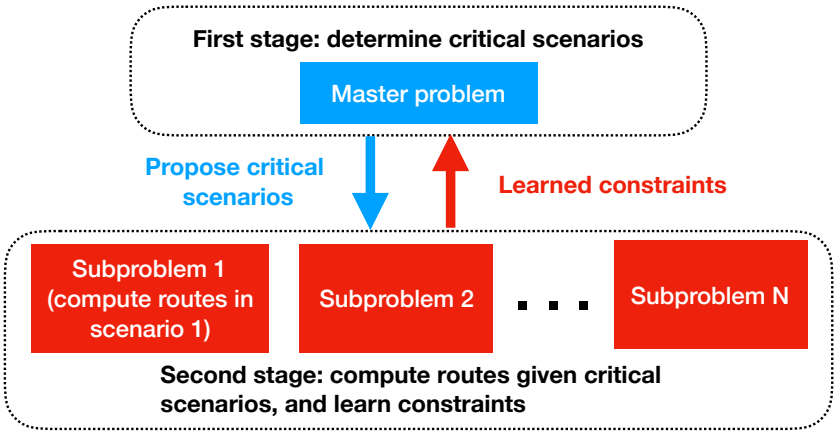


Figure 4.10. Systematic decomposition approach.

problem specific optimizations to enable faster convergence, and reduce running times. We discuss the basic strategy, followed by our optimizations.

Basic decomposition strategy. The original problem (I) simultaneously determines (i) the critical scenarios for each flow; and (ii) how the traffic should be routed in each failure scenario taking into account for which flows that scenario is critical. Instead, we decompose the problem into (i) a master problem that proposes the critical scenarios for each flow; and (ii) a sub-problem which routes traffic when given the proposed set of critical scenarios for each flow. The sub-problem learns new constraints that are added to the master problem, which then proposes another set of critical scenarios. By iterating, the process converges finitely with an optimal solution.

We now discuss optimizations over the standard approach.

Subproblem decomposition. Instead of writing the subproblem as a large LP, we observe that the subproblem can be decomposed into multiple subproblems, since routing in each scenario can be derived independently of other scenarios. Each smaller subproblem determines routing for one failure scenario given critical flows for that scenario. Each second stage subproblem provides the learned constraints to the master problem so that the master can alter its critical scenario proposal in the next iteration. Each LP subproblem is small and solves quickly. Moreover, further speed up is attained by solving the subproblems in

parallel. Fig. 4.10 illustrates our procedure. For each scenario q, we have the following smaller subproblem (note that z_{fq} is a parameter here):

$$(S_q) \min_{x,l,\alpha} \quad \sum_{k\in K} \alpha_k w_k$$

$$\text{s.t. } \alpha_k \geq l_{fq} - 1 + z_{fq} \quad \forall k \in K, f \in F_k \tag{4.9}$$

$$0 \leq l_{fq} \leq 1 \quad \forall k \in K, f \in F_k \tag{4.10}$$

$$\sum_{t\in R_k(\mathrm{i})} x_{ktq} y_{tq} \geq \sum_{f\in F_k, pr(f)=\mathrm{i}} (1 - l_{fq}) d_f \quad \forall k \in K, \mathrm{i} \in P \tag{4.11}$$

$$\sum_{k\in K, \mathrm{e}\in t} x_{ktq} \leq c_{\mathrm{e}} \quad \forall \mathrm{e} \in E \tag{4.12}$$

$$x_{ktq} \geq 0 \quad \forall k \in K, \mathrm{i} \in P, t \in R_k(\mathrm{i}). \tag{4.13}$$

Formally, we rewrite (I) as

$$(I') \min_{z} \quad \mathit{Penalty}(z) \text{ s.t. } (4.2), (4.7) \tag{4.14}$$

$$\mathit{Penalty}(z) = \min_{x,l,\alpha} \sum_{k\in K} \alpha_k w_k \text{ s.t. } (4.3), (4.4), (4.5), (4.6), (4.8) \tag{4.15}$$

We present some high-level intuition for the inner workings of the decomposition approach. Indeed, the optimal objective value for the inner problem (4.15) is convex in z. This follows from the fact that if, for all i, $(x^{\mathrm{i}}, l^{\mathrm{i}}, \alpha^{\mathrm{i}})$ is feasible when $z = z^{\mathrm{i}}$, then for some multipliers $\lambda_{\mathrm{i}} \geq 0$ such that $\sum_{\mathrm{i}} \lambda_{\mathrm{i}} = 1$, the solution $\sum_{\mathrm{i}} \lambda_{\mathrm{i}}(x^{\mathrm{i}}, l^{\mathrm{i}}, \alpha^{\mathrm{i}})$ is feasible when $z = \sum_{\mathrm{i}} \lambda_{\mathrm{i}} z^{\mathrm{i}}$. This shows that, the optimal value of the inner problem at z is no more than $\sum_{\mathrm{i}} \lambda_{\mathrm{i}} \alpha^{\mathrm{i}}$. The dual form of (4.15) provides a cut of *PercLoss*(z) because its feasible region does not depend on z). The decomposition algorithm essentially searches for the minimizer of *Penalty*(z) iteratively. Although the exact shape of *Penalty*(z)'s is unknown at any point in the algorithm, solving (4.15) gives us one point on the function *Penalty*(z). Moreover, the dual form of (4.15) provides a cut of *Penalty*(z). Thus, we can derive an underestimation of *Penalty*(z) by evaluating it at various z. Each cut is a lower bound of function *Penalty*(z), and the pointwise maximum of these cuts is an underestimate of *Penalty*(z). Then, we can find the current estimated minimizer of the estimated function. Solving (4.15) at the estimated minimizer

gives a new cut and a more accurate estimate of $Penalty(\mathrm{z})$. The process converges in finite time with an optimal solution. To understand why, consider that the inner problem is always feasible with $(x, l, \alpha) = (0, 1, 1)$ and bounded between 0 and 1. If we use an extreme point of the dual feasible region to generate a cut (as is the case using dual simplex algorithm), in finitely many iterations, a cut is developed for each extreme point, and we have an accurate representation of $PercLoss(\mathrm{z})$.

Reformulating the subproblem. To achieve further speed ups, we reformulate each subproblem S_q to make the LHS of the constraints the same across all scenarios, so the only change is in the RHS. This ensures that the the dual solution space is common across the LPs for different scenarios. This allows LP solvers to memoize the intermediate results from solving one scenario to speed up the solution of the next scenario. Specifically, we rewrite (4.11) and (4.12) as:

$$\sum_{t \in R_k(\mathrm{i})} x_{ktq} \geq \sum_{f \in F_k, pr(f)=\mathrm{i}} (1 - l_{fq}) d_f \quad \forall k \in K, \mathrm{i} \in P \tag{4.16}$$

$$\sum_{k \in K, \mathrm{e} \in t} x_{ktq} \leq c_{\mathrm{e}} m_{eq} \quad \forall \mathrm{e} \in E. \tag{4.17}$$

Rather than y_{tq} variables that capture tunnel failure, our reformulation introduces m_{eq} variables which represent whether an edge e is alive in scenario q. The reformulation adjusts the capacity of failed links based on their failure state rather than cancel allocations on failed tunnels. These changes ensure only the RHS varies for different scenario q.

