Flexile: Meeting bandwidth objectives almost always

Chuan Jiang
Purdue University
jiang486@purdue.edu

Zixuan Li
Purdue University
li3566@purdue.edu

Sanjay Rao
Purdue University
sanjay@ecn.purdue.edu

Mohit Tawarmalani
Purdue University
mtawarma@purdue.edu

ABSTRACT

Wide-area cloud provider networks must support the bandwidth requirements of network traffic despite failures. Existing traffic engineering (TE) schemes perform no better than an approach that optimally routes traffic for each failure scenario. We show that this results in sub-optimal routing decisions that hurt performance, and are potentially unfair to some traffic across scenarios. To tackle this, we develop *Flexile*, which exploits and discovers opportunities to improve network performance by prioritizing certain traffic in each failure state so that it can meet its bandwidth requirements. *Flexile* seeks to minimize a desired percentile of loss across all traffic flows, while modeling diverse needs of different traffic classes. To achieve this, *Flexile* consists of (i) an offline phase that identifies which failure states are critical for each flow; and (ii) an online phase, which on failure allocates bandwidth prioritizing critical flows for that failure state, while also judiciously allocating bandwidth to non-critical flows. For tractability, *Flexile*'s offline phase uses a decomposition algorithm aided with problem-specific accelerations. Evaluations using real topologies, and validated with emulation testbed experiments, show that *Flexile* outperforms state-of-the-art TE schemes including SWAN, SMORE, and Teavar in reducing flow loss at desired percentiles by 46% or more in the median case.

CCS CONCEPTS

• **Networks** → **Data path algorithms**;

KEYWORDS

network optimization, network resilience

ACM Reference Format:
Chuan Jiang, Zixuan Li, Sanjay Rao, and Mohit Tawarmalani. 2022. Flexile: Meeting bandwidth objectives almost always. In *The 18th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '22), December 6–9, 2022, Roma, Italy*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3555050.3569119

CoNEXT '22, December 6–9, 2022, Roma, Italy

ACM ISBN 978-1-4503-9508-3/22/12.
https://doi.org/10.1145/3555050.3569119

1 INTRODUCTION

Cloud providers must ensure that their networks are designed so as to ensure business-critical applications continually operate with acceptable performance [4, 8, 21]. Networks must meet their performance objectives while coping with failures, which are the norm given the global scale and rapid evolution of networks [17, 18, 21, 29, 33, 41].

Network requirements must typically be met a desired percentage of time. For instance, a network may be required to ensure "a bandwidth of at least B for latency-sensitive traffic between New York and San Francisco 99.9% of the time". The requirements of flows (a term we use to represent traffic between a pair of sites belonging to a given priority class) must be met taking into account the likelihood that the network may experience different failure states (e.g., a particular set of link failures), and the performance that is feasible under each failure state.

Most state-of-the-art traffic engineering (TE) schemes, unfortunately, do not explicitly provide ways to optimize performance at a desired percentile. We show that Teavar[10], one of the representative and few schemes that do consider percentiles, 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. 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 [25]). 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

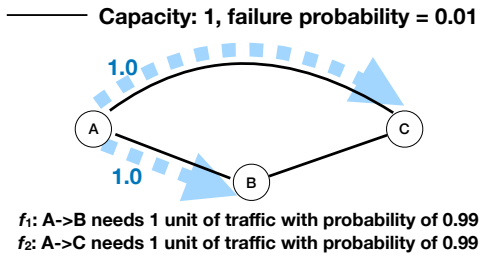


Figure 1: Illustrating *Flexile*'s opportunity. Flow 1 can be fully sent over link A-B 99% of time. Flow 2 can be fully sent over link A-C 99% of time.

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 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 [20, 25]. We have made our code publically available at https://github.com/Purdue-ISL/Flexile/.

2 BACKGROUND

Much early work on wide area TE schemes for failure recovery focuses on quickly re-routing traffic to restore connectivity [26, 27, 31, 36, 45]. However, these schemes are insufficient to prevent congestion under failures – e.g., [28] shows that failures can frequently lead to links getting $10-20\%$ more traffic than their capacity which can negatively impact the performance of demanding applications such as online retail, Web search, and video streaming. We discuss schemes that have emerged in recent years to ensure good performance under failures:

Congestion-free local mechanisms. A first class of schemes [23, 28, 37, 43] *proactively* guarantee the network remains congestion-free over a specified set of failure scenarios (e.g., all scenarios with f simultaneous link failures. See §4.1 for a formal definition of failure scenarios.), while only allowing the network to respond to failures using local rerouting mechanisms. For concreteness, consider FFC [28], a representative approach, which is set in the context of tunnel-based forwarding where traffic for each flow is carried over a set of pre-selected tunnels. On failure, traffic is proportionally rescaled on live tunnels associated with each flow. FFC conservatively admits traffic in a manner that guarantees the network does not experience congestion when local proportional routing is used in the failure scenarios that it plans for. Specifically, given a set of flows each associated with a traffic demand, and a parameter f, FFC solves a robust optimization problem *offline* (i.e., prior to any failure) which determines (i) how much bandwidth to allocate to each flow; and (ii) how the bandwidth must be split across the flow's tunnels so that the allocated bandwidth can be supported in any scenario where f or fewer links fail simultaneously.

Local mechanisms meeting performance percentiles. Most congestion-free mechanisms [23, 37, 43] including FFC design for f or fewer simultaneous failures. Recently, researchers [10, 11] have argued that the approach can be conservative given that service level objectives typically need bandwidth requirements to be met a desired percentage of time, and since the failure probabilities of links is highly heterogeneous. We focus on Teavar [10], a representative scheme that extends FFC to take failure probabilities into account while using the same proportional routing scheme. Given an enumerated set of failure scenarios, and the probability of each scenario, Teavar seeks to allocate bandwidth to flows so that there are sufficient scenarios (that together occur with probability of at least $\beta\%$) where *every flow* sees acceptable loss. The loss of each flow is the fraction of its unfulfilled demand, and we use ***ScenLoss*** to denote the loss of the worst flow in each scenario.

Definition 2.1. Let q be a scenario, F be the set of all flows and $loss_{fq}$ be the loss of flow f in scenario q, we define the loss of worst flow in scenario q as

$$ScenLoss_q = \max_{f \in F} loss_{fq} \tag{1}$$

Ideally, Teavar must find bandwidth allocations that minimize the β^{th} percentile of *ScenLoss*, but the problem is challenging, and instead Teavar uses a conservative approximation (see §5).

