R3: Resilient Routing Reconfiguration

Ye Wang[⋆] Hao Wang[†] Ajay Mahimkar[§] Richard Alimi[⋆]
Yin Zhang[§] Lili Qiu[§] Yang Richard Yang[⋆]
Google[†] The University of Texas at Austin[§] Yale University[⋆]
{ye.wang,richard.alimi,yang.r.yang}@yale.edu wanghao@google.com
{mahimkar,yzhang,lili}@cs.utexas.edu

ABSTRACT

Network resiliency is crucial to IP network operations. Existing techniques to recover from one or a series of failures do not offer performance predictability and may cause serious congestion. In this paper, we propose Resilient Routing Reconfiguration (R3), a novel routing protection scheme that is (i) provably congestion-free under a large number of failure scenarios; (ii) efficient by having low router processing overhead and memory requirements; (iii) flexible in accommodating different performance requirements (*e.g.*, handling realistic failure scenarios, prioritized traffic, and the trade-off between performance and resilience); and (iv) robust to both topology failures and traffic variations. We implement R3 on Linux using a simple extension of MPLS, called MPLS-ff. We conduct extensive Emulab experiments and simulations using realistic network topologies and traffic demands. Our results show that R3 achieves near-optimal performance and is at least 50% better than the existing schemes under a wide range of failure scenarios.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Network communications*; C.2.2 [**Computer Communication Networks**]: Network Protocols—*Routing protocols*; C.2.3 [**Computer Communication Networks**]: Network Operations

General Terms

Algorithms, Performance, Reliability.

Keywords

Network Resiliency, Routing, Routing Protection.

1. INTRODUCTION

Motivation: Network resiliency, which we define as the ability of an IP network to recover quickly and smoothly from one or a series of failures or disruptions, is becoming increasingly important in the operation of modern IP networks. Recent large-scale deployment of delay- and loss-sensitive services such as VPN and IPTV imposes stringent requirements on the tolerable duration and level of disruptions to IP traffic. In a recent survey of major network carriers including AT&T, BT, and NTT, Telemark [36] concludes that "The 3 elements which carriers are most concerned about when deploying communication services are network reliability, network usability and network fault processing capabilities." All three elements relate to network resiliency.

Unfortunately, the current fault processing techniques to achieve resiliency are still far from ideal. Consider fast-rerouting (FRR) [32], the major technique currently deployed to handle network failures. As a major tier-1 ISP pointed out at Multi-Protocol Label Switching (MPLS) World Congress 2007, there are major practical challenges when using FRR in its business core network [34]:

- *Complexity*: "the existing FRR bandwidth and preemption design quickly becomes too complicated when multiple FRR paths are set up to account for multiple failures;"
- *Congestion*: "multiple network element failure can cause domino effect on FRR reroute due to preemption which magnifies the problem and causes network instability;"
- *No performance predictability*: "service provider loses performance predictability due to the massive amount of combinations and permutations of the reroute scenarios."

In our own survey conducted in early 2010, two of the largest ISPs in the world (one in Europe and one in Asia) gave instances of serious congestion caused by FRR in their networks.

The importance of network resiliency has attracted major attention in the research community. Many mechanisms have been recently proposed to quickly detour around failed network devices (*e.g.*, [15, 16, 29, 38]). The focus of these recent studies, however, was mainly on reachability only (*i.e.*, minimizing the duration in which routes are not available to a set of destinations). Hence, they do not address the aforementioned practical challenges, in particular on congestion and performance predictability.

It is crucial to consider congestion and performance predictability when recovering from failures. Since the overall network capacity is reduced after failures, if the remaining network resources are not efficiently utilized, serious congestion may occur. As Iyer *et al.* observed in a measurement study on a major IP backbone [17], network congestion is mostly caused by traffic that has been rerouted due to link failures. As we will show in our evaluations using real traffic scenarios, focusing only on reachability can lead to long periods of serious congestion and thus violation of service level agreements (SLAs).

However, deriving a routing protection scheme to offer performance predictability and avoid congestion is extremely challenging. The main difficulty lies in the vast number of scenarios that can result from overlapping failures (*e.g.*, [42, 28]). For example, in January 2006, the Sprint backbone network was partitioned due to two fiber cuts that happened a few days apart [42]: Sprint workers were still busy repairing the first fiber cut in California when a second fiber was cut in Arizona. Moreover, multiple IP layer links may fail together due to sharing of lower layer physical components or planned maintenance operations.

A natural approach is to enumerate all possible failure scenarios (*e.g.*, [2]). However, the number of failure scenarios quickly explodes for multiple simultaneous link failures. Consider a network with 500 links, and assume that the network needs a routing protection plan to protect against 3 simultaneous link failures. The number of such scenarios exceeds 20 million! Despite much progress on intra-domain traffic engineering, optimizing routing simultaneously for just a few hundred network topologies is already beyond the means of any existing technique that we are aware of. As a result, existing routing protection schemes have to either focus exclusively on reachability (hoping that congestion does not occur),

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.

or consider only single-link failures (which is insufficient as SLAs become ever more demanding), or face scalability issues.

Our approach: To address the above challenges, we propose **R**esilient **R**outing **R**econfiguration (R3), a novel routing protection scheme that is (i) provably congestion-free under a wide range of failure scenarios, (ii) efficient in terms of router processing overhead and memory requirement, (iii) flexible in accommodating diverse performance requirements (*e.g.*, traffic classes with different SLAs), and (iv) robust to traffic variations and topology failures.

Note that by "congestion-free", we mean that all traffic demands, except those that have lost reachability due to network partition, are routed without creating any link overload. This is a much stronger guarantee than providing reachability alone (as is commonly done in existing schemes such as FRR).

At the heart of R3 is a novel technique for covering all possible failure scenarios with a compact set of linear constraints on the amounts of traffic that should be rerouted. Specifically, when F links fail, the traffic originally routed through each failed link has to be rerouted by the remaining network. While the amount of rerouted traffic for a failed link depends on the specific failure scenario, it is always upper bounded by the capacity of the failed link (so long as the routing before the failure is congestion-free). Therefore, by creating a virtual demand for every link in the network (whose volume is equal to its link capacity) and taking the convex combination of all such virtual demands, we can cover the entire space of rerouted traffic under all possible combinations of F link failures. Since the convex hull of virtual demands can be represented as a compact set of linear constraints, we can leverage linear programming duality to efficiently optimize routing over the entire set. In essence, we eliminate the needs for enumerating failure scenarios by *converting topology uncertainty (due to failures) into uncertainty in rerouted traffic*, which is easier to cope with.

Since the virtual demands are upper bounds of the rerouted traffic, we guarantee that if a routing is congestion-free over the actual demand plus the virtual demand set, it yields a link-based protection scheme that is congestion-free under all possible failure scenarios. Interestingly, the converse is also true for single-link failures: if there exists a link-based protection scheme that can guarantee no congestion for all single-link failure scenarios, then there must be a routing that is congestion-free over the actual demand plus the virtual demand set. Thus, the seemingly wasteful replacement of rerouted traffic with link capacities is actually efficient.

Based on the above idea, we develop R3 that consists of an offline precomputation phase and an online reconfiguration phase. During the offline phase, we compute routing for the actual demand plus the virtual demand on the original network topology. During the online phase, R3 responds to failures using a simple rescaling procedure, which converts the offline precomputed routing into a protection routing that does not traverse any failed links. A unique feature of R3 is that it is provably congestion-free under multiple link failures and optimal for single-link failure scenarios. We further extend R3 to handle (i) traffic variations, (ii) realistic failure scenarios, (iii) prioritized traffic with different protection levels, (iv) trade-off between performance under normal conditions and failures, and (v) trade-off between network utilization and delay.

We implement R3 protection using MPLS-ff, a simple extension of MPLS, while the base routing can use either OSPF or MPLS. Our Emulab evaluation and simulation based on real Internet topologies and traffic traces show that R3 achieves near-optimal performance. Its performance is at least 50% better than existing schemes including OSPF reconvergence, OSPF with CSPF fast rerouting, FCP [26], and Path Splicing [29].

2. OVERVIEW

A traditional traffic engineering algorithm computes an effective *base routing* r that optimizes a network metric, such as minimizing congestion cost or maximum link utilization (*e.g.*, [12, 13, 31, 39]). Then, a *protection routing* p is derived from r, for example, through fast rerouting (FRR). While simple and well studied, this traditional approach can easily result in serious network congestion and performance unpredictability under failures. Below we first formally define the problem of resilient routing and explain why it is challenging. We then introduce the key ideas of R3.

Notations: Let $G = (V, E)$ be an IP network under consideration, where V is the set of routers in the network, and E is the set of network links connecting the routers. Let d be the traffic matrix between the routers in V, where d_{ab} is the traffic demand originated at router a to router b. Let c_e or c_{ij} denote the capacity of a directed link $e = (i, j)$ from router i to router j. We refer to i as the source node of link e and j as its tail node.

To define routing precisely, we use the flow representation of routing [3, 6]. Formally, a flow representation of a routing r is specified by a set of values $\{r_{ab}(e)|a, b \in V, e \in E\}$, where $r_{ab}(e)$ or $r_{ab}(i, j)$ specifies the fraction of traffic for the origin-destination (OD) pair $a \to b$ that is routed over link $e = (i, j)$. For actual traffic d_{ab} of the OD pair $a \to b$, the contribution of this traffic to the load on link e is $d_{ab}r_{ab}(e)$. For $\{r_{ab}(e)\}$ to be a valid routing for a given OD pair $a \neq b$, it should satisfy the following conditions:

$$\begin{array}{ll} [R1] & \forall i \neq a, b : \sum_{(i,j)\in E} r_{ab}(i, j) = \sum_{(j,i)\in E} r_{ab}(j, i); \\ [R2] & \sum_{(a,i)\in E} r_{ab}(a, i) = 1; \\ [R3] & \forall (i, a) \in E : r_{ab}(i, a) = 0; \\ [R4] & \forall e \in E : 0 \leq r_{ab}(e) \leq 1. \end{array} \qquad (1)$$

The first condition indicates flow conservation at any intermediate nodes. The second condition specifies that all traffic from a source should be routed. The third condition prevents traffic from returning to the source. Finally, according to the definition of $r_{ab}(e)$, it is between 0 and 1.

Problem formulation: We consider the following basic formulation of resilient routing. In Section 3.5, we provide several useful extensions to the basic problem formulation.

DEFINITION 1 (RESILIENT ROUTING). *The problem of resilient routing is to design an effective base routing* r *and protection routing* p *for traffic matrix* d *to ensure that the network is congestion-free (*i.e.*, the maximum link utilization stays below 100%) under all possible failure scenarios involving up to F failed links. The base routing* r *can also be given as an input (*e.g.*, by OSPF), in which case only the protection routing* p *needs to be designed.*

Multiple protection routing schemes are possible. To minimize disruption and control overhead, we only consider protection routing schemes that change the route of an OD pair when the OD pair traverses a failed link before it fails. Among this class of routing reconfiguration techniques, *link-based protection* is the most widely used. Thus, we focus on link-based protection, but our scheme can extend to path-based protection, which can be viewed as a special case of link-based protection in an overlay topology. In link-based protection, the source node of a failed link reroutes the traffic originally passing through the failed link along a detour route to reach the tail node of the link. Thus, the protection routing p only needs to be defined for each link that requires protection; in contrast, the base routing r defines routing for each OD pair.