Master problem with decomposed subproblems We next present the master problem which derives an underestimate of the minimal penalty. This is improved by adding cut constraints learnt from solutions of the dual of S_q.

$$\begin{aligned} (M) \quad & \min_{z, Penalty} \quad Penalty \\ & \text{s.t. } (4.2), (4.7) \\ & \quad Penalty \geq g_q(z_{\cdot q}) \quad \forall g \in G, \forall q \in Q. \end{aligned} \tag{4.18}$$

G represents the set of all cuts computed so far. Note that since the subproblem is decomposed into (S_q) by scenarios, each $g \in G$ is expressed as a set of cuts $g_q(z_{\cdot q})$, each constraining critical flows in one scenario q. Suppose we get a dual solution of (S_q), and the dual variables of (4.9), (4.10), (4.16) and (4.17) are w_{kfq}, o_{kfq}, v_{kiq} and u_{eq} respectively. Then, solving (S_q) results in the following cuts which are added to the master problem in the subsequent iteration.

$$g_q(z_{\cdot q}) = \sum_{k,f\in F_k} (z_{fq} - 1) w_{kfq} + \sum_{k,f\in F_k} o_{kfq} + \sum_{k,\mathrm{i},f\in F_k, pr(f)=\mathrm{i}} v_{\mathrm{kiq}} d_f + \sum_{\mathrm{e}} u_{\mathrm{eq}} c_{\mathrm{e}} m_{\mathrm{eq}} \tag{4.19}$$

Our reformulation ensures a common dual solution space for all decomposed subproblems. Thus, the dual solution w_{kfq}, o_{kfq}, v_{kiq} and u_{eq} of (S_q) is also a dual solution of $(S_{q'})$ for any $q' \in Q$. So we can construct the following cut to constraint critical flows in scenario q' without solving $(S_{q'})$.

$$g^q_{q'}(z_{\cdot q'}) = \sum_{k,f\in F_k} (z_{fq'} - 1) w_{kfq} + \sum_{k,f\in F_k} o_{kfq} + \sum_{k,\mathrm{i},f\in F_k, pr(f)=\mathrm{i}} v_{\mathrm{kiq}} d_f + \sum_{\mathrm{e}} u_{\mathrm{eq}} c_{\mathrm{e}} m_{\mathrm{eq}'} \tag{4.20}$$

Ensure better stability. To speed up convergence and avoid oscillations around the optimal, we restrict the step when we update z. We achieve this by adding the following constraint in (M) to limit the hamming distance between current z variable and z variable achieved from last iteration.

$$\sum_{k\in K, f\in F_k, q\in Q} |z_{fq} - z'_{fq}| \leq Limit. \tag{4.21}$$

Here, z' is the z variable achieved from last iteration and *Limit* is the maximal hamming distance we allow. Another benefit of constraining z's change over iterations is that more scenarios will have the same critical flows as in the last iteration. So the subproblem (S_q) will stay unchanged for these scenarios and does not need to be solved again. The Hamming

distance constraint can be relaxed to prevent the solution from getting stuck in a local minima. However, we did not encounter this situation in our empirical evaluations.

Pruning scenarios. We further accelerate the decomposition strategy by recognizing that not all subproblems need to be solved each iteration. First, we prune out *perfect scenarios* where all flows can be simultaneously handled without loss. Second, we prune out scenarios for which the set of critical flows does not change. Third, our reformulation of (S_q) discussed earlier ensures that in (S_q) only the RHS varies for different q, and, so, all (S_q) share the same dual solution space. Thus, by solving only each (S_q) optimally, not only do we get a cut $g_q(z_{\cdot q})$ for scenario q, but also cuts $g^q_{q'}(z_{\cdot q'})$ for other scenarios $q' \in Q$. As a result, solving a few subproblems can give us many cuts.

Identifying a good starting point. It is desirable to start with a good initial choice of z so that the algorithm requires fewer iterations to converge. We observe that a flow must be connected in a failure scenario for that scenario to be critical. Thus, we add constraints $z_{fq} = 0$ in (M) if flow f is disconnected in scenario q, and $z_{fq} = 1$ otherwise. We have the proposition below which indicates this heuristic is a good starting point.

Proposition 4.2.1. *At the initial step of our algorithm (prior to any iteration of the master), the guarantee from our algorithm is already at least as good as TeaVar or ScenBest.*

Proof. Let α_q denote the maximum loss across all flows in scenario q, *i.e.*, α_q is the optimal value of S_q with $z_{fq} = 1$ for all f. Let Q' be any minimal subset of Q such that $\sum_{q' \in Q'} p_{q'} \geq \beta$ and for $q' \in Q'$ and $q \notin Q'$, $\alpha_q \geq \alpha_{q'}$. Then, we define $v = \max_{q' \in Q'} \alpha_{qv}$, which is the β^{th} percentile of $(\alpha_q)_{q \in Q}$. In our first step of the algorithm, we set $z_{fq'} = 1$ for all f and $q' \in Q'$. By definition, $\sum_{q \in Q'} p_q z_{aq} \geq \beta$. In particular, for each flow f and $q \in Q'$, $l_{fq} \leq v$. Therefore, for each f, the β^{th} percentile of $l_{fq} \leq v$. So, our performance guarantee, which is the maximum across all f of the β^{th} percentile of l_{fq} is no more than v. To see that TeaVar guarantees a performance no better than v, let x_t be the routing strategy obtained using TeaVar and observe that the maximum loss across all flows using x_t for a scenario q is at least α_q. Let $r = (1-\beta) - \sum_{q' \notin Q'} p_q$, $\bar{q} \in Q'$ be any scenario with $\alpha_{\bar{q}} = v$, and s be the corresponding optimal $s_{\bar{q}}$ (in TeaVar formulation). Then, observe that $r \leq p_{\bar{q}}$ and $\alpha + s \geq \alpha_{\bar{q}} = v$, where the inequality follows because there is at least one flow with a loss of $\alpha_{\bar{q}}$ since $\alpha_{\bar{q}}$ is the

minimum possible loss attainable across all flows for scenario $\bar{q}$. Then, we have that TeaVar objective is no less than $\alpha + \frac{1}{1-\beta}\sum_{q' \in Q'} p_{q'} s_{q'} + \frac{1}{1-\beta} rs \geq \frac{1}{1-\beta}\left(\sum_{q' \in Q'} p_{q'} \alpha_{q'} + rv\right) \geq v$, where the first inequality is because $\sum_{q' \in Q'} p_{q'} + r = 1 - \beta$, $\alpha + s_{q'} \geq \alpha_{q'}$, and $\alpha + s \geq v$. The second inequality is because $\sum_{q' \in Q'} p_{q'} + r = 1 - \beta$ and $\alpha_{q'} \geq v$ for $q' \in Q'$. Moreover, ScenBest guarantees a loss of v, since the guarantee for flows in any scenario q' not in Q' is $\alpha_{q'}$. It follows that the guarantee from the initial step of our algorithm is at least as good as the one obtained from either ScenBest or TeaVar. $\square$

Algorithm 1 summarizes our decomposition algorithm (Line 17-19 can be executed in parallel). We remark that in each iteration, the algorithm yields a routing strategy, and the corresponding *Penalty* can be computed easily by sorting the optimal values for (S_q) and computing the β^{th} percentile.

4.2.3 Critical flow-aware online allocation

The offline phase identifies the critical failure scenarios for each flow and guides which flows to prioritize in the online phase. When a failure occurs, necessary bandwidth is first allocated to critical flows. However, there is typically significant residual capacity remaining, which Flexile then allocates to non-critical flows while also favoring high priority traffic.

To achieve this, Flexile uses an adaptation of SWAN's max-min allocation algorithm [47], but with some important changes. A first major change is that Flexile assigns necessary bandwidth for critical flows as pre-decided by the offline phase. Then, a max-min approach is used to allocate bandwidth to non-critical flows, and additional bandwidth beyond the pre-determined minimum to critical flows. Like [47], allocations are first done for higher priority traffic classes.

Second, SWAN determines the allocation for each traffic class, as well as the routing, before allocating residual capacity to a lower class. We implement an optimization where we decide how much traffic the higher class gets, but do not pre-determine the routes. When solving for the lower priority class, we force a minimum required allocation for the higher priority class, and then simultaneously determine (i) the routing for both classes; and (ii)

Algorithm 1 Decomposition algorithm

```
1: function solve_master(G, z')
2:     Add hamming distance constraint with z' to (M)
3:     Solve (M) with G, and get new variable z
4:     return z
5: end function
6: function solve_subproblem(z,q)
7:     Solve (S_q) and construct cut constraint g_q
8:     return x_q, g_q
9: end function
10: function MAIN(max_iterations)
11:     Initialize z_fq to be 1 if f is connected in q and 0 otherwise
12:     Initialize x_q to be ∅ for all q ∈ Q
13:     cur_iteration ← 0
14:     G ← ∅
15:     while cur_iteration < max_iterations do
16:         g ← ∅
17:         for q ∈ Q do
18:             if q cannot be pruned then
19:                 x_q, g_q ← solve_subproblem(z, q)
20:                 g.add(g_q)
21:             end if
22:         end for
23:         G.add(g)
24:         z ← solve_master(G, z)
25:         cur_iteration ← cur_iteration + 1
26:     end while
27:     return x
28: end function
```

the allocation for flows in the lower priority class. Third, rather than do max-min allocations on bandwidth, we instead consider flow loss, and do a max-min allocation on flow loss.