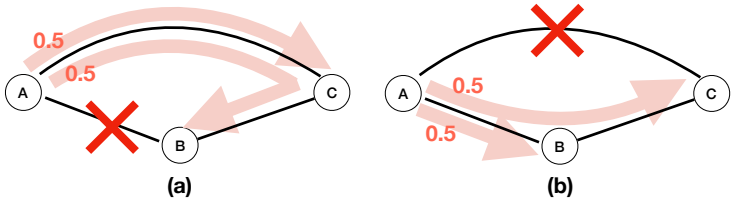
Flexible rerouting. Rather than use a local recovery scheme, another approach is to optimally reroute traffic when a failure occurs. While reaction time is a potential concern, a recent proposal (SMORE [25]) argues for a light-weight recovery mechanism that involves splitting traffic optimally among live tunnels (with the tunnels themselves not changing since adding new tunnels is a heavier weight operation)[1]. Depending on the target metric m, we term a scheme $ScenBest(m)$ if it always reroutes traffic to optimize the metric m when a failure occurs. Note that when m is MLU [2], $ScenBest(MLU)$ performs identically to SMORE. We henceforth refer to $ScenBest(MLU)$ as ScenBest for simplicity. Clearly, for any failure scenario, Teavar can perform no better than ScenBest given its local mechanism. Hence, Teavar cannot achieve a percentile performance any better than ScenBest.

3 *FLEXILE* MOTIVATION

While ScenBest provides the best percentile performance among existing schemes, it optimizes performance unilaterally for each failure state. In this section, we illustrate with an example that doing so may lead to sub-optimal decisions across states. We then

[1]While SMORE [25] does not extensively discuss SMORE's failure recovery mechanisms, we clarified this from the authors and the source code [3]

[2]Minimizing *ScenLoss* is equivalent to minimizing the maximum link utilization (MLU) of all links, the metric used in SMORE [25]. It is also equivalent to maximizing the demand scale factor using a maximum concurrent flow formulation [28]. See Appendix



Scenario(s)	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 2: Bandwidth objectives cannot be met for topology in Fig.1 by existing TE schemes. Each column in 2(c) represents one or more scenarios – e.g., the 2nd column represents all scenarios where AB and AC are both alive.

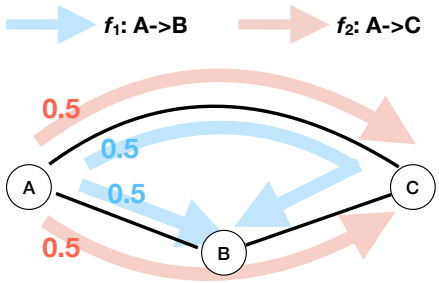


Figure 3: In Teavar's design, f_1 and f_2 are both split equally among 2 paths.

Scenario(s)	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: Critical scenarios for Fig. 1.

show *Flexile* has the potential to do better by optimizing different flows in different states.

Consider Fig.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 follows from the simple routing strategy which sends f_1 on link A–B and f_2 on link A–C. Clearly, the requirements of f_1 (resp. f_2) are met whenever link A–B (resp. A–C) is alive, which occurs 99% of the time.

State-of-the-art TE schemes cannot meet the requirements of flows. Unfortunately, ScenBest can only support 0.5 units for f_1 and f_2 99% of the time. To illustrate this, consider the scenarios in Fig 2(a) and Fig 2(b) where link A–B and link B–C fail respectively. Since ScenBest optimizes *ScenLoss* (i.e., the loss of the worst flow) in each scenario, it sends 0.5 units of each flow as shown in Fig 2(a) and Fig 2(b) so that neither flow will have loss worse than 50%. Fig.2(c) summarizes the throughput achieved by each flow under different failure scenarios. Since each of the scenarios shown in Fig 2(a) and Fig 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.

Teavar [10] too cannot support more than 0.5 units 99% of time for both flows. Fig. 3 shows Teavar's designed routing for the topology in Fig. 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. 2. Thus, like SMORE, Teavar cannot support more than 50% traffic for both flows 99% of time.

***Flexile*'s approach.** *Flexile* determines the **critical scenarios** associated with each flow where its loss must be acceptable so the flow objectives can be met. Unlike ScenBest which seeks to ensure all flows in a scenario see as low a loss as possible, *Flexile* prioritizes critical flows in any given scenario. Fig. 4 illustrates this for the topology in Fig. 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. *Flexile* can support 1 unit of each of f_1 and f_2 by prioritizing them in their critical scenarios.

Another way to interpret the above example is that ***Flexile* requires less capacity to be provisioned** to meet desired bandwidth objectives than existing TE schemes. In Fig.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.

A potential concern with *Flexile* is that 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, 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.5 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 §6 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 §2. 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.6 which shows a CDF of the loss penalty paid by *Flexile* relative to ScenBest (i.e., the increase in loss for the worst flow in each scenario relative to optimal for that scenario). For comparison, we also show loss penalties incurred by Teavar. 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%).

Other benefits. In the Appendix, we discuss additional benefits of *Flexile*. Specifically, (i) *Flexile* ensures bandwidth guarantees never degrade with additional links, while we present an example to show this is not the case for ScenBest; and (ii) existing approaches that are fair in each scenario may in fact be unfair when performance across scenarios is considered - in contrast, *Flexile* can mitigate this effect.

4 *FLEXILE* DESIGN

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. (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 has only recently received attention from the networking research community. While it is possible to approximate percentiles as done by Teavar [10], we show in §5 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 [20] 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*.

4.1 Optimizing flow loss percentiles

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 $pr(f)$.

Failure scenarios and probabilities. Q represents the set of considered failure scenarios with p_q denoting the probability of $q \in Q$. In our discussions so far, each failure scenario q represents a network state where a particular subset of links fail while others are alive. More generally, network failures are modeled using Shared Risk Link Groups (**SRLGs**), where each SRLG captures a group of links that fail together (e.g., owing to the failure of a shared optical component [40]). Each failure scenario q represents a network state where a particular subset of SRLGs fail while others are alive. Note that failure scenarios are disjoint with each other. *Flexile*'s models do not assume independent SRLG failures, and can model correlated SRLG failures if provided the joint probability of multiple SRLG failures occurring together. However, for practicality of estimating probabilities, SRLG failures are typically independent [10, 17, 29]. Failure probabilities may be obtained from historical failure data – e.g., given data from Microsoft's WAN on the status of links in every time epoch over a long period, Teavar [10] estimates the mean time between failure which it converts into failure probability. Further [8, 44] indicate that Facebook and Google have failure probability data available and are already using the same in simulation testing [4]. A similar approach has been used by past works [10, 11, 38].

Modeling desired percentile of flow loss. First, we define $FlowLoss(f, \beta_k)$.

Definition 4.1. In each traffic class k, for each flow $f \in F_k$, $FlowLoss(f, \beta_k)$ is 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 $FlowLoss(f, \beta_k)$.

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 $f \in F_k$.