Challenge in coping with topology uncertainty: Due to the frequency of failures, the delay in failure recovery [17, 26] and increasingly stringent SLAs for network services, it is essential for resilient routing to avoid congestion under multiple link failures overlapping in time. This requires the design of resilient routing to explicitly consider all possible failure scenarios. One natural approach to resilient routing is to enumerate all failure scenarios (*e.g.*, [2]) and derive a routing that works well for all these scenarios. However, this approach faces serious scalability and efficiency issues. Suppose a network with $|E|$ links needs to handle up to F link failures. Then there will be $\sum_{i=1}^{F} \binom{|E|}{i}$ failure scenarios, which result in prohibitive computation and configuration cost even for a small number of failures. On the other hand, in order to achieve the congestion-free guarantee, it is imperative to *protect* against all of the $\sum_{i=1}^{F}\binom{|E|}{i}$ scenarios, since a skipped scenario may arise in practice and cause network congestion and SLA violation. Therefore, fundamental challenges in achieving resilient routing involve (i) efficient computation of protection routing that is provably congestion-free even under multiple failures and (ii) simple re-configuration in response to failures.

From topology uncertainty to traffic uncertainty: The key idea of R3 is to convert topology uncertainty (due to various failure scenarios) into *traffic uncertainty* that captures the different traffic demands that need to be rerouted under different failure scenarios.

Specifically, suppose we want to design a routing so that we can protect against up to F arbitrary link failures. Under link-based protection, the rest of the network needs to carry traffic previously carried by the failed links. It is easy to see that the rerouted traffic is upper bounded by the capacity of each failed link (as long as no link is fully utilized under the base routing $\mathtt{r}$). Therefore, for every link in the network we create a virtual demand that is equal to its link capacity; the convex combination of all such virtual demands should cover the entire space of rerouted traffic. Formally, for each link $e \in E$, we associate a virtual demand variable x_e. We then form a *rerouting virtual demand set* X_F as

$$X_F \triangleq \left\{ x \,\middle|\, 0 \le \tfrac{x_e}{c_e} \le 1 \; (\forall e \in E), \textstyle\sum_{e\in E} \tfrac{x_e}{c_e} \le F \right\}. \qquad (2)$$

For any failure scenario that involves up to F link failures, the traffic that needs to be rerouted always belongs to the set X_F. Therefore, X_F represents an envelope (*i.e.*, superset) of the rerouted traffic under all possible failure scenarios.

Instead of optimizing the routing for the *fixed* traffic matrix d on a *variable* topology under all possible failure scenarios, we seek a routing that works well for the entire demand set $d + X_F$ but on the *fixed* original topology, where $d + X_F \triangleq \{d + x | x \in X_F\}$ denotes the sum of the actual demand d and the set of virtual demands X_F. In this way, we convert topology uncertainty into traffic uncertainty.

At the first glance, converting topology uncertainty into traffic uncertainty makes the problem more challenging, because the number of failure scenarios is at least finite, whereas $d + X_F$ contains an infinite number of traffic matrices. However, the rerouting virtual demand set X_F can be represented by a compact set of linear constraints in (2). By applying linear programming duality, we can find the optimal base routing $\mathtt{r}$ and protection routing $\mathtt{p}$ for the entire demand set $d + X_F$ without enumerating traffic matrices.

Another potential concern is that the definition of the rerouting virtual demand set X_F appears rather wasteful. When links $e_1, \cdots, e_F$ fail, the corresponding virtual demands in X_F can be as large as $x_{e_i} = c_{e_i}$ $(i = 1, \cdots, F)$. That is, we replace the rerouted traffic on failed link e_i with a virtual demand equal to the link capacity c_{e_i}. Interestingly, we prove in Section 3.4 that the seemingly wasteful replacement of rerouted traffic with link capacities is necessary at least for $F = 1$. Specifically, if there exists a link-based protection routing that guarantees no congestion for all single-link failure scenarios, then there must exist a routing that is congestion-free over the entire demand set $d + X_F$.

R3 overview: Based on the previous insight, we develop R3 that consists of the following two main phases:

- *Offline precomputation.* During the offline precomputation phase, R3 computes routing $\mathtt{r}$ (if not given) for traffic matrix d and routing $\mathtt{p}$ for rerouting virtual demand set X_F to minimize the maximum link utilization on the original network topology over the combined demand set $d + X_F$. The optimization is made efficient by leveraging linear programming duality.
- *Online reconfiguration.* During the online reconfiguration phase, after a failure, R3 applies a simple procedure called *rescaling* to convert $\mathtt{p}$ (which is defined on the original network topology and thus may involve the failed link) into a protection routing that does not traverse any failed link and thus can be used to reroute traffic on the failed links. The rescaling procedure is efficient and can be applied in real-time with little computation and memory overhead.

A unique feature of R3 is that it can provide several provable theoretical guarantees. In particular, R3 guarantees no congestion under a wide range of failure scenarios involving multiple link failures. As a result, it provides much stronger guarantee than simple reachability. Moreover, the conversion from topology uncertainty into traffic uncertainty is efficient in that the seemingly wasteful replacement of rerouted traffic with link capacity is indeed necessary for single-link failure scenarios. Finally, the online reconfiguration procedure is independent of the order in which the failed links are detected. Thus, routers can apply R3 independently.

In addition, R3 admits a number of useful extensions for (i) coping with traffic variations, (ii) supporting realistic failure scenarios, (iii) accommodating prioritized traffic with different protection levels, (iv) balancing the trade-off between performance under normal conditions and failures, and (v) balancing the trade-off between network utilization and delay.

3. R3 DESIGN

In this section, we present the detailed design of R3. We describe offline precomputation in Section 3.1 and online reconfiguration in Section 3.2, followed by an illustrative example in Section 3.3. We prove the theoretical guarantees of R3 in Section 3.4. We also introduce several useful extensions to R3 in Section 3.5.

3.1 Offline Precomputation

Problem formulation: The goal of offline precomputation is to find routing $\mathtt{r}$ for traffic matrix d and routing $\mathtt{p}$ for rerouting virtual demand set X_F defined in (2) to minimize the maximum link utilization (MLU) over demand set $d + X_F$. This is formulated as a problem shown in (3). The objective is to minimize MLU over the entire network. To remove redundant routing traffic forming a loop, we can either add to the objective a small penalty term including the sum of routing terms or postprocess. For clearer presentation, we focus on MLU as the objective. Constraint [C1] ensures that $\mathtt{r}$ and $\mathtt{p}$ are valid routing, *i.e.*, they both satisfy routing constraints (1). Constraint [C2] enforces all links have utilization below MLU.

$$\begin{array}{ll} \multicolumn{2}{l}{\textbf{minimize}_{(\mathtt{r},\mathtt{p})}\; MLU} \\ \multicolumn{2}{l}{\textbf{subject to}:} \\ \text{[C1]} & \mathtt{r} = \{r_{ab}(e) | a, b \in V, e \in E\} \text{ is a routing;} \\ & \mathtt{p} = \{p_\ell(e) | \ell, e \in E\} \text{ is a routing;} \\ \text{[C2]} & \forall x \in X_F, \forall e \in E: \\ & \frac{\sum_{a,b\in V} d_{ab} r_{ab}(e) + \sum_{l\in E} x_l p_l(e)}{c_e} \le MLU. \end{array} \qquad (3)$$

Note that $\mathtt{p}$ is defined for each link whereas $\mathtt{r}$ is defined for each OD pair. Also note that when $\mathtt{r}$ is pre-determined (*e.g.*, by OSPF), $\mathtt{r}$ becomes an input to the optimization in (3) instead of consisting of optimization variables.

Solution strategy: A key challenge in solving (3) is that there is a constraint [C2] for every element x belonging to the rerouting virtual demand set X_F. Since X_F has an infinite number of elements, the number of constraints is infinite. Fortunately, we can apply linear programming duality to convert (3) into an equivalent, simpler linear program with a polynomial number of constraints as follows.

First, constraint [C2] in (3) is equivalent to:

$$\forall e \in E: \; \frac{\sum_{a,b\in V} d_{ab} r_{ab}(e) + ML(\mathtt{p}, e)}{c_e} \le MLU, \qquad (4)$$

where $ML(\mathtt{p}, e)$ is the maximum load on e for $\forall x \in X_F$, and thus is the optimal objective of the following problem:

$$\begin{array}{ll} \textbf{maximize}_x & \sum_{l\in E} x_l p_l(e) \\ \textbf{subject to}: & \left\{ \begin{array}{l} \forall \ell \in E: x_\ell / c_\ell \le 1; \\ \sum_{\ell\in E} x_\ell / c_\ell \le F. \end{array} \right. \end{array} \qquad (5)$$

Here (5) is a linear program when $\mathtt{p}$ is a fixed input. From linear programming duality [5], the optimal objective of (5), $ML(\mathtt{p}, e)$, is no more than a given upper bound UB if and only if there exist dual multipliers $\pi_e(\ell)$ ($\ell \in E$) and λ_e such that:

$$\begin{array}{l} \sum_{\ell \in E} \pi_e(\ell) + \lambda_e F \leq UB; \\ \forall \ell \in E : \frac{\pi_e(\ell)+\lambda_e}{c_\ell} \geq p_\ell(e); \\ \forall \ell \in E : \pi_e(\ell) \geq 0; \\ \lambda_e \geq 0. \end{array} \tag{6}$$

Here $\pi_e(\ell)$ is the dual multiplier for constraint $x_\ell/c_\ell \leq 1$, λ_e is the dual multiplier for $\sum_{\ell \in E} x_\ell/c_\ell \leq F$, and the subscript e indicates that (5) computes the maximum load on link e.

Since all of the constraints in (6) are linear, we convert (4) into a set of linear constraints by substituting $ML(\mathtt{p}, e)$ with $\sum_{\ell \in E} \pi_e(\ell) + \lambda_e F$ and incorporating (6). We can show that the original problem (3) then becomes the following equivalent linear program, which we solve using `cplex` [8]:

$$\begin{array}{l} \textbf{minimize}_{(\mathtt{r},\mathtt{p},\pi,\lambda)} \; MLU \\ \textbf{subject to}: \\ \left\{ \begin{array}{l} \mathtt{r} = \{r_{ab}(e) | a, b \in V, e \in E\} \text{ is a routing;} \\ \mathtt{p} = \{p_\ell(e) | \ell, e \in E\} \text{ is a routing;} \\ \forall e \in E : \\ \quad \frac{\sum_{a,b \in V} d_{ab} r_{ab}(e) + \sum_{l \in E} \pi_e(\ell) + \lambda_e F}{c_e} \leq MLU; \\ \forall e, \ell \in E : \frac{\pi_e(\ell)+\lambda_e}{c_\ell} \geq p_\ell(e); \\ \forall e, \ell \in E : \pi_e(\ell) \geq 0; \\ \forall e \in E : \lambda_e \geq 0. \end{array} \right. \end{array} \tag{7}$$

Complexity: Linear program (7) has $O(|V|^2 \cdot |E| + |E|^2)$ variables and $O(|V|^3 + |E|^2)$ constraints. Even if we just want to find $\mathtt{r}$ to minimize the MLU for fixed traffic matrix d, routing constraints (1) already have $O(|V|^2 \cdot |E|)$ variables and $O(|V|^3)$ constraints. In most networks, $|E|^2 \leq |V|^3$. So (7) only causes moderate increase in the size of the linear program. Note that linear programming duality has also been exploited in recent research on oblivious routing [3, 39]. However, [3, 39] require $O(|V| \cdot |E|^2)$ constraints, which is much higher than (7).