More generally, Flexile can work with any online bandwidth allocation algorithm, not just SWAN, depending on the secondary design objective beyond minimizing flow loss percentiles. For instance, formulation (S_q) could be used online to allocate traffic so as to minimize the weighted loss of high and low priority flows given a set of critical flows, while in single class settings, ScenBest could be easily augmented to minimize MLU while prioritizing critical flows.

4.2.4 Generalizations

Constraining loss on non-critical flows. While §4.2.3 already ensures non-critical flows may be allocated bandwidth using residual capacity, we may explicitly constrain loss on non-critical flows in each scenario through a small change to (I). Suppose for scenario q, the optimal *ScenLoss* is $loss_q$. We can add constraints of the form $l_{fq} \leq \gamma + loss_q$, where γ is a constant representing the maximum factor by which the flow's loss may increase in that scenario. γ then serves as a knob that trades off the increase the flow sees in that scenario with *PercLoss*. By setting γ appropriately, we can ensure optimal performance in each scenario.

More general scenarios. Beyond link failures, Flexile can easily model Shared Risk Link Groups (**SRLGs**) where a group of links fail together. Scenarios now correspond to SRLG failures, and for each scenario q, the m_{eq} parameters capture the links of the SRLGs that fail. It is also easy to extend Flexile to design for a set of traffic matrices given their probability. In model (I), each scenario $q \in Q$ corresponds to a traffic matrix. The demand of flow f in (4.4) will become d_f^q, reflecting different traffic matrices in different scenarios. Flexile's decomposition algorithm still applies.

Explicit priority with multiple traffic classes. In Flexile, by altering the weight in the objective, more emphasis can be placed on *PercLoss* for high-priority traffic. Futher, our online allocation algorithm favors high-priority traffic when using residual capacity, which usually ensures high-priority traffic does not see loss across scenarios (Fig 4.15). If the

PercLoss of low-priority traffic is even subordinate to sending high-priority traffic in a non-critical scenarios then Flexile can be adapted as follows. First, Flexile determines critical flows to minimize *PercLoss* only considering high-priority traffic. Then Flexile uses the algorithm in §4.2.3 to push as much non-critical high-priority traffic as possible in each scenario. Next, Flexile may be used to design for low-priority traffic with additional constraints to meet bandwidth levels for high-priority traffic determined in the first step. The approach is easily generalized to multiple traffic classes.

Capacity augmentation to meet flow percentile requirements. Flexile can be generalized to perform minimum-cost capacity augmentation on the network which is more cost-effective than a scenario-centric approach. To do this, we may require that, for each $k \in K$, $PercLoss_k$ is constrained to be below a specified value and minimize $\sum_e w_e \delta_e$, where δ_e is the added capacity to link e, which changes the RHS of (4.5) to $c_e + \delta_e$, and w_e is the per-unit cost of adding capacity. (If there is a fixed-cost, we can include it by introducing a binary variable a_e which takes value 1 if link e is augmented, and add $\sum_e f_e a_e$ to the cost. To ensure fixed-cost is charged with any augmentation, we add upper-bounding constraints $0 \le \delta_e \le u_e a_e$, where u_e is an upper bound on the augmentation.) The decomposition strategy of §4.2.2 generalizes to this setting where c_e is replaced with $c_e + \delta_e$ in (4.19) and this cut now describes a cut of *Penalty* in the (z, δ) space.

4.3 Flexile Vs. Teavar

While Flexile minimizes the β^{th} percentile of losses (or Value at Risk or VaR), Teavar [27] approximates the same using the *Conditional Value at Risk (CVaR)*. CVaR minimizes the expected loss of the worst $(100 - \beta)^{th}$ percentile of scenarios. For example, consider a flow which sees a loss of 0%, 5% and 10% in three scenarios that respectively occur with probability 0.9, 0.09, and 0.01. Then, the 90^{th} percentile loss (VaR) is 0%, but the CVaR is $5 * 0.09 + 10 * 0.01 = 1.45\%$.

Recall there are two other differences between Teavar and Flexile. First, Teavar considers the β^{th} percentile of *ScenLoss*, unlike Flexile which focuses on the β^{th} percentile of flows. Second, on failure, Teavar rescales traffic of each source destination pair on live tunnels so

the same proportion is maintained. In contrast, Flexile like SMORE [30] allows greater flexibility in how traffic is split across tunnels.

To analyze the advantages of directly considering VaR in Flexile, and decouple these benefits from other benefits of Flexile, we design two new CVaR-based TE schemes, which may be viewed generalizations of Teavar:

• *Cvar-Flow-St.* Here, we use CVaR to approximate the computation of *PercLoss*. Instead of directly computing β^{th} percentile loss for flow f, i.e., $\mathit{FlowLoss}(f, \beta)$, we use CVaR of flow f (denoted by $CVaR(f, \beta)$) to approximate it. Then we seek to optimize the maximum CVaR of all flows, which we denote as *MaxFlowCVaR*. Formally,

$$\mathit{MaxFlowCVaR} = \max_{f \in F} CVaR(f, \beta) \tag{4.22}$$

. • *Cvar-Flow-Ad.* This is similar to *Cvar-Flow-St* except that we allow greater flexibility in terms of how traffic may be split across tunnels on failure.

We develop Linear Programming (LP) models for computing the routing and bandwidth allocations associated with these schemes. The following is the formulation for *Cvar-Flow-Ad*.

$$\begin{aligned}
\min_{x,t,\theta,\alpha,s} \quad & \theta \\
\text{s.t.} \quad & \theta \geq \theta_f \quad \forall f \in F && (4.23)\\
& \theta_f \geq \alpha_f + \frac{1}{1-\beta} \sum_{q \in Q} p_q s_{fq} \quad \forall f \in F && (4.24)\\
& \alpha_f + s_{fq} \geq l_{fq} \quad \forall f \in F, q \in Q && (4.25)\\
& s_{fq} \geq 0 \quad \forall f \in F, q \in Q && (4.26)\\
& (4.4), (4.5)
\end{aligned}$$

Here, l_{fq} is the loss for flow f in scenario q, θ_f models the conditional value-at-risk for flow f, and θ models $\max_f \theta_f$.

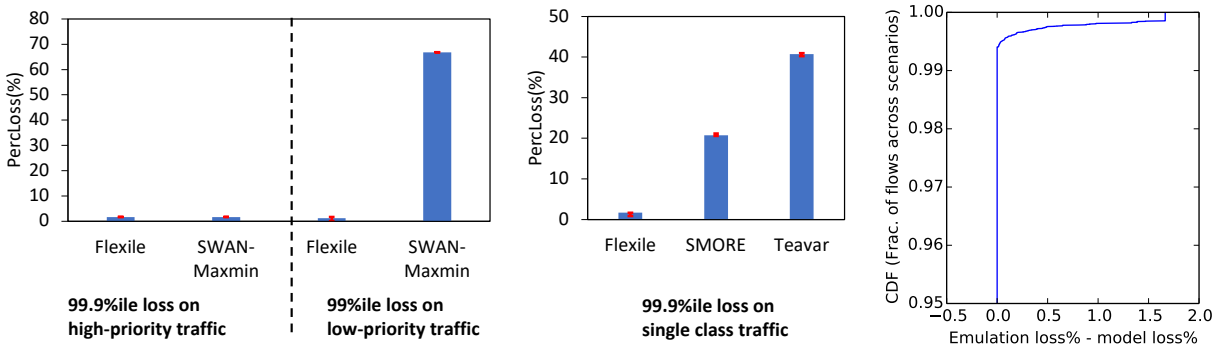
The following formulation, CVar-Flow-St, is derived from CVar-Flow-Ad by requiring that the routing strategy is the same across all scenarios, *i.e.*, we add the requirement that $x_{tq} = x_t$ for all q. More concretely, we obtain:

$$\begin{aligned}
\min_{x,t,\theta,\alpha,s} \quad & \theta \\
\text{s.t.} \quad & \theta \geq \theta_a \quad \forall f \in F && (4.27)\\
& \theta_f \geq \alpha_f + \frac{1}{1-\beta} \sum_{q \in Q} p_q s_{fq} \quad \forall f \in F && (4.28)\\
& \alpha_f + s_{fq} \geq l_{fq} \quad \forall f \in F, q \in Q && (4.29)\\
& s_{fq} \geq 0 \quad \forall f \in F, q \in Q && (4.30)\\
& \sum_{pr(f)=\mathrm{i}} (1 - l_{fq}) d_f \leq \sum_{t \in R(\mathrm{i})} x_t y_{tq} \quad \forall \mathrm{i} \in P, q \in Q && (4.31)\\
& \sum_{\mathrm{e} \in t} x_t \leq c_{\mathrm{e}} \quad \forall \mathrm{e} \in E && (4.32)\\
& x_t \geq 0 \quad \forall \mathrm{i} \in P && (4.33)
\end{aligned}$$

We have the following proposition showing that these more general strategies are still quite conservative, and there is significant potential to doing better with Flexile by directly considering VaR.