Definition 4.2. We define the following metric that we refer to as ***PercLoss*** (and may abbreviate as α_k).

$$\alpha_k := PercLoss_k = \max_{f \in F_k} FlowLoss(f, \beta) \tag{2}$$

Fig. 7 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

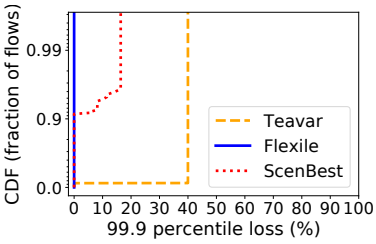


Figure 5: CDF of 99.9%ile loss across flows for IBM topology.

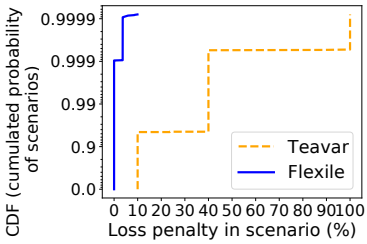


Figure 6: Increase in *ScenLoss* relative to ScenBest (optimal).

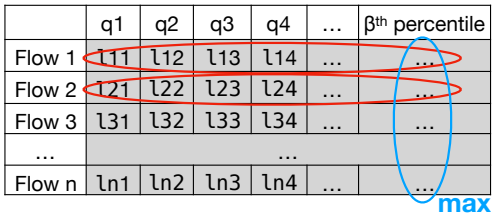


Figure 7: Meeting bandwidth requirements requires computing the β^{th} percentile of flow losses.

Notation	Meaning
Q	Set of considered 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

Table 1: Notation.

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 (§4.4).

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 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., $l_{fq} \le PercLoss_k$.

$$\sum_{q\in Q} z_{fq}p_q \ge \beta_k \quad \forall k \in K, f \in F_k \tag{3}$$

$$\alpha_k \ge l_{fq} - 1 + z_{fq} \quad \forall k \in K, f \in F_k, q \in Q \tag{4}$$

Here, (3) ensures that for each flow in k, we select enough critical scenarios to cover the probability β_k. When $z_{fq} = 1$, (4) becomes $PercLoss_k \ge l_{fq}$ meaning we care about the loss l_{fq}. When $z_{fq} = 0$, (4) 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 $R_k(i)$ 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 1 summarizes notation.

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

s.t. (3), (4)

$$\sum_{t\in R_k(i)} x_{ktq}y_{tq} \ge \sum_{pr(f)=i, f\in F_k} (1 - l_{fq})d_f \quad \forall k \in K, i \in P, q \in Q \tag{5}$$

$$\sum_{k\in K, e\in t} x_{ktq} \le c_e \quad \forall e \in E, q \in Q \tag{6}$$

$$x_{ktq} \ge 0 \quad \forall k \in K, i \in P, t \in R_k(i), q \in Q \tag{7}$$

$$z_{fq} \in \{0, 1\} \quad \forall k \in K, f \in F_k, q \in Q \tag{8}$$

$$0 \le l_{fq} \le 1 \quad \forall k \in K, f \in F_k, q \in Q. \tag{9}$$

(5) ensures that there is enough bandwidth allocated to each pair. The LHS of (5) 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 [10]. (6) and (7) 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. The number of constraints in the above formulation depends on the total number of scenarios in Q. While the total number of possible scenarios the network may encounter is large, we note that Q only need to contain sufficient scenarios that occur with probability higher than β. Nevertheless, the formulation is large, which we tackle next.

4.2 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 [34], to our domain with many problem specific

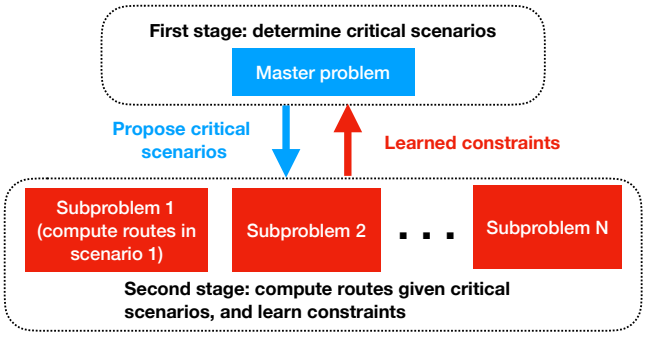


Figure 8: Systematic decomposition approach.

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 discuss why in the Appendix).

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. 8 illustrates our procedure. Formally, for each scenario q, we have the following smaller subproblem (note that z_{fq} is a parameter here):

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

$$\text{s.t. } \alpha_k \geq l_{fq} - 1 + z_{fq} \quad \forall k \in K, f \in F_k \tag{10}$$

$$0 \leq l_{fq} \leq 1 \quad \forall k \in K, f \in F_k \tag{11}$$

$$\sum_{t\in R_k(i)} x_{ktq} y_{tq} \geq \sum_{f\in F_k, pr(f)=i} (1 - l_{fq}) d_f \quad \forall k \in K, i \in P \tag{12}$$

$$\sum_{k\in K, e\in t} x_{ktq} \leq c_e \quad \forall e \in E \tag{13}$$

$$x_{ktq} \geq 0 \quad \forall k \in K, i \in P, t \in R_k(i). \tag{14}$$

Formally, we rewrite (I) as

$$(I') \min_{z} \quad Penalty(z) \text{ s.t. } (3), (8) \tag{15}$$

$$Penalty(z) = \min_{x,l,\alpha} \sum_{k\in K} \alpha_k w_k \text{ s.t. } (4), (5), (6), (7), (9) \tag{16}$$

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 memorize the intermediate results from solving one scenario to speed up the solution of the next scenario. Specifically, we rewrite (12) and (13) as:

$$\sum_{t\in R_k(i)} x_{ktq} \geq \sum_{f\in F_k, pr(f)=i} (1 - l_{fq}) d_f \quad \forall k \in K, i \in P \tag{17}$$

$$\sum_{k\in K, e\in t} x_{ktq} \leq c_e m_{eq} \quad \forall e \in E. \tag{18}$$

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.

$$(M) \min_{z,Penalty} \quad Penalty$$

$$\text{s.t. } (3), (8)$$

$$Penalty \geq g_q(z_{\cdot q}) \quad \forall g \in G, \forall q \in Q. \tag{19}$$

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. We present exact cut constraints in the appendix.

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 a constraint in (M) to limit the hamming distance between current z variable and z variable achieved from last iteration. We present more details in Appendix.

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, the reformulation of (S_q) discussed earlier allows further optimization. We can generate cuts for many scenarios even though only a few subproblems for a subset of scenarios are solved. See appendix for details.

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 (we defer a proof to the appendix).

Proposition 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.*

Algorithm 1 summarizes our decomposition algorithm (Line 17-19 can be executed in parallel). In the algorithm, z denotes the selection of critical scenarios for every flow (i.e., $z = \{z_{fq} | \forall f, q\}$) and x_q denotes the set of all routing variables in scenario q (i.e., $x_q = \{x_{ktq} | \forall k, t\}$). 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.