3.2 Online Reconfiguration

After the failure of link e is detected, two main tasks are performed by online reconfiguration. First, the source router of e immediately reroutes the traffic originally traversing e through a detour route. Second, in preparation for additional link failures, every router adjusts $\mathtt{r}$ and $\mathtt{p}$ so that no demand traverses the failed link e.

Fast rerouting of traffic on the failed link: After link e fails, the source router of e immediately uses $\mathtt{p}$ to derive a detour route (denoted by ξ_e) to reroute the traffic that traverses e before it fails. Note that we cannot directly use $p_e = \{p_e(\ell) | \ell \in E\}$ as the detour route ξ_e, because p_e is defined on the original topology and may assign non-zero traffic to e, *i.e.*, $p_e(e) > 0$. Fortunately, $\{p_e(\ell) | \ell \neq e\}$ already satisfies routing constraints [R1], [R3] and [R4] in (1). To convert it into a valid detour route ξ_e, we just need to perform the following simple re-scaling to ensure that all traffic originally traversing e is rerouted (thus satisfying [R2]):

$$\xi_e(\ell) = \frac{p_e(\ell)}{1-p_e(e)} \quad (\forall \ell \in E \setminus \{e\}). \tag{8}$$

An example illustrating this procedure is provided in Section 3.3.

Note that when $p_e(e) = 1$, we simply set $\xi_e(\ell) = 0$. As we show later, under the condition of Theorem 1, $p_e(e) = 1$ implies that link e carries no actual demand from any OD pairs or virtual demand from links other than e. So link e does not need to be protected and can be safely ignored.

Adjusting $\mathtt{r}$ and $\mathtt{p}$ to exclude the failed link: In preparation for additional link failures, R3 adjusts $\mathtt{r}$ and $\mathtt{p}$ to ensure that no (actual or virtual) demand traverses the failed link e. This can be achieved by moving the original traffic allocation on link e to the detour route ξ_e. Specifically, let $E' = E \setminus \{e\}$ and $G' = (V, E')$. The updated base routing $\mathtt{r'}$ is defined as:

$$r'_{ab}(\ell) = r_{ab}(\ell) + r_{ab}(e) \cdot \xi_e(\ell), \; \forall a, b \in V, \forall \ell \in E', \tag{9}$$

where $r_{ab}(\ell)$ is the original allocation on link ℓ for OD pair $a \rightarrow b$, and $r_{ab}(e) \cdot \xi_e(\ell)$ gives the increase due to using ξ_e to reroute the original allocation on the failed link (*i.e.*, $r_{ab}(e)$). Similarly, the updated protection routing $\mathtt{p'}$ is defined as:

$$p'_{uv}(\ell) = p_{uv}(\ell) + p_{uv}(e) \cdot \xi_e(\ell), \; \forall (u, v) \in E', \forall \ell \in E'. \tag{10}$$

Efficiency: All the operations in online reconfiguration are simple and thus highly efficient. Specifically, computing ξ_e from $\mathtt{p}$ requires only simple rescaling of $\{p_e(\ell)\}$. The rescaling operation can also be avoided if we directly store $p_e(e)$ and $\xi_e = \left\{ \frac{p_e(\ell)}{1-p_e(e)} | \ell \neq e \right\}$ instead of $\{p_e(\ell) | \ell \in E\}$. Meanwhile, updating $r'_{ab}(\ell)$ and $p'_{uv}(\ell)$ is also extremely simple and is only required for demands with non-zero traffic allocation on the failed link (*i.e.*, $r_{ab}(e) > 0$ and $p_{uv}(e) > 0$). Note that R3 does not require all routers to finish updating their $\mathtt{r}$ and $\mathtt{p}$ before recovering from the failed link e – the recovery reaches full effect as soon as the source router of e starts rerouting traffic through the detour route ξ_e.

3.3 R3 Example

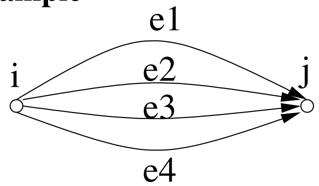


Figure 1: A simple example network with 4 parallel links.

To illustrate R3 protection routing and re-scaling, consider a simple network (shown in Figure 1) with 4 parallel links e_1, e_2, e_3, and e_4. For such a network, one can verify that the offline optimization results in a protection routing that splits each rerouting virtual demand among all 4 links in proportion to their link capacities.

Suppose the protection routing $\mathtt{p}$ for virtual demand e_1 specifies that $p_{e_1}(e_1) = 0.1$, $p_{e_1}(e_2) = 0.2$, $p_{e_1}(e_3) = 0.3$, and $p_{e_1}(e_4) = 0.4$. After e_1 fails, source router i detours the real traffic originally traversing e_1 through e_2, e_3, and e_4 in proportion to $p_{e_1}(e_2)$, $p_{e_1}(e_3)$, and $p_{e_1}(e_4)$. This is equivalent to using a detour route ξ_{e_1}, where $\xi_{e_1}(e_i) = \frac{p_{e_1}(e_i)}{\sum_{j=2}^{4} p_{e_1}(e_j)} = \frac{p_{e_1}(e_i)}{1-p_{e_1}(e_1)}$ for $i = 2, 3, 4$. This is effectively the re-scaling in (8) and yields $\xi_{e_1}(e_2) = \frac{2}{9}$, $\xi_{e_1}(e_3) = \frac{3}{9}$, and $\xi_{e_1}(e_4) = \frac{4}{9}$.

Now consider the protection routing of e_2 in $\mathtt{p}$, which also specifies that $p_{e_2}(e_1) = 0.1$, $p_{e_2}(e_2) = 0.2$, $p_{e_2}(e_3) = 0.3$, and $p_{e_2}(e_4) = 0.4$. After link e_1 fails, protection plan p_{e_2} is no longer valid, as it uses e_1 to carry the traffic originally passing through e_2, but e_1 is unavailable and e_1's protection even needs e_2. To solve this issue, we need to update p_{e_2} so that the usage of e_1 is replaced by remaining links. For this purpose, we use the same detour route ξ_{e_1} to detour the fraction $p_{e_2}(e_1) = 0.1$ of virtual demand x_{e_2} originally traversing e_1. This detour splits $p_{e_2}(e_1)$ to e_2, e_3 and e_4 in proportion to $\xi_{e_1}(e_2)$, $\xi_{e_1}(e_3)$, and $\xi_{e_1}(e_4)$, yielding $p'_{e_2}(e_i) = p_{e_2}(e_i) + p_{e_2}(e_1) \cdot \xi_{e_1}(e_i)$ for $i = 2, 3, 4$. This is effectively the reconfiguration in (10). Thus, after e_1 fails, the protection routing for e_2 becomes $p'_{e_2}(e_1) = 0$, $p'_{e_2}(e_2) = 0.2 + 0.1 \cdot \frac{2}{9}$, $p'_{e_2}(e_3) = 0.3 + 0.1 \cdot \frac{3}{9}$, and $p'_{e_2}(e_4) = 0.4 + 0.1 \cdot \frac{4}{9}$.

3.4 Theoretical Guarantees of R3

Sufficient condition for congestion-free guarantee: A key feature of R3 is that it can provide provable congestion-free guarantee under all possible failure scenarios as long as the optimal MLU in (7) is below 1. More formally, we have:

THEOREM 1. *Let X_F be the rerouting virtual demand set with up to F link failures, as defined in* (2)*. If offline precomputation (Section 3.1) obtains routing* $\mathtt{r}$ *and* $\mathtt{p}$ *such that the MLU for the entire demand set $d+X_F$ is no larger than* 1 *on the original topology $G=(V,E)$, then online reconfiguration (Section 3.2) guarantees that the MLU for the real traffic matrix d and the rerouted traffic is no larger than* 1 *under any failure scenario with up to F failed links.*

PROOF. Let e be the first failed link. Let $E' = E \setminus \{e\}$. Let $\mathtt{r'}$ and $\mathtt{p'}$ be the updated routing after online reconfiguration. Let X_{F-1} be the rerouting virtual demand set with up to $F-1$ failures in E'. Below we prove that $\mathtt{r'}$ and $\mathtt{p'}$ guarantee that the MLU for demand set $d+X_{F-1}$ is no larger than 1 on the new topology $G' = (V, E')$. Consider any $\ell \in E'$ and $x \in X_{F-1}$. Let $L(d, x, \mathtt{r}', \mathtt{p}', \ell)$ be the load on link ℓ coming from real traffic d and virtual demand x using base routing $\mathtt{r'}$ and protection routing $\mathtt{p'}$. We have:

$$
\begin{aligned}
&L(d, x, \mathtt{r}', \mathtt{p}', \ell)\\
&\quad= \textstyle\sum_{a,b\in V} d_{ab} r'_{ab}(\ell) + \sum_{(u,v)\in E'} x_{uv} p'_{uv}(\ell)\\
&\quad= \textstyle\sum_{a,b\in V} d_{ab}(r_{ab}(\ell) + r_{ab}(e)\xi_e(\ell))\\
&\qquad \textstyle+\sum_{(u,v)\in E'} x_{uv}(p_{uv}(\ell) + p_{uv}(e)\xi_e(\ell))\\
&\quad= L(d, x, \mathtt{r}, \mathtt{p}, \ell) + L(d, x, \mathtt{r}, \mathtt{p}, e) \cdot \tfrac{p_e(\ell)}{1-p_e(e)}. \qquad (11)
\end{aligned}
$$

Given $x \in X_{F-1}$, we can obtain $y \in X_F$ by adding a virtual demand for the failed link e to x. That is, $y_e = c_e$ and $y_{uv} = x_{uv}$ for $\forall(u, v) \in E'$. Since $\mathtt{r}$ and $\mathtt{p}$ guarantee no congestion for $d + X_F$, we have:

$$
\begin{aligned}
c_\ell &\ge L(d, y, \mathtt{r}, \mathtt{p}, \ell) = L(d, x, \mathtt{r}, \mathtt{p}, \ell) + c_e \cdot p_e(\ell); \qquad (12)\\
c_e &\ge L(d, y, \mathtt{r}, \mathtt{p}, e) = L(d, x, \mathtt{r}, \mathtt{p}, e) + c_e \cdot p_e(e). \qquad (13)
\end{aligned}
$$