Proposition 4.3.1. *There exists a setting where PercLoss found by Teavar, and all CVaR strategies is at least 48% even though there exists an optimal strategy where the network can achieve an PercLoss of zero.*

Proof. Refer to Fig. 4.1. Consider a strategy that distributes f_{AB} equally over disjoint paths $A{-}B$ and $A{-}C{-}B$. Similarly, f_{AC} is distributed equally along the disjoint paths $A{-}C$ and $A{-}B{-}C$. Since each flow is carried along two disjoint paths, it follows that in all scenarios where at most one link fails, none of the flows experiences a loss of more than 50%. Since single and no link failures cover 0.999702 probability, it follows that the CVar for this strategy is no more than $0.5 * 0.9702 + (1 - 0.9702) = 0.5149$. Observe that the strategy described above is non-adaptive and the best adaptive strategy cannot perform worse. In other words, optimal CVar is no more than 0.5149. Now, consider the case where link $A{-}C$



(a) (b) (c)

Figure 4.11. (a) Emulation testbed results. (a) Flexile vs. SWAN. (b) Flexile vs. Teavar and SMORE. (c) Comparing flow losses across scenarios in emulations with model predicted losses.

fails. Since CVar is the maximum expected loss across all flows and all sets of scenarios that occur with 1% or more probability, it follows that $0.5149 \geq CVar \geq 1 - \min\{f_{AB}, f_{AC}\}$ which implies that $\min\{f_{AB}, f_{AC}\} \geq 0.4851$. Since, both f_{AB} and f_{AC} must use link $A-B$, we have $\min\{f_{AB}, f_{AC}\} + \max\{f_{AB}, f_{AC}\} = f_{AB} + f_{AC} \leq 1$. It follows that $\max\{f_{AB}, f_{AC}\} \leq 0.5149$ which implies that both the flows experience at least 48.51% loss in this scenario. A similar argument shows that both flows experience at least 48.51% loss also in the scenario where link $A-C$ fails. Since the two scenarios cover a probability of 1.9602%, it follows that $PercLoss_{1\%} \geq 0.4851$. The alternate non-adaptive strategy that sends f_{AB} along link $A-B$ and f_{AC} along link $A-C$ experiences no loss at the 99^{th} percentile since each of the links does not fail with 0.99 probability. □

4.4 Evaluations

We compare Flexile with state-of-the-art TE schemes on multiple topologies, and validate the results on an emulation testbed. We discuss our methodology and then results.

Schemes compared. We compare Flexile with

• ***Teavar and other CVaR schemes:*** We consider Teavar and two enhanced CVaR schemes (*Cvar-Flow-St* and *Cvar-Flow-Ad*) that we developed (§4.3). These schemes enable us to separate Flexile's benefits related to directly optimizing loss percentiles (rather than approximate with CVaR), and its benefits related to considering flow losses.

Table 4.2. Topologies used in evaluation (Flexile).

Topology	# nodes	# edges	Topology	# nodes	# edges
B4	12	19	Janet Backbone	29	45
IBM	17	23	Highwinds	16	29
ATT	25	56	BTNorthAmerica	36	76
Quest	19	30	CRLNetwork	32	37
Tinet	48	84	Darkstrand	28	31
Sprint	10	17	Integra	23	32
GEANT	32	50	Xspedius	33	47
Xeex	22	32	InternetMCI	18	32
CWIX	21	26	Deltacom	103	151
Digex	31	35	IIJ	27	55

• ***SMORE:*** SMORE split traffic optimally among live tunnels upon failures. This is identical to ScenBest discussed in section §4.1 when the optimized metric is MLU.

• ***SWAN:*** We consider both variants of SWAN [47], which we refer to as SWAN-Throughput and SWAN-Maxmin. For each scenario, both schemes allocate bandwidth to higher priority traffic classes before lower priority ones. SWAN-Throughput maximizes throughput while SWAN-Maxmin uses an iterative algorithm to approximate max-min fairness.

We include SWAN because like Flexile, it can handle multiple traffic classes. In contrast, Teavar and SMORE are designed for single traffic class. Thus, our comparisons with SWAN are based on two traffic classes (a latency-sensitive class, and a lower priority class), while the comparisons with SMORE and Teavar consider a single traffic class.

When feasible (for smaller topologies), we also compare Flexile with *IP*, which uses the optimal routing designed by the MIP formulation (I). Our implementation of Flexile includes both the decomposition algorithm (§4.2.2) for the offline phase (run for a maximum of 5 iterations), and the online phase run on failure. We implement all our optimization models in Python, and use Gurobi 8.0 [38] to solve them.

Performance metric. Our primary performance metric for all schemes is the *PercLoss* for each class achieved by the scheme (i.e., we consider the β^{th} percentile of loss of each flow in a class, and take the maximum across flows.). We evaluate all the schemes based on post-analysis. For each scheme, we determine the routing and bandwidth allocation in each failure scenario, compute the loss of each flow in each scenario, and then compute *PercLoss*.

Topologies and traffic model. We evaluate the schemes on 20 topologies obtained from [24] and [30] (see Table 4.2). Our largest network contains 151 edges and 103 nodes. We remove one-degree nodes in the topologies recursively so that the networks are not disconnected with any single link failure. We choose tunnels balancing latency and disjointness like prior works [10], [27], [48]. For latency-sensitive high-priority traffic we choose three shortest paths that are not disconnected by single link failures. For low-priority traffic which is not as latency-sensitive, we add three additional tunnels from a larger pool of shortest paths prioritizing disjointness. Our single class experiments use three physical tunnels per pair that are as disjoint as possible, preferring shorter ones when there are multiple choices. We used the gravity model [39] to generate traffic matrices with the utilization of the most congested link (MLU) in the range [0.5, 0.7] across all topologies. The resulting traffic matrix was used as such for the single traffic class experiments. For the two-class experiments, the traffic of each pair was randomly split into high and low priority. We then scaled low priority traffic by a factor of 2 given the network can run closer to saturation with low priority traffic.

Failure scenarios. For each topology, we use the Weibull distribution to generate the failure probability of each link, like prior work [27]. We choose the Weibull parameter so that the median failure probability is approximately 0.001, matching empirical data characterizing failures in wide-area networks [2], [32], [51]. Given a set of link failure probabilities, we sample failure scenarios based on the probability of the occurrence. Our evaluations assume independent link failures but Flexile's approach easily generalizes to shared risk link groups (§4.2.4). We discard scenarios with insignificant probability ($< 10^{-6}$). For single-class experiment, our design target is set as high a probability target as possible for which all flows in the network remain connected for the sampled scenarios since the network will trivially see a *PercLoss* of 1 when designing for a higher target. We also use this as the design target for high-priority class in two-class experiments. For low-priority class, we always use 0.99 as the design target.