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:  function solve_subproblem(z,q)
6:      Solve (S_q) and construct cut constraint g_q
7:      return x_q, g_q
8:  function MAIN(max_iterations)
9:      Initialize z_fq to be 1 if f is connected in q and 0 otherwise
10:     Initialize x_q to be ∅ for all q ∈ Q
11:     cur_iteration ← 0
12:     G ← ∅
13:     while cur_iteration < max_iterations do
14:         g ← ∅
15:         for q ∈ Q do
16:             if q cannot be pruned then
17:                 x_q, g_q ← solve_subproblem(z, q)
18:                 g.add(g_q)
19:         G.add(g)
20:         z ← solve_master(G, z)
21:         cur_iteration ← cur_iteration + 1
22:     return x
```

4.3 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. Over long time scales (every 5-10 minutes), the offline problem is solved using a prediction of traffic matrix, and an estimation of failure probabilities. When a failure occurs, the controller solves an online LP to generate the routing weights to be installed in the routers (similar to SMORE and SWAN). This online LP makes sure that necessary bandwidth is first allocated to critical flows. However, there is typically significant residual capacity remaining, which this LP 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 [20], 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 [20], 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) 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.

Flexile's approach to identify traffic of different priority classes, and to ensure each flow of a class gets its share is the same as existing systems such as SWAN. For instance, SWAN uses DSCP bits to identify traffic of different priority classes; and uses a token bucket in each end host that regulates traffic to the destination. The same approach is applicable to Flexile - the only difference is in terms of how much traffic is allocated to each flow when a failure happens.

With *Flexile*, routers only maintain forwarding entries associated with the current failure scenario. Hence the storage required at each router is identical to existing TE schemes. The only additional storage required is for storing the information of critical flows in the centralized controller. This requires a single bit for each combination of scenario and flow and requires $O(MN)$ space where M is the number of scenarios and N is the total number of flows. For a topology of 100 nodes, and 1000 failure scenarios, this storage requirement is only 1.25MB.

4.4 Generalizations

Constraining loss on non-critical flows. While §4.3 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. It is 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 (5) 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 13). 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 §4.3 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 percentiles. *Flexile* can be generalized to perform minimum-cost capacity augmentation on the network which is more cost-effective than a scenario-centric approach (see Appendix).

Handling imperfect probability prediction. *Flexile* can tolerate errors in the probabilities of individual scenarios so long as the cumulative probability of the scenarios it designs matches the SLO target. When the predicted probabilities are imperfect, applying *Flexile* directly may design for 99% but the true probability of the scenarios it designs for is lower, implying the actual guarantee is weaker. To compensate, we can design for a slightly higher target probability, sufficient to tolerate prediction error. Note that we do not need to know the probability of all failure scenarios. It suffices to know the probability of a subset of scenarios that occur with sufficiently high probability.

5 *FLEXILE* VS. TEAVAR

While *Flexile* minimizes the β^{th} percentile of losses (or Value at Risk or VaR), Teavar [10] 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 [25] 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., *FlowLoss*(f, β), 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,

$$MaxFlowCVaR = \max_{f \in F} CVaR(f,\beta) \tag{20}$$

. • *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, which we present in the appendix. We have the following proposition.

Proposition 2. *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 a PercLoss of zero.*

The proof follows from an analysis of Fig. 1, and we defer details to the appendix. The proposition shows that these more general strategies are still quite conservative, and there is significant potential to doing better with *Flexile* by directly considering VaR. We empirically show the benefits in §6.2.

6 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 (§5). 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.

• ***SMORE:*** SMORE split traffic optimally among live tunnels upon failures. This is identical to ScenBest discussed in section §2 when the optimized metric is MLU.

• ***SWAN:*** We consider both variants of SWAN [20], 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) 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 [22] 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*.

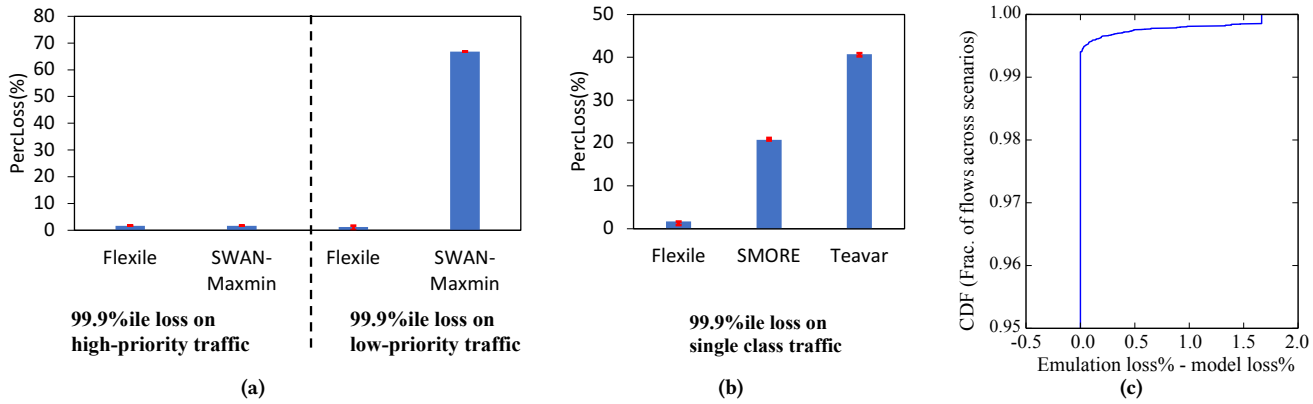


Figure 9: (a) Emulation testbed results. (a) *Flexile* vs. SWAN. (b) *Flexile* vs. Teavar and SMORE. (c) Comparing flow losses across scenarios in emulations with model predicted losses.

Topologies and traffic model. We evaluate the schemes on 20 topologies obtained from [24] and [25] (see Table 2 in the Appendix). 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. For each topology, the number of flows is $K\frac{N*(N-1)}{2}$, where K is the number of traffic classes and N is the number of nodes. We choose tunnels balancing latency and disjointness like prior works [10, 23, 28]. 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 [48] 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 [10]. We choose the Weibull parameter so that the median failure probability is approximately 0.001, matching empirical data characterizing failures in wide-area networks [17, 29, 41]. 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 generalizes to shared risk link groups with correlated failures (§4.1). We discard scenarios with insignificant probability ($< 10^{-6}$). For single-class experiment, our design target is set to as high a probability target as possible, while ensuring all flows remain connected for the sampled scenarios. This is because the network will trivially see a *PercLoss* of 1 for any 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 emulations are conducted on a Mininet cluster [2] running on six Cloudlab servers [1]. 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.