From (13) and when $p_e(e) < 1$, we have:

$$
c_e \ge L(d, x, \mathtt{r}, \mathtt{p}, e)/(1 - p_e(e)). \qquad (14)
$$

Substituting c_e in (12) with the R.H.S. of (14), we have:

$$
c_\ell \ge L(d, x, \mathtt{r}, \mathtt{p}, \ell) + L(d, x, \mathtt{r}, \mathtt{p}, e)\tfrac{p_e(\ell)}{1-p_e(e)}. \qquad (15)
$$

Combining (11) and (15), we have $c_\ell \ge L(d, x, \mathtt{r}', \mathtt{p}', \ell)$ (for $\forall \ell \in E'$). Note that this also holds when $p_e(e) = 1$. In this case, under our assumption that $MLU \le 1$, no other actual or virtual demand traverses e and thus needs to be rerouted. So we simply set $\xi_e(\ell) = 0$ and $L(d, x, \mathtt{r}', \mathtt{p}', \ell) = L(d, x, \mathtt{r}, \mathtt{p}, \ell) \le c_\ell$. Therefore, $\mathtt{r'}$ and $\mathtt{p'}$ guarantee that the MLU for $d+X_{F-1}$ on $G' = (V, E')$ is no larger than 1. Thus, $\mathtt{r'}$ guarantees that the MLU for d is no larger than 1. By induction, we can then prove that d is congestion-free for any failure scenario with up to F failed links. □

Note that depending on the value of F and the connectivity of G, we may not be able to find $\mathtt{r}$ and $\mathtt{p}$ that meet the sufficient condition. For example, if there exist F failures that partition the network, then it is impossible to find $\mathtt{r}$ and $\mathtt{p}$ to ensure that the MLU is no larger than 1 for the entire demand set $d + X_F$. Interestingly, our evaluation shows that when such a scenario occurs, the online reconfiguration of R3 can automatically remove those demands that have lost reachability due to the partition of the network (by setting $\xi_e(\ell) = 0$ when $p_e(e) = 1$). Moreover, by choosing $\mathtt{r}$ and $\mathtt{p}$ that minimize the MLU over the entire demand set $d+X_F$, R3 achieves much lower MLU than existing methods.

Necessary condition for resilient routing: A potential concern on R3 is that it may be rather wasteful, as it enforces MLU to be within 1 when routing both real traffic and rerouting virtual demand up to the link capacity. However, it is more economical than it seems. In particular, Theorem 2 shows that the requirement in Theorem 1 is tight for single-link failures (*i.e.*, $F = 1$). Our evaluation will further show that it is efficient under general failure scenarios.

THEOREM 2. *Let X_1 be the rerouting virtual demand set for* single-link *failures, as defined in* (2)*. If there exists base routing* $\mathtt{r}$ *and link-based protection routing* $\mathtt{p}^*$ *such that for all cases of* single-link *failures, the MLU (due to both regular traffic and rerouted traffic) is no larger than* 1 *and there is no traffic loss due to network partitioning, then there exists* $\mathtt{p}$ *such that with* $\mathtt{r}$ *and* $\mathtt{p}$*, $d + X_1$ can be routed without creating any congestion.*

PROOF. Let $L(d, \mathtt{r}, e) = \sum_{a,b\in V} d_{ab} r_{ab}(e)$ be the load on link e due to real traffic d and base routing $\mathtt{r}$. We construct $\mathtt{p}$ as:

$$
\forall e, \ell \in E : p_e(\ell) = \begin{cases} 1 - \frac{L(d,\mathtt{r},e)}{c_e}, & \text{if } \ell = e;\\ p^*_e(\ell) \cdot \frac{L(d,\mathtt{r},e)}{c_e}, & \text{otherwise.} \end{cases} \qquad (16)
$$

We next show that the resulted routing $\mathtt{p}$ together with the base routing $\mathtt{r}$ ensures that there is no congestion for demand set $d+X_1$. Since MLU is a convex function, the maximum MLU for routing ($\mathtt{r}$,$\mathtt{p}$) over the entire demand set $d + X_1$ will be reached at an extreme point of $d + X_1$, which corresponds to having a single $x_e/c_e = 1$ and all the other $x_\ell/c_\ell = 0$ ($\forall \ell \ne e$). It is easy to see that for $\forall \ell \ne e$, we have

$$
\begin{aligned}
L(d, x, \mathtt{r}, \mathtt{p}, \ell) &= L(d, \mathtt{r}, \ell) + x_e p_e(\ell)\\
&= L(d, \mathtt{r}, \ell) + c_e \tfrac{L(d,\mathtt{r},e)}{c_e} p^*_e(\ell)\\
&= L(d, \mathtt{r}, \ell) + L(d, \mathtt{r}, e)p^*_e(\ell).
\end{aligned}
$$

That is, $L(d, x, \mathtt{r}, \mathtt{p}, \ell)$ is the same as the link load on ℓ when protection routing p^*_e is used to reroute traffic traversing the failed link e, which is no larger than c_ℓ by assumption. Meanwhile, it is easy to see that:

$$
\begin{aligned}
L(d, x, \mathtt{r}, \mathtt{p}, e) &= L(d, \mathtt{r}, e) + x_e p_e(e)\\
&= L(d, \mathtt{r}, e) + c_e\left(1 - \tfrac{L(d,\mathtt{r},e)}{c_e}\right) = c_e.
\end{aligned}
$$

Therefore, the maximum MLU is no larger than 1 under routing ($\mathtt{r}$, $\mathtt{p}$) over the entire demand set $d + X_1$. □

The case with multiple link failures is more challenging. As a special case, we consider a network with two nodes and multiple parallel links connecting them (*e.g.*, the network shown in Figure 1). We have the following proposition:

PROPOSITION 1. *For a network with two nodes connected by parallel links, R3 protection routing and reconfiguration produce an optimal protection routing under any number of failed links.*

We leave the tightness of R3 in a general network topology with multiple link failures as an open problem.

Order independent online reconfiguration: In case multiple link failures occur close in time, it is possible that different routers may detect these failures in different orders. Theorem 3 ensures that the online reconfiguration procedure in Section 3.2 will eventually result in the same routing as long as different routers eventually discover the same set of failed links. In other words, the order in which the failures are detected does not affect the final routing. This is useful because different routers can then apply R3 in a distributed, independent fashion without requiring any central controller to synchronize their routing states.

THEOREM 3. *The online reconfiguration procedure is order independent. That is, any permutation of a failure sequence e_1, e_2, $\cdots$, e_n always yields the same routing after online reconfiguration has been performed for all links in the permuted failure sequence.*

PROOF. Omitted in the interest of brevity. □

3.5 Extensions

Handling traffic variations: So far we consider a fixed traffic matrix d. In practice, traffic varies over time. To accommodate such variations, a traffic engineering system typically collects a set of traffic matrices $\{d_1, \cdots, d_H\}$ and uses their convex combination to cover the space of common traffic patterns (*e.g.*, see [31, 44, 39]). That is, we replace a fixed traffic matrix d with the convex hull of $\{d_1, \cdots, d_H\}$:

$$D \triangleq \left\{ d \mid d = \sum_{h=1}^{H} t_h d_h, \sum_{h=1}^{H} t_h = 1, t_h \geq 0 \, (\forall h) \right\}.$$

Constraint [C2] in (3) then becomes:

$$\begin{aligned} &\forall d \in D, \forall x \in X_F, \forall e \in E : \\ &\quad \frac{\sum_{a,b \in V} d_{ab} r_{ab}(e) + \sum_{\ell \in E} x_\ell p_\ell(e)}{c_e} \leq MLU. \end{aligned} \tag{17}$$

As in Section 3.1, we can apply linear programming duality to convert (17) into a set of linear constraints.

Handling realistic failure scenarios: We have considered arbitrary K link failures. Next we take into account of potential structures in realistic failure scenarios and classify failure events into the following two classes:

- Shared risk link group (SRLG). A SRLG consists of a set of links that are disconnected simultaneously. For example, due to sharing of lower layer physical components (*e.g.*, optical switch), multiple IP layer links may always fail together. Another example is the high-bandwidth composite links, in which a single member link down will cause all links in the composite link to be shut down. Let F_{SRLG} be the set consisting of all SRLGs. Each element in F_{SRLG} consists of a set of links.
- Maintenance link group (MLG). A network operator may shut down a set of links in the same maintenance operation. Let F_{MLG} be the set consisting of all MLG events. Each element F_{MLG} consists of a set of links.

To capture these failure characteristics, we introduce an indicator variable I_f, where $I_f = 1$ if and only if the basic event set f is down. Then (5) is changed to (18), where the first constraint limits the maximum number of concurrent SRLGs, the second constraint expresses the fact that maintenance is carefully scheduled so that at most one MLG undergoes maintenance at any instance of time, and the last constraint encodes the fact that the rerouting traffic for a link is upperbounded by whether the link belongs to any SRLG or MLG. We can then apply linear programming duality in a similar way to compute resilient routing.

$$\begin{aligned} &\textbf{maximize}_x \sum_{l \in E} p_l(e) x_l \\ &\textbf{subject to} : \\ &\begin{cases} \sum_{f \in \mathcal{F}_{\mathcal{SRLG}}} I_f \leq K; \\ \sum_{f \in \mathcal{F}_{\mathcal{MLG}}} I_f \leq 1; \\ \forall e \in E : \frac{x_e}{c_e} \leq 1; \\ \forall e \in E : \frac{x_e}{c_e} \leq \sum\limits_{f \in \mathcal{F}_{\mathcal{SRLG}} : e \in f} I_f + \sum\limits_{f \in \mathcal{F}_{\mathcal{MLG}} : e \in f} I_f. \end{cases} \end{aligned} \tag{18}$$

Supporting prioritized resilient routing: So far, we consider that all traffic requires equal protection. Operational networks increasingly provide different levels of SLAs to different classes of traffic. For example, some traffic has a more stringent SLA requirement and needs to tolerate more overlapping link failures. Given an SLA requirement, we can translate it into the number of overlapping link failures to tolerate. We extend R3 to support prioritized resilient routing by associating each class of traffic with a protection level.

Let F_i be the number of link failures that traffic with protection level i should tolerate. Let d^i be the total traffic demands that require protection level i or higher. Let X_{F_i} be the rerouting virtual demand set with up to F_i failures. Then our goal is to find $(\mathtt{r}, \mathtt{p})$ such that for any i, the network has no congestion for the entire demand set $d^i + X_{F_i}$. To achieve this goal, we simply replace [C2] in (3) with (19), which can again be converted into linear constraints by applying linear programming duality:

$$\begin{aligned} &\forall i, \forall x^i \in X_{F_i}, \forall e \in E : \\ &\quad \frac{\sum_{a,b \in V} d^i_{ab} r_{ab}(e) + \sum_{l \in E} x^i_l p_l(e)}{c_e} \leq MLU. \end{aligned} \tag{19}$$

As an example, consider a network with three traffic protection classes: TPRT d_F for real-time IP, TPP d_P for private transport, and general IP d_I, with decreasing protection levels: TPRT should be protected against up to three link failures, TPP up to two link failures, and IP any single-link failure scenarios. Then the algorithm computes $d^1 = d_F + d_P + d_I$, $d^2 = d_F + d_P$, $d^3 = d_F$, and sets $F_i = i$, for $i = 1, 2, 3$. This essentially means that resilient routing should carry $d^1 + X_1$, $d^2 + X_2$, and $d^3 + X_3$, where X_i denotes the rerouting virtual demand set with up to i link failures.