Emulation setup. Our emulation experiments are conducted on a Mininet cluster [52] running on six Cloudlab servers [53]. Link bandwidths were set to 10 Mbps to avoid software switch bottlenecks. The traffic demands generated using the approach above was accordingly normalized. Tunneling was implemented using MPLS labels. We emulate the performance

of a TE scheme in a failure scenario by starting the network in the normal condition and failing the appropriate set of links. The source switch uses select groups supported by Open vSwitch, and weights are set so each tunnel is chosen with a probability determined by the appropriate TE scheme. The TE scheme also determines how much data each flow is permitted to transmit. We measure the loss seen by each flow on the emulation testbed relative to the original demand requested, accounting for both throttling required by the TE scheme, and losses in the testbed. We compute loss at a desired percentile for each flow given the emulated losses for each scenario, and its probability, which in turn enables us to compute the *PercLoss* for each traffic class.

4.4.1 Comparisons on emulation testbed

We emulate the IBM topology which has 17 switches, and 23 links, generating necessary traffic using 34 end hosts. The comparisons with SWAN used two traffic classes for all 272 pairs (544 flows in all), while the comparisons with Teavar and SMORE used a single traffic class for all pairs. Each scheme was emulated in each of 138 sampled scenarios (which cover more than 99.992% probability) five times.

Flexile vs. SWAN. Fig. 4.11a shows *PercLoss* achieved by Flexile and SWAN-Maxmin on the IBM topology for both high and low priority traffic. Each bar shows the median *PercLoss* across 5 runs. The error bars show the minimum and maximum. For high priority traffic, we consider the 99.9%*ile* loss of each flow, while for low priority traffic, we consider the 99%*ile* of each flow. The figure shows that *PercLoss* is nearly zero for both schemes for high priority traffic indicating all high priority flows can be sustained without loss 99.9% of the time. However, while *PercLoss* is nearly zero for low priority with Flexile, it is fairly high ($> 60\%$) for SWAN-Maxmin. This indicates that Flexile can carry all low priority flows with almost no loss 99% of the time, but some flows may see large loss with SWAN-Maxmin 99% of the time.

Flexile vs. SMORE and Teavar. Fig. 4.11b compares Flexile with SMORE and Teavar using a single traffic class considering the 99.9%*ile* loss for flows. The *PercLoss* with Flexile is nearly zero indicating it can support all flows with minimal loss 99.9% of the time,

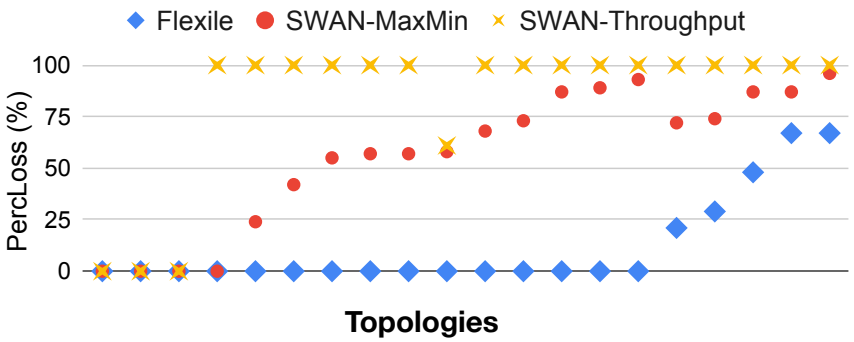


Figure 4.12. Flexile Vs. SWAN. Flexile matched optimal whenever it was computable. Vertically aligned dots correspond to the same topology.

while the *PercLoss* achieved by SMORE and Teavar is 17% and 40% respectively indicating some flows could see significant 99.9%*ile* loss.

Models vs. Emulation. While our models assume continuous split ratios and traffic demands, Open vSwitch only takes integer weights in select groups, and some discretization occurs since testbed traffic is packet-based. To assess the impact of such discretization, we compare the losses observed in the emulations, and losses predicted by the optimization models of TE schemes across all flows and all scenarios using the Pearson Correlation Coefficient (PCC). The PCC values are more than 0.999 in both single-class and two-class setting. Fig. 4.11c shows a CDF of the losses observed in emulation and simulation across all flows and scenarios. There is no difference in over 99% of the cases, and a difference of less than 1.67% in all cases. For all schemes, and all runs, *PercLoss* in the emulations is within 1.67% of the models, which is much smaller than the performance gap across the schemes. These results indicate that the emulation results closely match our optimization models.

4.4.2 Comparisons across topologies

Flexile vs. SWAN. We compare Flexile with both SWAN variants – SWAN-Throughput and SWAN-Maxmin. For high priority traffic, all schemes achieve *PercLoss* of zero across all topologies. Fig. 4.12 compares the *PercLoss* for low priority traffic across topologies. Clearly, Flexile significantly outperforms both SWAN variants for most topologies. The median *PercLoss* across topologies for Flexile is 0%, while the median for SWAN-Maxmin is 58%. In some cases, SWAN-Maxmin sees *PercLoss* as high as 93%. Interestingly, SWAN-

Throughput sees extremely high *PercLoss* of 100% in many cases (median across topologies is 100%). This is because optimizing throughput may lead to significant unfairness across flows. Some flows may be consistently sacrificed without any demand serviced in many scenarios. As an example, consider a path A-B-C with each link having unit capacity. Here, SWAN-Throughput would prioritize sending one unit of demand for the AB and BC flows, and allocate no traffic to the AC flow, as this maximizes throughput.

Flexile vs. Teavar and our CVaR variants. Fig. 4.13 compares Flexile relative to Teavar and the new CVaR-based schemes that we designed (§4.3) for the single traffic class setting. Each curve shows a CDF of the *PercLoss* achieved by a particular scheme across all topologies.

First, we see that Flexile (left-most curve) achieves significantly lower *PercLoss* relative to Teavar (right-most curve). Interestingly, Teavar achieves an *PercLoss* of 100% in many cases. To understand this, consider a failure scenario that disconnects the network. Teavar cannot count such a scenario towards meeting the requirement of any flow since it optimizes the maximum loss across all source-destination pairs in that scenario. If the topology is connected less than β% of the time, Teavar can only achieve a 100% loss at the β%*ile*. In contrast, each individual flow could still be connected in scenarios that occur β% of the time or higher, allowing Flexile to achieve a much lower loss at the β%*ile* (in some extreme cases, Flexile could guarantee 0% loss for all flows). We also evaluate Teavar in more richly connected topologies later.

Second, our enhanced schemes *Cvar-Flow-Ad* and *Cvar-Flow-St* (which both consider flow losses) significantly outperform Teavar, but still see high *PercLoss* relative to Flexile. This is because the schemes use CVaR to approximate the percentile, while Flexile directly optimizes the percentile. *Cvar-Flow-Ad* does better than *Cvar-Flow-St* as expected because of more adaptive routing.

Finally, *Cvar-Flow-St* significantly reduces *PercLoss* relative to Teavar, with a reduction of more than 50% in the median case. This indicates considering flow losses offers significant benefits despite limited routing flexibility, and the CVaR approximation.

Flexile vs. SMORE. SMORE, like Teavar, optimizes the loss across all flows in each scenario. Since the topology may get disconnected, we considered a variant of SMORE where

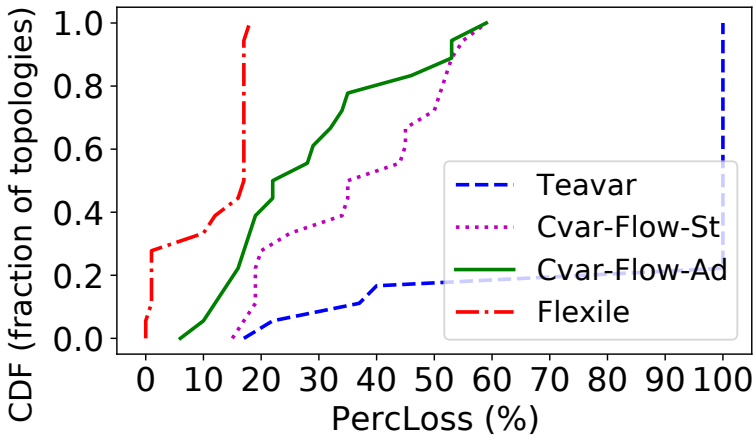


Figure 4.13. Flexile Vs. Teavar and our CVaR variants.

in each scenario we turned off traffic for disconnected flows. This approach performed similar to Flexile in many cases although there were topologies where Flexile still gave benefits. Even so, Flexile can verify that the network cannot perform better, while SMORE is unable to do so.