6.1 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. 9a 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. 9b 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

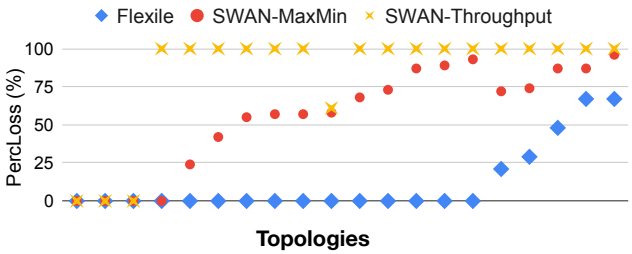


Figure 10: *Flexile* Vs. SWAN. *Flexile* matched optimal whenever it was computable. Vertically aligned dots correspond to the same topology.

indicating it can support all flows with minimal loss 99.9% of the time, 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. 9c 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.

6.2 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. 10 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. Finally, in the Appendix we show that with *Flexile*, lower priority traffic can be scaled by a larger factor while ensuring zero *PercLoss*.

***Flexile* vs. Teavar and our CVaR variants.** Fig. 11 compares *Flexile* relative to Teavar and the new CVaR-based schemes that we designed (§5) for the single traffic class setting. Each curve shows a CDF of the *PercLoss* achieved by a particular scheme across all topologies.

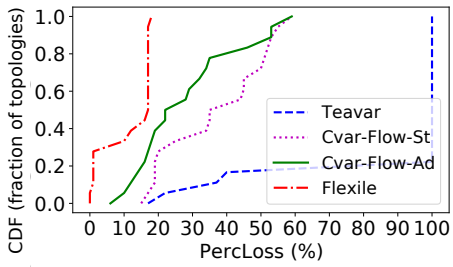


Figure 11: *Flexile* Vs. Teavar and our CVaR variants.

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 $\beta\%$ of the time, Teavar can only achieve a 100% loss at the $\beta\%ile$. In contrast, each individual flow could still be connected in scenarios that occur $\beta\%$ of the time or higher, allowing *Flexile* to achieve a much lower loss at the $\beta\%ile$ (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 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.12 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.

6.3 Does *Flexile* increase loss in scenarios?

While *Flexile* optimizes performance across failure scenarios, this could potentially be at the expense of performance of non-critical

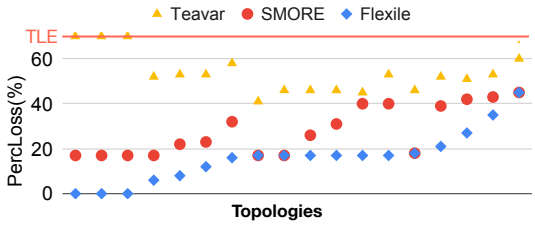


Figure 12: *Flexile* Vs. SMORE and Teavar in richly connected topologies. TLE indicates Time Limit Exceeded.

flows within a scenario. We evaluate how well *Flexile*'s techniques (§3 and §4.4) can mitigate the impact.

Single class traffic: We evaluate *Flexile* with respect to *ScenLoss* (§2) focusing on the loss of the worst performing connected flow in each scenario. For all but the IBM topology, *Flexile* achieves identical *ScenLoss* as ScenBest (which is optimal for each scenario). For IBM, *Flexile* achieves only modestly higher loss than ScenBest as already shown in §3 (Fig. 6). In contrast, Teavar performs poorly – the 99.9%ile *ScenLoss* with Teavar is 100% for all except 4 topologies while *Flexile* and ScenBest achieve *ScenLoss* under 17% for all topologies.

Multiple class traffic: Fig. 13 shows a distribution of the loss of the worst performing flow in each traffic class for the Sprint topology. ScenBest-Multi generalizes ScenBest for two classes and represents the optimal scheme when performance within each scenario is considered. *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. For low priority traffic, the loss for the worst flow is only modestly higher with *Flexile* compared to ScenBest-Multi, and much better than SWAN-Maxmin. Note that ScenBest-Multi performs poorly in *PercLoss* which looks across scenarios.

Flexile achieves similarly strong results for most other topologies. In a few cases, *Flexile* did see slightly higher loss penalties relative to optimal in some scenarios, but this was limited to low priority traffic. Further, here, a variant of *Flexile* which added a constraint limiting the increase in loss of non-critical flows (§4.4) worked well. For instance, for the Quest topology, the variant only increased the loss of the worst low priority flow by at most 5% in each scenario, yet significantly outperformed in *PercLoss* (the variant achieved an *PercLoss* of 16%, compared to 35% for ScenBest-Multi and 57% for SWAN-Maxmin). Overall, the results show *Flexile* can bound the loss in scenarios, yet substantially improve flow loss at a desired percentile.

6.4 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. 14 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) 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.

Solving time. Fig. 15 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. 15 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. Further optimizations are possible for *Flexile* – e.g., the *PercLoss* for Deltacom was under 1% after 2 iterations indicating we could have stopped earlier.

Note that the solving time depends on (i) the number of iterations; and (ii) the time spent on subproblems in each iteration. Empirically, the number of iterations does not grow with size, and the majority of the time is spent on the subproblems. The number of these subproblems is the number of scenarios we consider. For each subproblem, the number of variables grows linearly with the number of flows, and the number of constraints grows linearly with the number of edges in the network. We solve each subproblem using the simplex algorithm. Smoothed analysis shows that the complexity is polynomial in the size of the LP and the perturbation to its data [13] while empirical performance has been observed to be linear in the number of constraints and variables [35].

Interestingly, we found that Teavar has significantly higher solving times 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 phase only solves one subproblem. Hence solving time grows with the time to solve one subproblem. The scaling is similar to existing TE schemes such as SMORE and SWAN. We found that 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.

7 RELATED WORK