Trade-off between performance under normal condition and failures: A potential concern about optimizing performance for failures is that good performance after failures may come at the expense of poor performance when there are no failures. To address this issue, we can bound MLU under no failures to be close to the optimal. This can be achieved by adding additional constraints, which we call penalty envelope, to the previous optimization problem: $\sum_{a,b \in V} d_{ab} r_{ab}(e)/c_e \leq MLU_{opt} \times \beta$, where MLU_{opt} is MLU under optimal routing and $\beta \geq 1$ is an operator-specified input that controls how far the normal-case performance is away from the optimal. With these constraints, we not only optimize performance under failures but also ensure acceptable performance under normal conditions. β is a tunable parameter. A small β improves the normal-case performance at the cost of degrading the performance after failures by reducing the feasible solution space over which the optimization takes place.

Trade-off between network utilization and delay: Similarly, we can use a delay penalty envelope γ to bound the average end-to-end delay under no failures for any OD pair. Note that the average delay of a link is usually dominated by the propagation delay when the link utilization is not too high. Let PD_e denote the propagation delay of link e. Then the delay penalty envelope for OD pair $a \rightarrow b$ can be achieved by adding the following constraint: $\sum_{e \in E} PD_e r_{ab}(e) \leq PD^*_{ab} \times \gamma$, where PD^*_{ab} is the smallest end-to-end propagation delay from a to b. This enforces that the average delay under R3 is no more than γ times that of the optimal delay.

4. R3 LINUX IMPLEMENTATION

To evaluate the feasibility and effectiveness of R3 in real settings, we implement R3 in Linux (kernel version 2.6.25). In this section, we describe the R3 implementation.

4.1 Overview

A key challenge in implementing R3 protection routing is its flow-based representation of $\mathtt{p}$, because current routers do not readily support such a routing scheme.

One way to address the issue is to convert a flow-based routing to a path-based routing (*e.g.*, using the flow decomposition technique [40]). A path-based routing can then be implemented using MPLS. A problem of this approach, however, is that after each failure the protection routing should be rescaled and the rescaled protection routing may decompose to new sets of paths, which have to be signaled and setup.

To address this problem, we design a more efficient implementation. We choose MPLS as the base mechanism since it is widely supported by all major router vendors. We implement a flow-based routing using MPLS, called MPLS-ff. MPLS-ff involves a simple modification to MPLS and can be easily implemented by router vendors. For wider interoperability, R3 can also be implemented using traditional MPLS, but with larger overhead.

4.2 MPLS-ff

Forwarding data structure: We use Linux MPLS to illustrate our implementation. When an MPLS packet with label l arrives at a router, the router looks up the label l in a table named incoming label mapping (ILM), which maps the label to a forward (FWD) instruction. The FWD contains a next-hop label forwarding entry (NHLFE), which specifies the outgoing interface for packets with the incoming label.

MPLS-ff extends MPLS forwarding information base (FIB) data structure to allow multiple NHLFE entries in a FWD instruction. Furthermore, each NHLFE has a *next-hop splitting ratio*. Thus, after looking up the label of an incoming packet in ILM, the router selects one of the NHLFE entries contained in the FWD according to the splitting ratios.

Implementing next-hop splitting ratios: Consider the implementation of the protection routing for link (a, b). Let l_{ab} be the label representing (a, b) at router i. For all traffic with label l_{ab}, router i should split the traffic so that the fraction of traffic to neighbor j is $\frac{p_{ab}(i,j)}{\sum_{j',(i,j')\in E,(i,j')\neq(a,b)} p_{ab}(i,j')}$.

One straightforward approach of implementing splitting is random splitting. However, this may cause packets of the same TCP flow to follow different routes, which will generate out-of-order packets and degrade TCP performance. To avoid unnecessary packet reordering, packets belonging to the same TCP flow should be routed consistently. This is achieved using hashing. The hash function should satisfy two requirements:

- The hash of the packets belonging to the same flow should be equal at the same router.
- The hash of a flow at different routers should be independent of each other (*i.e.*, the input to the hash should include router ID in addition to flow identification fields). If the hash value is only determined by the flow, the probability distribution of the hash values might be "skewed" on some routers. For example, for flow ab, if router i only forwards the packets with hash values between 40 and 64 to router j, then router j may never see packets in flow ab with hash values less than 40.

To meet these two requirements, we use a hash function that takes as input both the flow fields in the packet header (Source IP Address, Destination IP Address, Source Port, Destination Port) and a 96-bit router-dependent private number based on router ID. The output of the hash function is a 6-bit integer. To further improve the granularity of splitting, additional techniques, such as FLARE [19], could also be used.

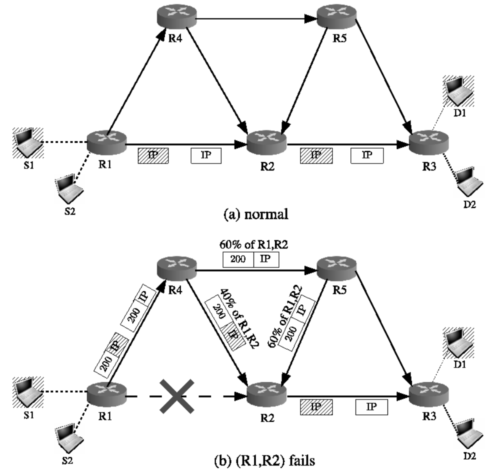
4.3 Routing Reconfiguration with MPLS-ff and Label Stacking

With MPLS-ff support, we implement resilient routing reconfiguration. In our implementation, a central server performs precomputation of protection routing p, establishes labels for each protected link, signals of MPLS-ff setup, and distributes p. The central server can be integrated with Routing Control Platform [11] or Path Computation Element (PCE) [10]. Online reconfiguration is distributed, and conducted by each router locally. It has three components: failure detection and notification, failure response, and protection routing update. Below we go over each component.

Failure detection and notification: We detect link failure using layer 2 interface monitoring. Upon a local failure event, a notification is generated and flooded to all other routers in the network through ICMP packets with type 42. In operational networks, failure detection and notification can be made more efficient using the deployed network management infrastructure. For example, SRLG failure can be detected by a risk modeling algorithm based on network monitoring [25]. The detection could be conservative (*e.g.*, if any link in a SRLG is down, assume all links in the SRLG are down). Also, the operator can issue preparation notifications to all routers before starting a MLG maintenance operation.

Failure response: After a failure is detected, MPLS-ff for the detected failure is activated by label stacking.

Figure 2 is a simple example illustrating the failure response. An IP packet of flow (S1,D1) reaches router R1. R1 looks up the packet using the base forwarding table and decides that the next-hop for this packet is R2. Normally, the packet follows the base routing and is sent to R2.



(a) normal

(b) (R1,R2) fails

Figure 2: Failure response example: (a) normal - R1 routes flows to R3 through R2; (b) link (R1,R2) fails - R4 and R5 carrying protection traffic by label stacking.

If link (R1,R2) fails, R1 activates the protection routing for (R1,R2), looks up the protection label 200 for link (R1,R2) in ILM, and pushes label 200 onto the MPLS stack of the packet. The lookup in ILM indicates that the next-hop neighbor is R4, so R1 forwards the packet to R4. When the packet reaches router R4, R4 looks up the ILM for the incoming label 200. For the protection label 200, R4 has two NHLFEs: 40% of the flows to R2, and 60% to R5. For simplicity, we assume that the outgoing labels of the two NHLFEs are still 200; in practice the signaling may assign different labels. Assume that the hash of flow (S1,D1) on R4 selects R2, then R4 forwards the packet to R2. Similarly, protection traffic for flow (S2,D2) through R4 can be carried by R5. At R2, the protection label of the packet will be popped. The packet will be forwarded to R3 following the remaining base routing of OD pair (R1,R3). When the network recovers from a failure event, the base routing is immediately re-activated and the protection routing is disabled.

Protection routing update: After a failure, each router needs to update the protection routing (*i.e.*, reconfiguring next-hop splitting ratios) for other protected links. To facilitate local update, each router stores p in its RIB (routing information base). The storage requirement is $O(|E|^2)$. Considering that backbone routers already maintain the network topology information (*e.g.*, in Link State Database), this additional storage overhead is acceptable.

Due to the order independence of rescaling, when multiple failures happen, different routers can perform rescaling on their local copies of p. When all routers are notified of all failures, the routers will have a consistent protection routing p. During the transition process, different routers may have inconsistent p, which may lead to transient loops. If transient loops are of concern, techniques such as failure-carry packets (FCP) [26] can be integrated with R3.

5. EVALUATIONS

We now evaluate R3 using both real experiments and extensive simulations based on realistic network topologies and traffic traces.

5.1 Evaluation Methodology

Network topology: For simulations, we use the PoP-level topology of a large tier-1 ISP network, called US-ISP. In addition, we use PoP-level topologies of three IP networks, Level-3, SBC, and UUNet (2003), as inferred by Rocketfuel [35]. We recursively merge the leaf nodes of the topologies with their parents until no nodes have degree one so that we have the backbone of the networks. We use OC192 as the capacity for links in the Rocketfuel topologies. We also generate a large backbone topology using GT-ITM. For our experimental results, we create the Abilene backbone topology (2006) on Emulab [41]. We scale down the link capacities

Network	Aggregation level	# Nodes	# D-Links
Abilene	router-level	11	28
Level-3	PoP-level	17	72
SBC	PoP-level	19	70
UUNet	PoP-level	47	336
Generated	router-level	100	460
US-ISP	PoP-level	-	-

Table 1: Summary of network topologies used.

to be 100 Mbps, and configure link delays to be measured values. Table 1 summarizes the used topologies. The data for US-ISP are not shown due to privacy concerns.

Traffic: We obtained real hourly traffic demand matrices of US-ISP for a one-week period. For Rocketfuel topologies, we use the gravity model [45] described in [30] to generate synthetic traffic demands. To generate realistic traffic during our experiments on Emulab, we extract Abilene traffic matrix from measurement data and scale down the values. Then we generate traffic for each OD pair at the rate encoded in the traffic matrix. We use CAIDA Anonymized 2008 Internet traces [7] for real-time IP packet generation.

Failure scenarios: To evaluate the performance under failures, we enumerate all possible single- and two-link failures, and randomly sample around 1100 scenarios of three- and four-link failures. We use random sampling for three- and four-link failures due to the large number of all possible such failures. This sampling is only needed for quantifying the performance under failures and not required for computing protection routing, since R3 does not require enumeration of failure scenarios. In addition, for US-ISP, we obtain real maintenance link groups (*i.e.*, the sets of links that were under maintenance together) for a 6-month period, and treat each maintenance link group as a single failure event.