We also compare Flexile with SMORE in more richly connected settings, which we create by assuming each link consists of two sub-links that fail independently. We ensure the topology remains connected in all sampled failure scenarios. Fig.4.14 compares the *PercLoss* achieved by Flexile, SMORE and Teavar in these more richly connected topologies. Flexile consistently outperforms Teavar and SMORE in most topologies. In the median case, the % reduction in *PercLoss* achieved by Flexile over SMORE is 46% and over Teavar is 63%. In a few cases, we do not report results for Teavar since it did not finish within several hours.

4.4.3 Does Flexile increase loss in scenarios?

As discussed in §4.1 and §4.2.4, although Flexile prioritizes critical flows in each scenario, it has several techniques to curtail the impact on non-critical flows. We next evaluate Flexile's effectiveness in this regard.

Single class traffic: We evaluate Flexile with respect to the *ScenLoss* metric (§4.1.1) (i.e., the loss of the worst performing flow in each scenario), focusing on connected flows. We compare Flexile with ScenBest, the optimal scheme in this metric. For all but the IBM topology, Flexile achieves identical *ScenLoss* as ScenBest. We have already shown in §4.1.2

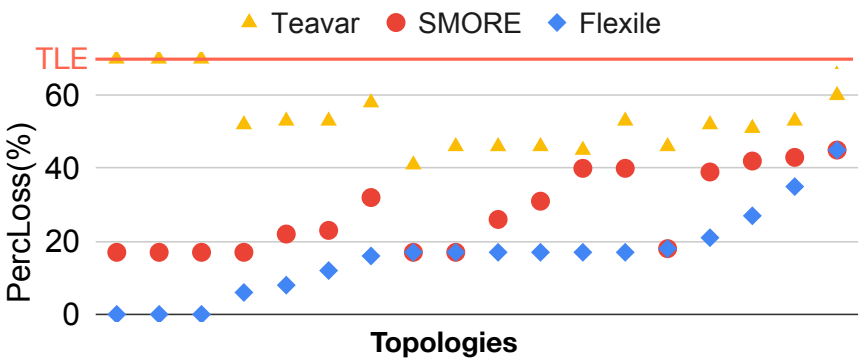


Figure 4.14. Flexile Vs. SMORE and Teavar in richly connected topologies. TLE indicates Time Limit Exceeded.

(Fig. 4.5) that the increase in *ScenLoss* with Flexile for IBM is modest. In contrast, Teavar sees high *ScenLoss* for all topologies. The 99.9%ile *ScenLoss* with Teavar is 100% for all except 4 topologies while this metric never exceeds 17% with Flexile and ScenBest for any topology.

Multiple class traffic: For two class traffic, we generalize the *ScenLoss* metric to consider the loss of the worst performing flow for each class separately. We also generalize the ScenBest scheme to optimize the loss of the worst performing connected flow for each class, with preference to the higher priority class. We refer to this scheme as ScenBest-Multi. Fig. 4.15 shows a distribution of the generalized scenario loss metric for various schemes for the Sprint topology for both traffic classes. We make several observations. First, *for high priority traffic, Flexile incurs no loss for any flow in any scenario.* Note that all three schemes see no loss and overlap on the left. Second, for low priority traffic, while Flexile does result in an increase in the loss for the worst flow, the increase is modest. Note that unlike Flexile, ScenBest-Multi performs poorly for the *PercLoss* metric when loss of flows at a desired percentile is considered. Finally, Flexile performs much better than SWAN-Maxmin. This is owing to the optimizations described in §4.2.3.

Flexile ensures all high priority flows see no loss for all other topologies as well. For low priority flows, Flexile ensures the additional loss in the worst-performing flow is small for most other topologies. In the remaining topologies, Flexile's approach to constraining the increase in loss of non-critical flows works well (§4.2.4). Specifically, we consider a variant Flexile(0.05), which ensures all flows with Flexile sees bandwidth loss at most 5% more

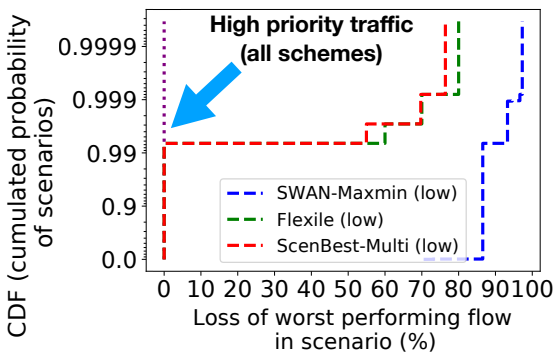


Figure 4.15. Flexile sees modest penalty in *ScenLoss* relative to the optimal.

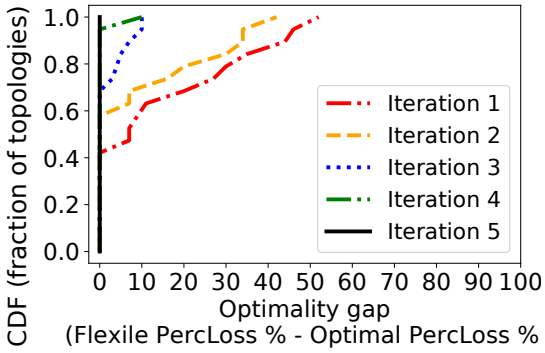


Figure 4.16. Performance improvement with each iteration.

than ScenBest-Multi in all scenarios. The variant provides significant benefits in *PercLoss* even though *ScenLoss* only increases modestly and is strictly controlled. For instance, for the Quest topology, Flexile(0.05) achieves an *PercLoss* of 16%, significantly outperforming both ScenBest-Multi (35% *PercLoss*) and SWAN-Maxmin (57% *PercLoss*). Overall, the results show Flexile can bound the loss in scenarios, yet provide substantial benefits when minimizing flow loss at a desired percentile.

4.4.4 Evaluating other aspects of Flexile

Convergence to optimality. We next compare Flexile to the optimal *PercLoss* that the network can achieve for topologies for which we could compute the optimal. Fig. 4.16 shows the CDF of the optimality gap (*PercLoss* achieved by Flexile - optimal *PercLoss*) across topologies after each iteration of Flexile's decomposition algorithm (§4.2.2) for the two-class traffic setting. Across all topologies, Flexile achieves the optimal in 5 iterations, frequently achieving it in fewer iterations. Interestingly, for 40% of the topologies, Flexile achieves the optimal in the first iteration showing the effectiveness of our starting point heuristic. We found Flexile typically converged to optimal even faster in the single-class experiments.

Sensitivity to scale factor. To see how Flexile responds to different scales of traffic, we scale the low priority traffic by different factors in our two class traffic experiments. Fig. 4.17 shows the maximum factor we can scale without incurring any 99%tile loss using Flexile and

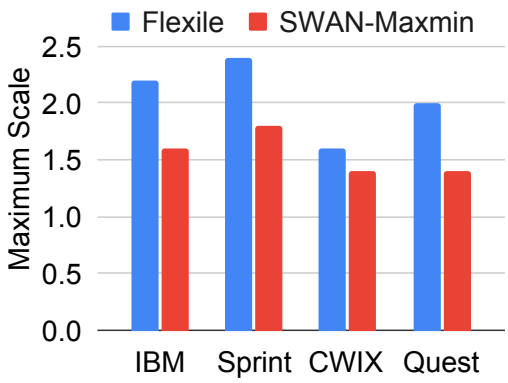


Figure 4.17. Flexile can achieve higher traffic scale.

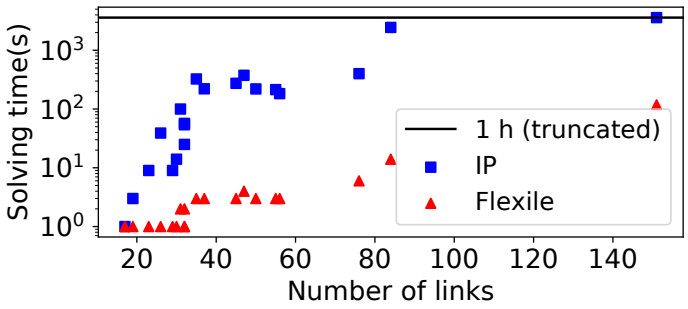


Figure 4.18. Reduction in solving time with Flexile

SWAN-Maxmin on different topologies. We can see that Flexile can support much higher scale factor than SWAN-Maxmin.