In this section, we only discuss related work not discussed in §2. There has been recent interest in designing TE schemes with probabilistic requirements [10, 11, 47]. Lancet [11] focuses on local rerouting using a link bypass rather than *Flexile*'s flexible rerouting approach, and does not consider flow losses. Beta [47] tackles an orthogonal problem where traffic arrives incrementally and decides whether to accept the demand in an online fashion. However, the admission decision can be overly conservative, and newly arriving higher priority traffic may be rejected because of existing lower priority traffic. Given router configurations and failure probabilities, NetDice [38] verifies path lengths are under a threshold with desired

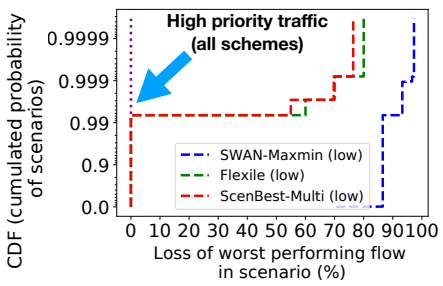


Figure 13: *Flexile* sees modest penalty in *ScenLoss* relative to the optimal.

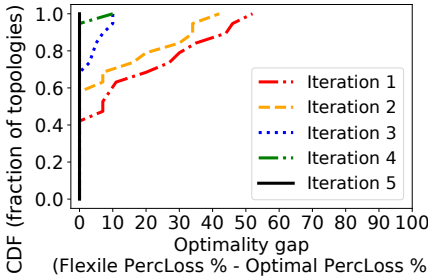


Figure 14: Performance improvement with each iteration.

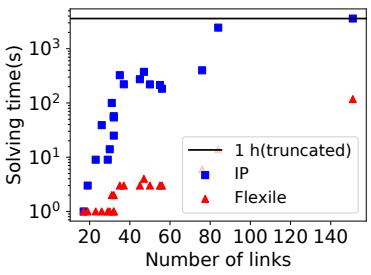


Figure 15: Reduction in solving time with *Flexile*

probability. Other work [15] models packet delivery probability on failures.

Recent work [5] shows how an MCF problem could be solved quickly on failure through decomposition which enables flexible routing with fast reaction time. This may be viewed as a variant of a ScenBest scheme that trades off optimality for computation speed. In contrast *Flexile* optimizes across scenarios to meet flow percentile requirements. Researchers have explored robust network design under single link or node failures [6, 9, 14, 19, 32, 40, 49], verified link utilizations under failure [12, 39], and explored robust design across traffic matrices [6, 7, 42, 46]. While we focus on IP topology failures, ARROW [50] allows restoration of link capacity by readjusting the underlying fiber path on fiber failure. Decomposition techniques have been explored in other contexts such as distributed SDN controllers [16].

8 CONCLUSIONS

In this paper, we have presented *Flexile*, a new system to minimize flow loss at a desired percentile in a cloud provider WAN, while modeling the diverse needs of different traffic classes. *Flexile* exploits a key opportunity that each flow could meet its bandwidth requirements over a different set of failure states. Evaluations over 20 real topologies validated with emulation testbed experiments show *Flexile* out-performs existing TE schemes. Across topologies, the median reduction in flow loss at desired percentiles with *Flexile* is 46% for SMORE, and 63% for Teavar, while the benefits are even higher for SWAN. Finally, its decomposition approach aided with problem-specific optimizations ensures solving times of under 15 seconds for most topologies and *Flexile* is an order of magnitude faster than Teavar for the largest topology. Overall, the results show the promise of *Flexile*. **This work does not raise any ethical issues.**

9 ACKNOWLEDGEMENTS

We thank our shepherd, Gábor Rétvári, and the anonymous reviewers for their feedback which greatly helped improve the paper. This work was funded in part by the National Science Foundation (NSF) Award 1910234.

REFERENCES

[1] Cloudlab. https://cloudlab.us/.
[2] Mininet. http://mininet.org/.
[3] SMORE source code. https://github.com/cornell-netlab/yates/.
[4] Cisco WAN automation engine (WAE), 2016. http://www.cisco.com/c/en/us/products/routers/wan-automation-engine/index.html.
[5] Firas Abuzaid, Srikanth Kandula, Behnaz Arzani, Ishai Menache, Matei Zaharia, and Peter Bailis. Contracting wide-area network topologies to solve flow problems quickly. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 175–200. USENIX Association, 2021.
[6] David Applegate, Lee Breslau, and Edith 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*, SIGMETRICS '04/Performance '04, pages 270–281, 2004.
[7] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of ACM SIGCOMM*, pages 313–324, 2003.
[8] Ajay Kumar Bangla, Alireza Ghaffarkhah, Ben Preskill, Bikash Koley, Christoph Albrecht, Emilie Danna, Joe Jiang, and Xiaoxue Zhao. Capacity planning for the google backbone network. In *ISMP 2015 (International Symposium on Mathematical Programming)*, https://research.google/pubs/pub45385/, 2015.
[9] Randeep S. Bhatia, Murali Kodialam, T. V. Lakshman, and Sudipta Sengupta. Bandwidth guaranteed routing with fast restoration against link and node failures. *IEEE/ACM Transactions on Networking*, 16(6):1321–1330, December 2008.
[10] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjorner, Asaf Valadarsky, and Michael Schapira. Teavar: Striking the right utilization-availability balance in wan traffic engineering. In *Proceedings of ACM SIGCOMM*, 2019.
[11] Yiyang Chang, Chuan Jiang, Ashish Chandra, Sanjay Rao, and Mohit Tawarmalani. Lancet: Better network resilience by designing for pruned failure sets. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3:1–26, 12 2019.
[12] Yiyang Chang, Sanjay Rao, and Mohit Tawarmalani. Robust validation of network designs under uncertain demands and failures. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 347–362, 2017.
[13] Daniel Dadush and Sophie Huiberts. *Smoothed Analysis of the Simplex Method*, page 309–333. Cambridge University Press, 2021.
[14] Bernard Fortz and Mikkel Thorup. Robust optimization of OSPF/IS-IS weights. In *Proceedings of International Network Optimization Conference*, pages 225–230, 2003.
[15] Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. Probabilistic netkat. In *Proceedings of the 25th European Symposium on Programming Languages and Systems - Volume 9632*, pages 282–309, 2016.
[16] A. Ghosh, Sangtae Ha, E. Crabbe, and J. Rexford. Scalable multi-class traffic management in data center backbone networks. *IEEE Journal on Selected Areas in Communications*, 31:2673–2684, 2013.
[17] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proceedings of ACM SIGCOMM*, pages 350–361, 2011.
[18] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of ACM SIGCOMM*, pages 58–72, 2016.
[19] Fang Hao, Murali Kodialam, and T. V. Lakshman. Optimizing restoration with segment routing. In *Proceedings of IEEE INFOCOM*, pages 1–9, April 2016.
[20] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *Proceedings of ACM SIGCOMM*, pages 15–26, 2013.
[21] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 74–87, 2018.
[22] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2016. http://www.gurobi.com.

[23] Chuan Jiang, Sanjay Rao, and Mohit Tawarmalani. Pcf: Provably resilient flexible routing. In *Proceedings of ACM SIGCOMM*, page 139–153, 2020.
[24] Simon Knight, Hung Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29:1765 – 1775, October 2011.