Performance metrics: For simulation results, we use two performance metrics in our simulations: (1) bottleneck traffic intensity, and (2) performance ratio. Bottleneck traffic intensity measures network congestion. The performance ratio of an algorithm is defined as the ratio between the bottleneck traffic intensity of the algorithm and that of optimal flow-based routing, under the same network topology and traffic demand, and measures how far the algorithm is from being optimal under the given network topology and traffic demand. It is always no less than 1, and a higher value indicates that the performance of the algorithm is farther away from the optimal. We further evaluate the router storage overhead and the efficiency of resilient routing reconfiguration using measurement data using Emulab experiments.

Algorithms: We consider the following base routing schemes:

- *OSPF*: This is widely used in IP/MPLS networks for traffic engineering. For US-ISP, we use the IGP weight optimization technique in [13] and compute a set of optimized weights for each day during the evaluation period based on the 24 traffic demand matrices of that day.
- *MPLS-ff*: The base routing is computed using the algorithms in Section 3.

We consider the following protection algorithms.

- *CSPF-detour*: This algorithm is widely used in fast rerouting. The bypass routing for a set of failed links is computed using OSPF algorithm with the failed links removed. The implementation of the bypass routing is generally based on standard MPLS.
- *OSPF reconvergence (recon)*: In this algorithm, the OSPF routing protocol is allowed to re-compute routing for every changed topology.
- *Failure-Carrying Packet (FCP)*: This is the algorithm as described in [26]. In this algorithm, individual data packet keeps track of topology changes that have been encountered by the packet, and the packet is routed along the OSPF path in the current snapshot of topology.
- *Path Splicing (PathSplice)*: This algorithm is proposed in [29]. In our evaluation, we compute $k = 10$ slices with $a = 0, b = 3$ and $Weight(a, b, i, j) = (degree(i) + degree(j))/degree_{max}$, where $degree_{max}$ is the maximal node degree of the network. When forwarding traffic, if a router detects that the outgoing link for a destination is unavailable, it detours traffic to this destination through other connected slices using uniform splitting.
- *R3*: The protection routing is computed using the algorithms in Section 3.
- *Flow-based optimal link detour routing (opt)*: This is the optimal link detour routing for each given traffic and failure scenario. Specifically, for each failure scenario f, this scheme computes an optimal protection plan (*i.e.*, a rerouting for each link in f). Since the detour routing varies according to each failure scenario, it is challenging to implement in a practical way. Its performance is used to bound the best performance that can be possibly achieved by any practical protection algorithms.

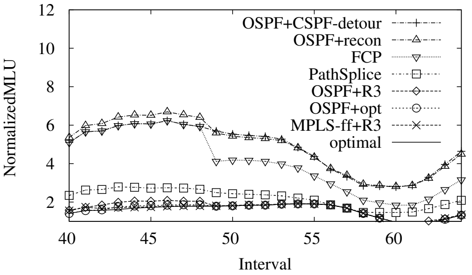


Figure 3: Time series of worst-case normalized traffic intensity with one failure during a given day for US-ISP.

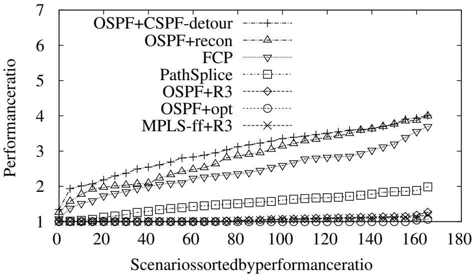
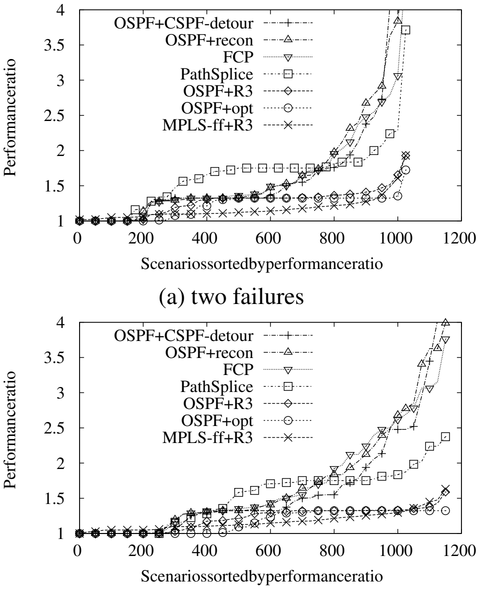


Figure 4: Summary of one failure: US-ISP.

5.2 Simulation Results

US-ISP: To preserve confidentiality of US-ISP, we do not report the absolute traffic intensity on the bottleneck link. Instead, we report normalized bottleneck traffic intensity. Specifically, for each interval in the trace, we compute the bottleneck traffic intensity using optimal flow-based routing when there is no failure. We then normalize the traffic intensity during different intervals by the highest bottleneck traffic intensity observed in the trace.

Single failure: We first introduce one failure event (SRLG or MLG). At each interval, we assume that the network topology deviates from the base topology by only one failure event. We identify the worst case performance upon all possible single failure events, and report normalized traffic intensity on the bottleneck link. Figure 3 shows the results. For clarity, we zoom in to a one-day time frame during the evaluation period; thus, there are 24 intervals. We make the following observations. First, R3 based protection (MPLS-ff+R3 and OSPF+R3) performs close to the optimal, and achieves performance similar to flow-based optimal link detour routing on top of OSPF (OSPF+opt). However, flow-based optimal link detour (opt) requires the computation of optimal protection routing for each individual topology-change scenario, whereas R3 achieves similar performance with only a single protection routing and a simple, light-weight routing reconfiguration. Second, comparing the two R3 schemes, we observe that MPLS-ff+R3 performs better than OSPF+R3 (see intervals 40 to 48). This is expected since OSPF is less flexible than MPLS. Third, without a good protection



(a) two failures

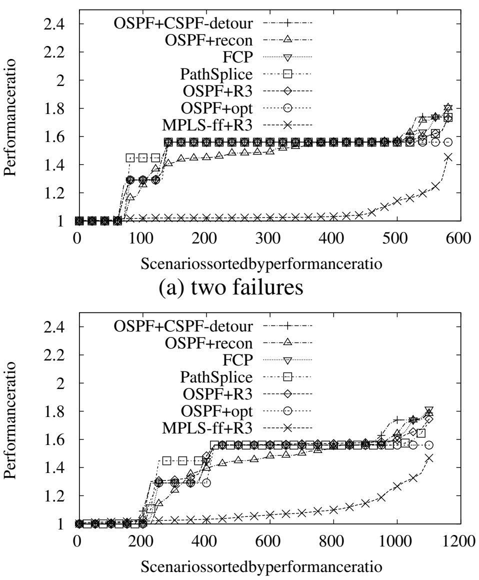
(b) sampled three failures

Figure 5: Sorted performance ratio under multiple failures during peak hour: US-ISP.

scheme, OSPF+recon, OSPF+CSPF-detour, and FCP all lead to higher levels of normalized traffic intensity. In the early part of the day, their traffic intensity can be as high as 3 times that of the other routing protection schemes (∼5 vs. ∼1.5). Fourth, starting from interval number 49, FCP starts to have better performance than OSPF+recon, OSPF+CSPF-detour. But its traffic intensity in the later part of the day can still be as high as 2 times (*e.g.*, during interval number 60) that of MPLS-ff+R3, OSPF+R3 and OSPF+opt. Finally, by rerouting traffic to multiple slices in a "best effort" fashion, PathSplice leads to less congestion and achieves much better performance than other existing protection algorithms, though it is still less efficient than R3 based algorithms.

The previous evaluation shows the effectiveness of R3 during one day. Next, we summarize the overall performance during the entire evaluation period (which lasts seven days). Figure 4 shows the performance ratio versus the time interval sorted based on the performance ratio. We make the following observations. First, MPLS-ff+R3, OSPF+R3, and OSPF+opt consistently perform within 30% of the optimal throughout the entire evaluation period. Second, OSPF+recon, OSPF+CSPF-detour, PathSplice, and FCP all cause significant performance penalty. The performance of OSPF+recon, OSPF+CSPF-detour, and FCP can be 260% higher than optimal. PathSplice performs better, but it still can be 100% higher than the optimal while R3 based schemes are within 30%. Thus, the traffic intensity of PathSplicing can be 54% higher than R3.

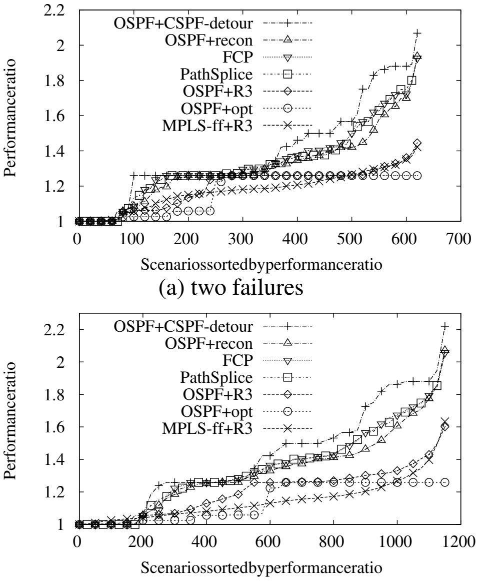
Multiple failure events: Next we evaluate using multiple failure events in US-ISP. For clarity of presentation, we fix the interval (a peak hour) and evaluate the failure events. We report results for two failures and sampled three failures. We report only sampled three failures because there are too many failure scenarios to enumerate; thus, we use random sampling. Figure 5 shows the performance ratio versus the scenario sorted based on the performance ratio. To make it easier to read, we have truncated the y-axis of Figure 5 at the value of 4. We observe that under two and three failures, MPLS-ff+R3 and OSPF+R3 continue to significantly outperform OSPF+recon, OSPF+CSPF-detour, FCP, and PathSplice. From Figure 5(a), we observe that OSPF+recon, OSPF+CSPF-detour, FCP and PathSplice can cause bottleneck traffic intensity to be more than 3.7 times of the optimal for two failures. This is 94% higher than the highest of MPLS-ff+R3 and OSPF+R3 (they reach around 1.9). For three failures, OSPF+recon, OSPF+CSPF-detour, FCP, and PathSplice reach at least 2.4 times of optimal, while MPLS-ff+R3 and OSPF+R3 reach only 1.6; thus they are at least 50% higher than R3 based protection.



(a) two failures

(b) sampled three failures

Figure 6: Sorted performance ratio: SBC.



(a) two failures

(b) sampled three failures

Figure 7: Sorted performance ratio: Level 3.

Summary: For US-ISP, R3 based schemes consistently achieve better performance than OSPF+recon, OSPF+CSPF-detour, FCP, and PathSplice, outperforming them by at least 50% in all scenarios and much higher in some scenarios.