Solving time. Fig. 4.18 presents the solving time (Y-Axis) for different topology sizes (X-Axis) for *IP* and Flexile, assuming 5 iterations for Flexile. Note that this is the offline solving time and done prior to failure. Flexile solves multiple small LP subproblems in each iteration, and a master problem (a MIP). For Tinet (one of our larger topologies), each subproblem takes 0.10-0.15 seconds. The master problem is much smaller than the IP (I), and takes less than 0.10 seconds for Tinet. We report the solving time of the master and all subproblems, based on solving up to 10 subproblems in parallel.

Fig. 4.18 shows that Flexile reduces solving time significantly, and is under 15 seconds for all topologies except the largest (Deltacom) which takes 118 seconds. In contrast, *IP* cannot finish within 1 hour for Deltacom and takes more than 40 minutes for Tinet. Note that further optimizations are possible for Flexile – e.g., the *PercLoss* for Deltacom was under 1% after 2 iterations indicating we could have stopped earlier.

Interestingly, we found that the time to solve Teavar is often significantly higher than Flexile – e.g., Teavar is unable to finish Deltacom even after several hours. Although Teavar solves a single LP, its solving time can be large since it bundles all the enumerated scenarios in a single problem.

Finally, the online solving time incurred on failure is comparable to SWAN-Maxmin, and typically under 3 seconds. Further reductions are achievable with coarse buckets for the

max-min scheme, or using an even lighter-weight scheme such as SMORE augmented with critical flows.

4.5 Related work

There has been recent interest in designing TE schemes with probabilistic requirements [26], [27], [54]. Lancet [26] designs protection routing schemes that ensure the network does not experience congestion a desired percentage of time. The primary metric is MLU, and Lancet does not consider flow losses, which is our focus. Beta [54] tackles an orthogonal problem where traffic arrives incrementally and in an online fashion, decides whether to accept the demand. However, the admission decision can be overly conservative, and newly arriving higher priority traffic may be rejected because of existing lower priority traffic.

NetDice [55] verifies that network configurations meet a probabilistic requirement. The focus is on distributed control planes (shortest paths, route redistribution, BGP etc.), and verifying properties such as path length. Earlier work [56] allows modeling probabilistic network behaviors, such as packet delivery probability on failures.

Resilient routing schemes [10], [20], [23], [31], [48] guarantee the network remains congestion-free over scenarios with f or fewer failures, focusing on metrics such as MLU, or a demand scale factor. Instead, Flexile considers failure probability, and flow losses. Researchers have explored verification of distributed control planes to ensure load is not violated on failures [57], robust network design under single link or node failures [11]–[17], and robust design across traffic matrices [15], [43]–[45].

A linear program is decomposed in [58] to distribute the centralized TE controller, while Flexile involves decomposing an Integer Program. Recent work [59] decomposes an MCF problem into a set of subproblems, while Flexile solves a problem over multiple network states. Finally, much research explores how to re-route traffic to restore connectivity on failures [5]–[9] but does not consider meeting flow loss percentiles.

4.6 Conclusions

In this thesis, we have presented Flexile, a new approach for designing cloud provider WANs in a manner that meets the bandwidth requirements of flows over failure scenarios that occur with a desired probability. Unlike existing TE schemes that seek to meet the requirements of all flows over the same set of failure scenarios making them unduly conservative, Flexile exploits a key opportunity that each flow could meet its bandwidth requirements over a different set of failure scenarios. As part of Flexile, we have presented an approach to optimize the β^{th} percentile of bandwidth losses of all flows, a hard problem, and tackled the same using a novel decomposition approach accelerated with problem-specific insights. We have extended a CVaR-based approach to our setting, a well-accepted method to approximating loss percentiles, and shown that it can be conservative. Evaluations over 20 real topologies show the benefits with Flexile are significant. Flexile (i) reduces *PercLoss* by over 80% relative to Teavar and SMORE for over 50% of topologies; and (ii) has a solving time of tens of seconds, acceptable for the offline phase, and scales well with the number of flows. Overall, the results show the promise of Flexile.

5. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

5.1 Conclusion

To cope with uncertainty in network operations, this thesis aims at developing routing schemes that can provide provable worst-case guarantees and schemes that can ensure the network performance be met a percentage of time.

To provide worst-case guarantees, the research community has developed routing schemes to ensure the networks can handle the desired traffic under a set of failures. This thesis discovers that the state-of-the-art scheme can be highly conservative. In order to bridge the gap between existing schemes and the network intrinsic capability, this thesis designs a resilient routing (PCF) to support up to f simultaneous failures. PCF introduces a notion called logical sequence to carefully increase flexibility in network response. This allows us to achieve high throughput, tractable failure analysis and low response overhead at the same time. Formal results show that PCF is provably better than the state-of-the-art scheme, and our experiments show that PCF achieves up to 50% improvement over state-of-the-art on average.

Most of the existing works don't consider percentile performance. This thesis contributes to Lancet, a system for analyzing protection routing schemes that can meet a performance target a desired percentage of time, by showing how to realize a distributed protection routing scheme, and extending the scheme to support multiple traffic classes. Further, this thesis develops a new system (Flexile) for designing a routing scheme to provide provable percentile guarantees. Flexile exploits a key opportunity that different flows can meet their requirements through different scenarios, while all existing schemes use the same set of failure states to meet requirement of all flows. This opportunity enables Flexile to design routing much more flexibly and to guarantee better percentile performance. Flexile consists of an offline phase which decides which set of scenarios should be used to meet percentile target for each flow, and an online phase which designs the actual routing based on the current network state and information achieved from offline phase. In order to speed up model solving in Flexile, we develop a decomposition algorithm use many domain specific accelerations to reduce the solving time. Our experiments show that Flexile reduces traffic loss by more than

36% comparing to state-of-the-art in the median case across 20 topologies within reasonable solving time.

5.2 Future research directions

This thesis tackles the problem of designing resilient routing schemes with provable performance in order to keep up with the fast growing traffic demand and increasing performance requirement. There are still many interesting aspects in resilient routing schemes for further exploration. Here, we introduce two for potential future research directions: varying traffic demands and satisfying latency requirements.

Varying traffic demands. In our designs for both PCF and Flexile, we make the assumption that we are aware of the amount of the traffic we need to carry when we design the routing. In practice, although we can predict the traffic demand based on the history, the prediction may not be perfectly accurate so the assumption can be too strong. Further, there is always a possibility that some unexpected event occurs which brings sudden traffic spikes. It remains a question how to provide performance guarantees when uncertainty in traffic demand is considered. We remark here that our approach has the potential to be adapted to deal with variance in traffic demands. For example, we can introduce traffic variance into network states so that our designed routing is resilient to a set of scenarios with different traffic demands. This approach works if only a limited and discrete number of traffic matrices need to be supported. It is still challenging to consider more sophisticated and complicated traffic models when designing routing because the range of variance can be very large.

Satisfying latency requirements. In this thesis, we focus on optimizing the routing to send more traffic (minimizing the loss) and to guarantee performance in this sense. However, we pay less attention to how to keep a low latency when designing the routing. While latency is a critical criteria in practice, in fact, it remains a challenge to design a routing which can guarantee low latency under a set of failures (or with a certain probability), especially due to the complex model to describe latency in routing. A simple way to deal with this is to embed latency requirements in tunnel selection. But this may result in congestion or unsatisfactory

availability under failures when the selected tunnels are not diverse enough. Thus, how to design routing with low latency while providing desirable performance guarantees is still an interesting research direction.

REFERENCES

[1] J. C. Mogul, R. Isaacs, and B. Welch, "Thinking about availability in large service infrastructures," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, ser. HotOS '17, 2017, pp. 12–17.

[2] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proceedings of ACM SIGCOMM*, 2011, pp. 350–361.

[3] R. Potharaju and N. Jain, "When the network crumbles: An empirical study of cloud network failures and their impact on services," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ser. SOCC '13, 2013, 15:1–15:17.