[25] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. Semi-oblivious traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170, 2018.
[26] Kin-Wah Kwong, Lixin Gao, Roch Guérin, and Zhi-Li Zhang. On the feasibility and efficacy of protection routing in ip networks. *IEEE/ACM Transactions on Networking*, 19(5):1543–1556, October 2011.
[27] Karthik Lakshminarayanan, Matthew Caesar, Murali Rangan, Tom Anderson, Scott Shenker, and Ion Stoica. Achieving convergence-free routing using failure-carrying packets. In *Proceedings of ACM SIGCOMM*, pages 241–252, 2007.
[28] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. Traffic engineering with forward fault correction. In *Proceedings of ACM SIGCOMM*, pages 527–538, 2014.
[29] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. Characterization of failures in an operational ip backbone network. *IEEE/ACM Trans. Netw.*, 16(4):749–762, 2008.
[30] Manfred Padberg. *Linear optimization and extensions*, volume 12. Springer Science & Business Media, 2013.
[31] P. Pan, G. Swallow, and A. Atlas. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090, May 2005.
[32] Michal Pióro and Deepankar Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
[33] Rahul Potharaju and Navendu 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*, SOCC '13, pages 15:1–15:17, 2013.
[34] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801 – 817, 2017.
[35] Ron Shamir. The efficiency of the simplex method: A survey. *Management Science*, 33(3):301–334, 1987.
[36] M. Shand and S. Bryant. IP Fast Reroute Framework. RFC 5714, January 2010.
[37] R. K. 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)*, pages 1–8, April 2014.
[38] Samuel Steffen, Timon Gehr, Petar Tsankov, Laurent Vanbever, and Martin Vechev. Probabilistic verification of network configurations. In *Proceedings of ACM SIGCOMM*, page 750–764, 2020.
[39] Kausik Subramanian, Anubhavnidhi Abhashkumar, Loris D'Antoni, and Aditya Akella. Detecting network load violations for distributed control planes. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2020, page 974–988, 2020.
[40] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. Network architecture for joint failure recovery and traffic engineering. *SIGMETRICS Perform. Eval. Rev.*, 39(1):97–108, 2011.
[41] Daniel Turner, Kirill Levchenko, Alex C. Snoeren, and Stefan Savage. California fault lines: Understanding the causes and impact of network failures. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 315–326, 2010.
[42] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: Traffic engineering in dynamic networks. In *Proceedings of ACM SIGCOMM*, pages 99–110, 2006.
[43] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. R3: Resilient routing reconfiguration. In *Proceedings of ACM SIGCOMM*, pages 291–302, 2010.
[44] Yiting Xia, Ying Zhang, Zhizhen Zhong, Guanqing Yan, Chiun Lin Lim, Satyajeet Singh Ahuja, Soshant Bali, Alexander Nikolaidis, Kimia Ghobadi, and Manya Ghobadi. A social network under social distancing: Risk-Driven backbone management during COVID-19 and beyond. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 217–231. USENIX Association, April 2021.
[45] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng. Keep forwarding: Towards k-link failure resilient routing. In *Proceedings of IEEE INFOCOM*, pages 1617–1625, April 2014.
[46] C. Zhang, Zihui 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.
[47] Han Zhang, Xingang Shi, Xia Yin, Jilong Wang, Zhiliang Wang, Yingya Guo, and Tian Lan. Boosting bandwidth availability over inter-dc wan. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, page 297–312, New York, NY, USA, 2021. Association for Computing Machinery.
[48] Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pages 30–30, 2005.
[49] Jiaqi Zheng, Hong Xu, Xiaojun Zhu, Guihai Chen, and Yanhui Geng. We've got you covered: Failure recovery with backup tunnels in traffic engineering. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–10, 2016.
[50] Zhizhen Zhong, Manya Ghobadi, Alaa Khaddaj, Jonathan Leach, Yiting Xia, and Ying Zhang. Arrow: Restoration-aware traffic engineering. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 560–579, New York, NY, USA, 2021.

APPENDIX

A. Supporting materials for §2 and §3

Equivalence of minimizing *ScenLoss*, minimizing MLU and maximum concurrent flow formulations. Many TE schemes [25, 43] 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 $\mathit{ScenLoss} = \max\{0, 1 - z\}$, and $\mathit{ScenLoss} = \max\{0, 1 - 1/\mathrm{MLU}\}$.

Unlike existing TE schemes, *Flexile* ensures bandwidth guarantees never degrade with additional links. Consider Fig. 16, which is similar to the topology in Fig. 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. 16, it cannot meet requirements in Fig 1 which has an additional link. *Flexile* prevents such anomalies since it ensures for any network that all flows see a loss percentile that is as small as possible.

Fair allocation in each scenario may not translate to fairness across scenarios. We show that if max-min scheme is used for every scenario, it may not lead to fairness across scenarios while *Flexile* can mitigate this effect. To see this, consider Fig 17, which depicts a topology similar to Fig. 1, except that links are directional. Notice that f_1's traffic can only be carried via link A–B, while f_2's traffic can be carried over two disjoint paths. Consider a requirement that f_1 and f_2 must each carry 1 unit of traffic 99% of the time. We make the following points:

- 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.
- With max-min, f_1 cannot meet the target. This follows from the fact that (i) f_1 can only be carried on A–B which is alive 99% of the time; and (ii) there are some scenarios where A–B is alive, yet max min only supports 0.5 units of f_1 (e.g., the scenario shown in Fig 17(b)). In contrast, f_2 meets the target with max-min. This is because whenever A–C is alive (which occurs with 99% probability), max-min routes 1 unit of f_2 on A–C (since it is the only flow that can be carried on A–C). Note that this includes a scenario such as the one shown in Fig 17(a) where no f_1 traffic can be carried.