Rocketfuel topologies: Next we evaluate using Rocketfuel topologies. For each Rocketfuel topology, we randomly generate one traffic matrix using gravity model. Due to lack of SRLG information, we generate all two-link failures and randomly sample around 1100 three-link failures. We then evaluate the performance of different algorithms under these failures.

Figure 6 and Figure 7 show the performance ratios for SBC and Level 3, respectively. We choose these two topologies as they give representative results among the Rocketfuel topologies. From these figures, we make the following observations. First, on SBC, MPLS-ff+R3, which jointly optimizes base routing and protection routing, significantly out-performs all OSPF based algorithms, including OSPF+opt. This demonstrates the advantage of joint optimization of base routing and protection routing. Second, on Level 3, MPLS-ff+R3 and OSPF+R3 have very similar performance, and consistently out-perform other OSPF based algorithms, except for OSPF+opt. In fact, on Level 3, OSPF+opt performs very close to optimal and slightly better than MPLS-ff+R3 and OSPF+R3. Recall that it is substantially more expensive to implement OSPF+opt, this indicates that on networks with very good OSPF routing, R3 on top of OSPF can be used to achieve most of the gains of R3 while retaining the simplicity of OSPF routing.

Prioritized R3: We evaluate prioritized R3 using three classes of traffic with different priorities. Specifically, we extract traffic of TPRT and TPP from the US-ISP backbone traffic in a peak interval. For confidentiality, we rescale the traffic volumes of TPRT and TPP. We then subtract these two types of traffic from the total traffic and treat the remaining traffic as IP. For prioritized R3, we set the protection levels of TPRT, TPP, and IP to four failures, two failures, and one failure, respectively. For general R3, all traffic is protected against one failure. We report results for all single failures, top 100 worst-case two-failure scenarios, and top 100 worst-case four-failure scenarios out of the sampled four failures.

Figure 8 shows the normalized bottleneck traffic intensities for the three classes of traffic under R3 with and without priority. We make the following observations. First, both prioritized and general R3 provide congestion-free rerouting under single failures. Comparing the performance between prioritized and general R3, we observe that IP traffic has lower bottleneck traffic intensity under prioritized R3 than under general R3, while the bottleneck traffic intensities of TPP and TPRT under prioritized R3 are slightly higher than under general R3. The reason for the latter is because even though IP traffic has lower priority than TPP and TPRT under multiple failures, prioritized R3 can give IP better treatment under single failures as long as TPP and TPRT traffic are well protected, which is the case (*i.e.*, the bottleneck traffic intensities of TPP and TPRT are always smaller than 0.4 under single failures). Second, under two-link failures, prioritized R3 guarantees no congestion for TPRT and TPP, whereas TPRT and TPP experience congestion under general R3. The bottleneck traffic intensities of IP traffic are higher under prioritized R3 than under general R3, which is inevitable due to limit of resources. Third, under four-link failures, TPRT incurs no congestion using prioritized R3, whereas all traffic experiences congestion using general R3. Even TPP, which is protected up to two-link failures, achieves lower traffic intensities under prioritized R3 than under general R3. As expected, IP traffic experiences congestion under both general and prioritized R3 during four-link failures. These results demonstrate that prioritized R3 is effective in providing differentiation to different classes of traffic.

Penalty envelope: Our R3 formulation introduces a penalty envelope on normal case MLU. The goal is to balance between being robust to topology changes and being optimal when there are no topology changes. We demonstrate the importance of this technique by evaluating network performance under no topology changes. In Figure 9, we show the performance of four algorithms: R3 without penalty envelope, OSPF, R3 with penalty envelope, and optimal. We pick a time period when OSPF performs particularly well with optimized IGP weights. We make the following observations. First, adding the penalty envelope significantly improves normal case performance. The 10% penalty envelope is effective and R3 performs within the envelope during normal operations. Second, R3 without penalty envelope can lead to significant performance penalty in normal cases. Its normalized traffic intensity sometimes goes as high as 200% of the optimal and may perform even worse than OSPF. The reason is that R3 without penalty envelope optimizes exclusively for the performance under failures and only enforces no congestion during normal network topology and traffic.

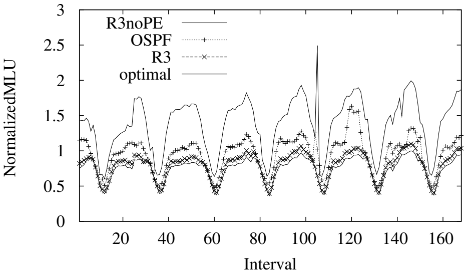
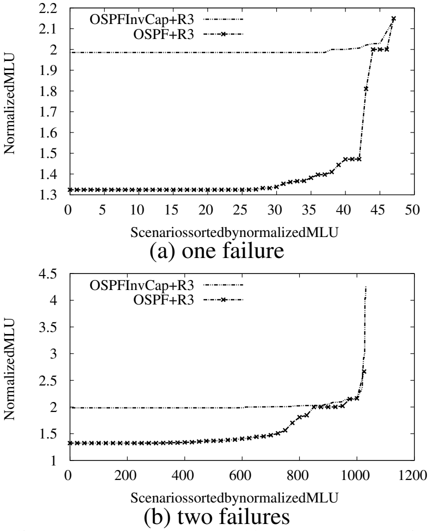


Figure 9: Time series of normalized traffic intensity with no change during a week for US-ISP.



(a) one failure

(b) two failures

Figure 10: Sorted normalized traffic intensity: US-ISP during peak hour.

Robustness on base routing: The previous evaluation shows that R3, which jointly optimizes base routing and protection routing, out-performs OSPF+R3. So a better base routing leads to better overall performance. To further understand the impact of base routing, we conduct the following evaluation. We compare two versions of OSPF as base routing: (i) OSPFInvCap+R3 and (ii) OSPF+R3, where in the former the IGP weights of the base routing is inverse proportional to link capacity and in the latter IGP weights are optimized. As shown in Figure 10, R3 based on OSPFInvCap is significantly worse than R3 based on an optimized OSPF routing. These results further demonstrates the importance of base routing.

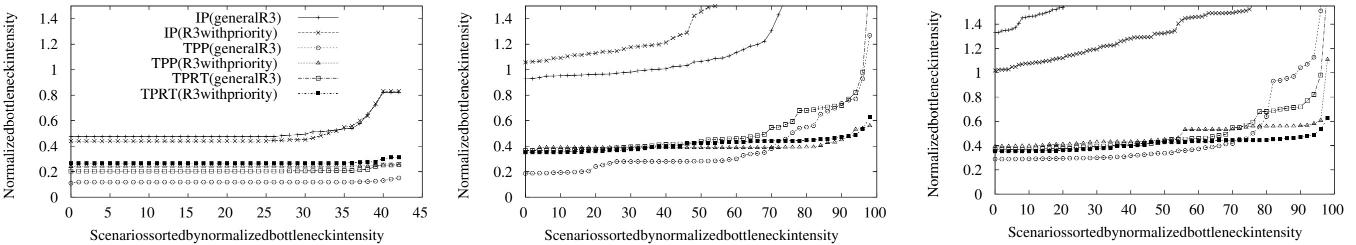
5.3 Implementation Results

We implement R3 on Linux and evaluate its efficiency.

Offline computation complexity: We first evaluate the computation complexity of R3. We run R3 offline precomputation for the 6 topologies with different failure guarantees. All computation is done using a single Linux machine with commodity hardware configuration (2.33 GHz CPU, 4 GB memory). We employ ILOG CPLEX 10.0 [8] as the linear program solver. Table 2 summarizes the results. We observe that the precomputation time fluctuates with the number of failures and is typically below half an hour. This is because the complexity of the linear program (7) is independent of the number of failures. In contrast, explicit enumeration of failure scenarios can quickly become prohibitive as the number of failures increases.

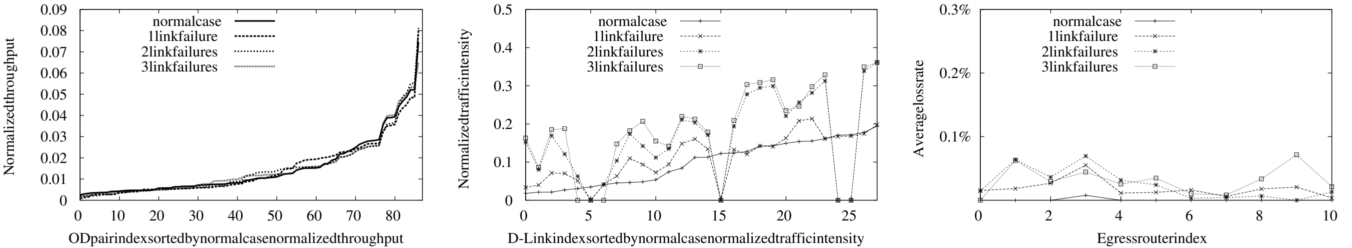
Storage and MPLS overhead: One concern about R3 protection implementation based on MPLS-ff is router storage overhead (*i.e.*, FIB and RIB size), given that routers need to maintain the protection labels for all protected links and store local copies of the protection routing p.

To evaluate the storage overhead, for a given topology, we run R3 MPLS-ff protection assuming that all backbone links are protected



(a) 1-link failures (b) worst-case scenarios of 2-link failures (c) worst-case scenarios of 4-link failures

Figure 8: Sorted bottleneck traffic intensity for prioritized traffic: US-ISP (note that the legend is the same for all three plots).



(a) normalized throughput (b) normalized link load (c) aggregated loss rate at egress routers

Figure 11: Network performance using R3 under multiple link failures.

Network/# failures	1	2	3	4	5	6
Abilene	0.3	0.30	0.30	0.32	0.33	0.29
Level-3	1.80	1.97	2.56	2.71	2.46	2.43
SBC	1.46	1.76	1.75	1.76	1.92	1.91
UUNet	1010	572	1067	810	864	720
Generated	1388	929	1971	2001	1675	2131
US-ISP	21.3	21.9	21.4	20.1	22.1	21.8

Table 2: R3 offline precomputation time (seconds).

Network	# ILM	# NHLFE	FIB memory	RIB storage
Abilene	28	71	<9 KB	<83 KB
Level-3	72	304	<36 KB	<535 KB
SBC	70	257	<31 KB	<503 KB
UUNet	336	2402	<267 KB	<11 MB
Generated	460	2116	<251 KB	<20 MB
US-ISP	-	-	<39 KB	<656 KB

Table 3: Router storage overhead of R3 implementation.

except the stub links which cannot be bypassed. We measure the ILM table size, the NHLFE table size, the FIB size, and the RIB size per router. Table 3 summarizes the results for 6 topologies. We observe that all of these 6 network topologies can be protected by R3 with modest FIBs (<267 KB) and RIBs (< 20 MB).

A related overhead is MPLS labels. Recall that the number of MPLS labels used by MPLS-ff for protection routing is bounded by the number of links in the network. Since many routers can support at least tens of thousands of MPLS labels, the number of MPLS labels used in protection routing is not an issue.