[4] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: High-availability design principles drawn from googles network infrastructure," in *Proceedings of ACM SIGCOMM*, 2016, pp. 58–72.

[5] P. Pan, G. Swallow, and A. Atlas, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels," RFC 4090, May 2005.

[6] M. Shand and S. Bryant, "IP Fast Reroute Framework," RFC 5714, Jan. 2010.

[7] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *Proceedings of ACM SIGCOMM*, Kyoto, Japan, 2007, pp. 241–252.

[8] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng, "Keep forwarding: Towards k-link failure resilient routing," in *Proceedings of IEEE INFOCOM*, Apr. 2014, pp. 1617–1625.

[9] K.-W. Kwong, L. Gao, R. Guérin, and Z.-L. Zhang, "On the feasibility and efficacy of protection routing in ip networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 5, pp. 1543–1556, Oct. 2011.

[10] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proceedings of ACM SIGCOMM*, 2014, pp. 527–538.

[11] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004, ISBN: 0125571895.

[12] R. S. Bhatia, M. Kodialam, T. V. Lakshman, and S. Sengupta, "Bandwidth guaranteed routing with fast restoration against link and node failures," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1321–1330, Dec. 2008.

[13] F. Hao, M. Kodialam, and T. V. Lakshman, "Optimizing restoration with segment routing," in *Proceedings of IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[14] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," *SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 1, pp. 97–108, 2011.

[15] D. Applegate, L. Breslau, and E. Cohen, "Coping with network failures: Routing strategies for optimal demand oblivious restoration," in *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '04/Performance '04, 2004, pp. 270–281.

[16] J. Zheng, H. Xu, X. Zhu, G. Chen, and Y. Geng, "We've got you covered: Failure recovery with backup tunnels in traffic engineering," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, 2016, pp. 1–10.

[17] B. Fortz and M. Thorup, "Robust optimization of OSPF/IS-IS weights," in *Proceedings of International Network Optimization Conference*, 2003, pp. 225–230.

[18] *Cisco WAN automation engine (WAE)*, http://www.cisco.com/c/en/us/products/routers/wan-automation-engine/index.html, 2016.

[19] A. K. Bangla, A. Ghaffarkhah, B. Preskill, B. Koley, C. Albrecht, E. Danna, J. Jiang, and X. Zhao, "Capacity planning for the google backbone network," in *ISMP 2015 (International Symposium on Mathematical Programming)*, 2015.

[20] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang, "R3: Resilient routing reconfiguration," in *Proceedings of ACM SIGCOMM*, 2010, pp. 291–302.

[21] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford, "Optimizing bulk transfers with software-defined optical wan," in *Proceedings of ACM SIGCOMM*, 2016, pp. 87–100.

[22] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, *Site reliability engineering: How google runs production systems*, https://landing.google.com/sre/book.html, O'Reilly Media, Inc., 2016.

[23] R. Sinha, F. Ergun, K. N. Oikonomou, and K. K. Ramakrishnan, "Network design for tolerating multiple link failures using Fast Re-route (FRR)," in *2014 10th International Conference on the Design of Reliable Communication Networks (DRCN)*, Apr. 2014, pp. 1–8.

[24] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, pp. 1765–1775, Oct. 2011.

[25] Y. Chang, *Ensuring network designs meet performance requirements under failures*, Aug. 2019.

[26] Y. Chang, C. Jiang, A. Chandra, S. Rao, and M. Tawarmalani, "Lancet: Better network resilience by designing for pruned failure sets," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, pp. 1–26, Dec. 2019. DOI: 10.1145/3366697.

[27] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjorner, A. Valadarsky, and M. Schapira, "Teavar: Striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of ACM SIGCOMM*, (to appear), 2019.

[28] J. Tantsura, I. Milojevic, E. Crabbe, S. Previdi, M. Horneffer, A. Bashandy, R. Shakir, C. Filsfils, B. Decraene, S. Ytti, *et al.*, "Segment routing architecture," *IETF Internet draft*, 2013.

[29] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *Proceedings of ACM SIGCOMM*, 2015, pp. 15–28.

[30] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-oblivious traffic engineering: The road not taken," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 157–170.

[31] Y. Chang, S. Rao, and M. Tawarmalani, "Robust validation of network designs under uncertain demands and failures," in *14^{th} USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017, pp. 347–362.

[32] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, 2008.

[33] J. H. Bramble and B. E. Hubbard, "On a finite difference analogue of an elliptic boundary problem which is neither diagonally dominant nor of non-negative type," *Journal of Mathematics and Physics*, vol. 43, no. 1-4, pp. 117–132, 1964.

[34] J. M. Borwein and A. S. Lewis, *Convex analysis and nonlinear optimization*. New York: Springer, 2000.

[35] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, ser. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1994.

[36] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*. Englewood Cliffs, New Jersey: Prentice Hall, 1993.

[37] T. J. Jech, *The axiom of choice*. Courier Corporation, 2008.

[38] G. O. Inc., *Gurobi optimizer reference manual*, http://www.gurobi.com, 2016.

[39] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, 2005, pp. 30–30.

[40] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of ACM SIGCOMM*, 2013, pp. 3–14.

[41] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of ACM SIGCOMM*, 2013, pp. 15–26.

[42] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, 2009, pp. 111–122.

[43] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *Proceedings of ACM SIGCOMM*, 2003, pp. 313–324.

[44] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic engineering in dynamic networks," in *Proceedings of ACM SIGCOMM*, 2006, pp. 99–110.

[45] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Towsley, "Optimal routing with multiple traffic matrices tradeoff between average and worst case performance," in *Network Protocols, 2005. ICNP 2005. 13th IEEE International Conference on*, 2005.

[46] *Abilene traffic matrices*, http://www.cs.utexas.edu/~yzhang/research/AbileneTM/, 2014.

[47] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of ACM SIGCOMM*, 2013, pp. 15–26.

[48] C. Jiang, S. Rao, and M. Tawarmalani, "Pcf: Provably resilient flexible routing," in *Proceedings of ACM SIGCOMM*, 2020, pp. 139–153.

[49] *SMORE source code*, https://github.com/cornell-netlab/yates/.

[50] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, "The benders decomposition algorithm: A literature review," *European Journal of Operational Research*, vol. 259, no. 3, pp. 801–817, 2017, ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2016.12.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221716310244.

[51] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California fault lines: Understanding the causes and impact of network failures," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 315–326.

[52] *Mininet*, http://mininet.org/.

[53] *Cloudlab*, https://cloudlab.us/.

[54] H. Zhang, X. Shi, X. Yin, J. Wang, Z. Wang, Y. Guo, and T. Lan, "Boosting bandwidth availability over inter-dc wan," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 297–312.

[55] S. Steffen, T. Gehr, P. Tsankov, L. Vanbever, and M. Vechev, "Probabilistic verification of network configurations," in *Proceedings of ACM SIGCOMM*, 2020, pp. 750–764.

[56] N. Foster, D. Kozen, K. Mamouras, M. Reitblatt, and A. Silva, "Probabilistic netkat," in *Proceedings of the 25th European Symposium on Programming Languages and Systems - Volume 9632*, 2016, pp. 282–309.

[57] K. Subramanian, A. Abhashkumar, L. D'Antoni, and A. Akella, "Detecting network load violations for distributed control planes," in *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2020, 2020, pp. 974–988.

[58] A. Ghosh, S. Ha, E. Crabbe, and J. Rexford, "Scalable multi-class traffic management in data center backbone networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, pp. 2673–2684, 2013.

[59] F. Abuzaid, S. Kandula, B. Arzani, I. Menache, M. Zaharia, and P. Bailis, "Contracting wide-area network topologies to solve flow problems quickly," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, USENIX Association, 2021, pp. 175–200.

VITA

Chuan Jiang is a PhD candidate in the School of Electrical and Computer Engineering at Purdue University, advised by Professor Sanjay Rao. He works in the area of computer systems and networking with a focus on synthesizing wide-area network designs with provable performance and designing resilient traffic engineering schemes. Prior to his doctoral studies, he received his bachelor's degree from Shanghai Jiaotong University, China.