- *Flexile* can however meet the target for both flows 99% of the time by prioritizing critical flows. For instance, the scenario shown in Fig 17(b) is critical for f_1 but not f_2, and hence *Flexile* will prioritize f_1.

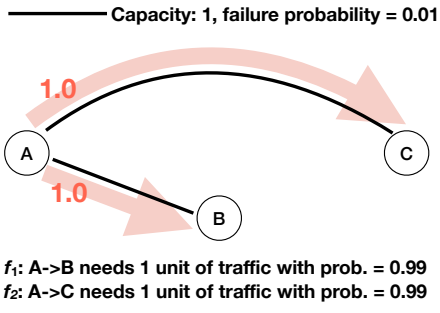


Figure 16: ScenBest can meet objectives in above topology but not in Fig.1 which has an additional link.

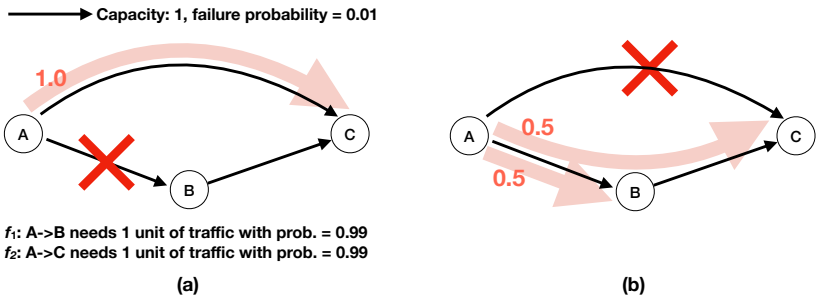


Figure 17: Example topology to illustrate unfairness with max-min across scenarios. While the network can meet the 99% requirement for both flows, max-min meets the requirements for f_2 but not f_1.

B. Supporing materials for §4

Intuition behind decomposition strategy. We present some high-level intuition for the inner workings of the decomposition approach.

First, we show that the optimal objective value for the inner problem (16) is convex in z. This follows from the fact that if, for all i, (x^i, l^i, α^i) minimizes $\sum_k \alpha_k w_k$ in (16) and yields $Penalty(z^i)$ when $z = z^i$, then for some multipliers $\lambda_i \geq 0$ such that $\sum_i \lambda_i = 1$, the solution $\sum_i \lambda_i (x^i, l^i, \alpha^i)$ is feasible to (16) when $z = \sum_i \lambda_i z^i$, and the corresponding objective value is $\sum_i \lambda_i Penalty(z^i)$. This shows that $Penalty(z^i)$ is no more than $\sum_i \lambda_i Penalty(z^i)$. Therefore, the inner problem (16) is convex in z. This property of linear programming problems is well-known; see, for example, Section 6.5 in [30].

The dual form of (16) 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 (16) gives us one point on the function $Penalty(z)$. Moreover, the dual form of (16) 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 (16) at the estimated minimizer gives a new cut and a more accurate estimate of $Penalty(z)$. 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 $PercLoss(z)$.

Explaining cut constraints in detail. Suppose we get a dual solution of (S_q), and the dual variables of (10), (11), (17) and (18) 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.

$$g_q(z_{\cdot q}) = \sum_{k,f\in F_k} (z_{fq} - 1) w_{kfq} + \sum_{k,f\in F_k} o_{kfq} + \sum_{k,i,f\in F_k, pr(f)=i} v_{kiq} d_f + \sum_e u_{eq} c_e m_{eq} \quad (21)$$

As explained in §4.2, 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 $(S_{q'})$.

$$g^q_{q'}(z_{\cdot q'}) = \sum_{k,f\in F_k} (z_{fq'} - 1) w_{kfq} + \sum_{k,f\in F_k} o_{kfq} + \sum_{k,i,f\in F_k, pr(f)=i} v_{kiq} d_f + \sum_e u_{eq} c_e m_{eq'} \quad (22)$$

Ensure better stability. We avoid oscillations in our algorithm by adding the following constraint in (M) to limit the hamming distance between current z variable and z variable achieved from last iteration.

$$\sum_{k\in K, f\in F_k, q\in Q} |z_{fq} - z'_{fq}| \leq Limit. \quad (23)$$

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.

Generating many cuts by solving a few subproblems. Our reformulation of (S_q) 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.

Capacity augmentation to meet flow percentile requirements. To generalize *Flexile* to perform minimum-cost capacity augmentation for percentile metric, 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 (6) 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 $0 \leq \delta_e \leq u_e a_e$,

where u_e is an upper bound on the augmentation.) The decomposition strategy of §4.2 generalizes to this setting where c_e is replaced with $c_e + \delta_e$ in (21) and this cut now describes a cut of *Penalty* in the (z, δ) space.

C. Supporing materials for §5

Formulations for CVaR-based schemes The following formulation, *Cvar-Flow-Ad*, minimizes the maximum conditional value at-risk across all flows. It allows the routing strategy to depend on each scenario.

$$\begin{aligned}
\min_{x,t,\theta,\alpha,s} \quad & \theta \\
\text{s.t.} \quad & \theta \geq \theta_f \quad \forall f \in F && (24)\\
& \theta_f \geq \alpha_f + \frac{1}{1-\beta} \sum_{q \in Q} p_q s_{fq} \quad \forall f \in F && (25)\\
& \alpha_f + s_{fq} \geq l_{fq} \quad \forall f \in F, q \in Q && (26)\\
& s_{fq} \geq 0 \quad \forall f \in F, q \in Q && (27)\\
& (5), (6)
\end{aligned}$$

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:

$$\begin{aligned}
\min_{x,t,\theta,\alpha,s} \quad & \theta \\
\text{s.t.} \quad & \theta \geq \theta_a \quad \forall f \in F && (28)\\
& \theta_f \geq \alpha_f + \frac{1}{1-\beta} \sum_{q \in Q} p_q s_{fq} \quad \forall f \in F && (29)\\
& \alpha_f + s_{fq} \geq l_{fq} \quad \forall f \in F, q \in Q && (30)\\
& s_{fq} \geq 0 \quad \forall f \in F, q \in Q && (31)\\
& \sum_{pr(f)=i} (1 - l_{fq}) d_f \leq \sum_{t \in R(i)} x_t y_{tq} \quad \forall i \in P, q \in Q && (32)\\
& \sum_{e \in t} x_t \leq c_e \quad \forall e \in E && (33)\\
& x_t \geq 0 \quad \forall i \in P && (34)
\end{aligned}$$

Proof of Proposition 1. 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 $v = \max_{q' \in Q'} \alpha_{q'}$, 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. □

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

Table 2: Topologies used in evaluation.

Proof of Proposition 2. Refer to Fig. 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 $\text{CVar}_{99\%}$ 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 $\text{CVar}_{99\%}$ is no more than 0.5149. Now, consider the case where link A–C fails. Since $\text{CVar}_{99\%}$ is the maximum expected loss across all flows and all sets of scenarios that occur with 1% or more probability, it follows that $0.5149 \geq \text{CVar}_{99\%} \geq 1 - \min\{f_{AB}, f_{AC}\}$ 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$^{\text{th}}$ percentile since each of the links does not fail with 0.99 probability. □

D. Supporing materials for §6

Topologies summary Our evaluation is done using 20 topologies obtained from [24] and [25]. The number of nodes and the number of edges of each topology is shown in Table 2.

Sensitivity to scale factor. By default, in our experiments of two class traffic, we scale low priority traffic by a factor of 2 (see

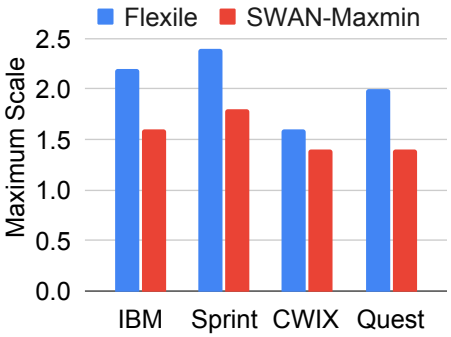


Figure 18: *Flexile* can achieve higher traffic scale.

§6). We next study the impact of scaling low priority traffic by different factors. Fig. 18 shows the maximum factor we can scale without incurring any 99%tile loss using *Flexile* and SWAN-Maxmin on different topologies. We can see that *Flexile* can support much higher scale factor than SWAN-Maxmin.