Effective resilient routing reconfiguration: Next, we evaluate the effectiveness of protection routing. We generate failure scenarios by disconnecting three links (Houston-Kansans, Chicago-Indianapolis, Sunnyvale-Denver) sequentially on the emulated Abilene topology (each link is two directed links). After failing one link, we delay by about one-minute before failing the next link. During the evaluation, bursty traffic is generated to allow us to measure the traffic throughput between every OD pair, the traffic intensity on each link, and the aggregated loss rate at each egress router (the traffic matrix encodes the expected outgoing traffic).

As shown in Figure 11, our R3 implementation successfully reroutes traffic without overloading any link. From Figure 11(b), we see that despite three failed links, the bottleneck traffic intensity is always within 0.37. Figure 12 further plots the real-time RTT of a flow between Denver and Los Angeles during our test process. We can clearly identify the three-step increases of RTT, due to the three link failures. We observe that our R3 protection routing implementation achieves smooth and efficient routing protection.

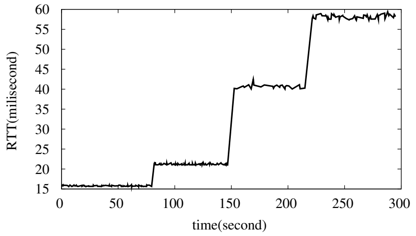


Figure 12: RTT of a flow (Denver-Los Angeles).

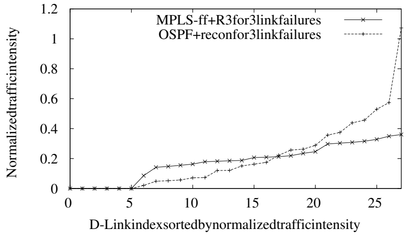


Figure 13: Sorted normalized traffic intensity on each directed link under three link failures: MPLS-ff+R3 vs OSPF+recon.

To appreciate the effectiveness of R3, we run the same failure scenario using OSPF reconvergence protection. Figure 13 compares the traffic intensity by OSPF+recon vs MPLS-ff+R3. Using OSPF, the traffic intensity on the link between Washington and Atlanta (link index 28) reaches as high as 1.07 (instantaneous rate). Due to the congestion, we observe from the trace that the throughput for the OD pair New York City to Indianapolis drop by up to 32.6% using OSPF+recon.

6. RELATED WORK

The existing work can be classified into two categories: (i) routing under failures and (ii) routing under variable traffic.

Routing under failures: Many recent studies focus on minimizing the duration of disruption due to failures (*e.g.*, [4, 20, 21, 23, 24, 26, 27, 29, 32]). These techniques precompute protection and quickly reroute traffic upon detecting failures (and before routing convergence) [33]. However, they do not provide performance predictability or avoid congestion. As we have seen, they may lead to serious congestion and thus violation of service level agreements.

Meanwhile there are also significant studies on optimizing performance under failures. In [14], the authors studied optimization of OSPF/IS-IS weights under failures. However, it is a heuristics based approach and does not provide performance guarantee or avoidance of congestion. In MATE [9] and TeXCP [18], the authors study how to react to instantaneous traffic load and redistribute traffic on alternate links or paths. Many previous studies achieve optimal performance by re-optimizing routing after each failure (*e.g.*, MPLS routing [39]). A major advantage of these approaches is that the new routing is computed specifically for the new topology. Thus, the new routing can efficiently utilize the remaining network resources and provide certain guarantees (*e.g.*, how close the rerouting response compared with the optimal [2]). A drawback of these approaches, however, is their slow response time. Re-optimization from scratch for the new topology can be computationally expensive. In addition, the new routing could be very different from the existing one and thus take substantial delay in installation and convergence. This can cause significant service disruption because of operation errors, forwarding loops and packet loss during long convergence process. As a result, network operators are highly reluctant to completely change their routing. Instead, they prefer simple routing reconfiguration. They completely re-optimize only periodically or after a major change, instead of after each topology failure. The only work that optimizes routing simultaneously for different topologies is [2], but it requires enumeration of all possible topologies after failures and faces scalability issues under multiple failures.

Routing under variable traffic demand: High variability in Internet traffic has motivated researchers to design robust traffic engineering that works well under variable traffic. One class of algorithms [1, 9, 18, 31, 43] maintains a history of observed traffic demand matrices, and optimizes for the representative traffic demand matrices. Another class of algorithms is oblivious routing [2, 3, 22, 37, 46], which optimizes the worst-case performance over all possible traffic demands. More recently, Wang *et al.* [40] further combined oblivious routing with prediction-based optimization to provide good performance under typical demands while guaranteeing the worst-case performance. These works focus on traffic variability and do not consider topology variability.

7. CONCLUSIONS

In this paper, we propose Resilient Routing Reconfiguration (R3) to find a single protection routing that can be effectively reconfigured to provide congestion-free guarantee under multiple failures. We introduce a novel compact representation of a large number of failure scenarios, and compute a protection scheme that is resilient to both link failures and traffic variability. We further extend R3 to handle realistic failure scenarios, prioritized traffic, and the tradeoff between performance and resilience. We fully implement R3 on Linux using MPLS-ff, and demonstrate its effectiveness through real experiments and extensive simulations using realistic network topologies and traffic.

Acknowledgments: The research is supported in part by NSF Grants CNS-0546720, CNS-0546755, CNS-0626878, and CNS-0627020. We are grateful to Jia Wang, Murali Kodialam, anonymous reviewers, C. Tian, and Dave Wang for valuable comments.

8. REFERENCES

[1] S. Agarwal, A. Nucci, and S. Bhattacharyya. Measuring the shared fate of IGP engineering and interdomain traffic. In *Proc. ICNP*, Nov. 2005.

[2] D. Applegate, L. Breslau, and E. Cohen. Coping with network failures: Routing strategies for optimal demand oblivious restoration. In *Proc. ACM SIGMETRICS*, June 2004.

[3] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proc. ACM SIGCOMM*, Aug. 2003.

[4] A. Atlas and Z. Zinin. Basic specification for IP Fast-Reroute: loop-free alternates. (IETF Internet-Draft), draft-ietf-rtgwg-ipfrr-spec-base-10.txt, 2007.

[5] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[6] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1992.

[7] CAIDA. `http://www.caida.org/tools/`.

[8] ILOG CPLEX: optimization software. http://www.ilog.com/products/cplex/.

[9] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proc. IEEE INFOCOM*, Apr. 2001.

[10] A. Farrel, J.-P. Vasseur, and J. Ash. *A Path Computation Element (PCE)-based Architecture, RFC 4655*, Aug. 2006.

[11] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. van der Merwe. The case for separating routing from routers. In *Proc. ACM SIGCOMM FDNA Workshop*, Sept. 2004.

[12] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communication Magazine*, Oct. 2002.

[13] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. IEEE INFOCOM*, Mar. 2000.

[14] B. Fortz and M. Thorup. Robust optimization of OSPF/IS-IS weights. In *Proc. INOC*, Oct. 2003.

[15] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving sub-second IGP convergence in large IP networks. *ACM CCR*, 35(3), 2005.

[16] G. Iannaccone, C. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier-1 backbone. *IEEE Network Magazine*, 18(2):13–19, 2004.

[17] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot. An approach to alleviate link overload as observed on an IP backbone. In *Proc. IEEE INFOCOM*, Apr. 2003.

[18] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, Aug. 2005.

[19] S. Kandula, D. Katabi, S. Sinha, and A. Berger. Dynamic load balancing without packet reordering. *SIGCOMM CCR*, 37(2), 2007.

[20] K. Kar, M. S. Kodialam, and T. V. Lakshman. Routing restorable bandwidth guaranteed connections using maximum 2-route flows. *IEEE/ACM Transactions on Networking*, 11(5):772–781, 2003.

[21] M. Kodialam and T. V. Lakshman. Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information. In *Proc. IEEE INFOCOM*, Apr. 2001.

[22] M. Kodialam, T. V. Lakshman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *Proc. HotNets-III*, Nov. 2004.

[23] M. Kodialam, T. V. Lakshman, and S. Sengupta. A simple traffic independent scheme for enabling restoration oblivious routing of resilient connections. In *Proc. IEEE INFOCOM*, Apr. 2004.

[24] M. S. Kodialam and T. V. Lakshman. Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information. *IEEE/ACM Transactions on Networking*, 11(3):399–410, 2003.

[25] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. IP fault localization via risk modeling. In *Proc. NSDI*, 2005.

[26] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving convergence-free routing using failure-carrying packets. In *Proc. ACM SIGCOMM*, Aug. 2007.

[27] A. Li, P. Francois, and X. Yang. On improving the efficiency and manageability of NotVia. In *Proc. CoNEXT*, Dec. 2007.

[28] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of failures in an IP backbone network. In *Proc. IEEE INFOCOM*, Apr. 2004.

[29] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path splicing. In *Proc. ACM SIGCOMM*, 2008.

[30] M. Roughan. First order characterization of Internet traffic matrices. In *Proc. 55th Session of the International Statistics Institute*, Apr. 2005.

[31] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *Proc. IMC*, Oct. 2003.

[32] M. Shand and S. Bryant. IP fast reroute framework. (IETF Internet-Draft), draft-ietf-rtgwg-ipfrr-framework-06.txt, 2007.

[33] V. Sharma, B. M. Crane, S. Makam, K. Owens, C. Huang, F. Hellstrand, J. Weil, L. Andersson, B. Jamoussi, B. Cain, S. Civanlar, and A. Chiu. *Framework for MPLS-Based Recovery. RFC 3469*, Feb. 2003.

[34] N. So and H. Huang. Building a highly adaptive, resilient, and scalable MPLS backbone. `http://www.wandl.com/html/support/papers/VerizonBusiness_WANDL_MPLS2007%.pdf`, 2007.

[35] N. Spring, R. Mahajan, and D. Wetherall. Rocketfuel: An ISP topology mapping engine. Available from `http://www.cs.washington.edu/research/networking/rocketfuel/`.

[36] Telemark. Telemark survey. http://www.telemarkservices.com/, 2006.

[37] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(7):350–361, 1982.

[38] J. P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery: Protection and Restoration of Optical, SONET-SDH, and MPLS*. Morgan Kaufmann, 2004.

[39] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. COPE: Traffic engineering in dynamic networks. In *Proc. ACM SIGCOMM*, 2006.

[40] H. Wang, Y. R. Yang, P. H. Liu, J. Wang, A. Gerber, and A. Greenberg. Reliability as an interdomain service. In *Proc. ACM SIGCOMM*, Aug. 2007.

[41] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. OSDI*, Dec. 2002.

[42] Wired News. The backhoe: A real cyberthreat, Jan. 2006. `http://www.wired.com/news/technology/1,70040-0.html`.

[43] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Towsley. Optimal routing with multiple traffic matrices: Tradeoff between average case and worst case performance. In *Proc. ICNP*, Nov. 2005.

[44] Y. Zhang and Z. Ge. Finding critical traffic matrices. In *Proc. DSN '05*, 2005.

[45] Y. Zhang, M. Roughan, C. Lund, and D. L. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. ACM SIGCOMM*, Aug. 2003.

[46] R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone network. In *Proc. HotNets-III*, Nov. 2004.