FREKit: A Flexible Simulator for Probabilistic Congestion and Rerouting in Multi-Protocol Networks

Juan Vanerio*†, Pedro Casas†, Stefan Schmid*‡
*University of Vienna, †AIT Austrian Institute of Technology, ‡TU Berlin

***Abstract*—We introduce FREKit, a novel network simulator tailored for research in congestion modeling, fast rerouting, and multi-protocol environments. Existing simulators are either too slow, lack multi-protocol support, or do not offer rich congestion-based metrics and aggregations, limiting their suitability for the type of research FREKit targets. Designed around the pipe-based abstraction – assuming known traffic matrices, FREKit evaluates system-wide behaviors with elastic and inelastic congestion computation models, providing fine-grained insights over interface-level loads. FREKit supports multiple protocols – including IPv4, IPv6, MPLS, and Segment Routing – along with various path search algorithms and rapid prototyping. Its probabilistic link failure framework enables congestion-aware resilience studies and advanced congestion metrics such as value-at-risk (VaR). The simulator is implemented in Python with efficient data structures (NumPy, NetworkX) and optimized via multiprocessing and dynamic programming. Despite being in active development, FREKit is already proving valuable in research on fast rerouting and source routing, demonstrating its potential to enable sophisticated congestion-aware network studies.**

I. Introduction

As communication networks scale in size and complexity, network link failures are becoming more frequent [1]. To achieve high dependability despite these incidents, networks deploy fast rerouting (FRR) mechanisms in the data plane [2], which enable routers to rapidly redirect traffic to alternative paths when disruptions occur. Designing FRR mechanisms that deliver robust resilience remains a significant challenge and is an active area of research [2]–[5]. In this setting, routers are required to react to failures as soon as possible, with a traditional deadline of at most 50 ms. Routers are therefore required to make their decisions based on available local information only, without knowledge of downstream issues. This limitation can lead to incorrect forwarding when multiple failures occur, and risk incurring in potentially damaging forwarding loops. This is an acknowledged risk; the growth in network size and the prevalence of shared risk link groups now require operators to plan for multiple simultaneous failures [6].

There is an opportunity to advance previous work on multi-failure fast rerouting by addressing two key limitations: (1) enabling multi-protocol applicability through the use of modern Segment Routing architectures prevalent in ISP networks, and (2) incorporating the impact of link congestion, which is often overlooked in existing rerouting strategies. However, experimenting with such networks or even laboratory setups is expensive and restricted to a few nodes. Simulating large-scale computer networks with realistic protocol stacks and congestion dynamics is essential for researchers working on resilience, routing optimization, and fault tolerance. On this front, traditional tools like OMNeT++ [7] and MPLSKit [8] offer limited protocol support or become impractical to perform high-level congestion analysis over a sufficiently large number of failure scenarios.

Probabilistic bandwidth guarantees have become a central concept in WAN traffic engineering, offering a more efficient alternative to traditional over-provisioning strategies designed for worst-case failures. By allowing operators to specify acceptable risk thresholds, these approaches enhance resource utilization without compromising reliability. Systems like TeaVar [9] and Flexile [10] exemplify this shift, providing practical frameworks and metrics for risk-aware traffic engineering. TeaVaR introduces a risk-aware framework that leverages probabilistic modeling to balance availability and performance, achieving up to a three-fold improvement in throughput over traditional methods. Building on this, Flexile refines the reliability-efficiency trade-off by targeting bandwidth objectives that are met 'almost always,' offering greater flexibility while maintaining strong performance guarantees.

This paper introduces FREKit, a novel simulation framework designed to address key limitations in existing multi-protocol network simulators by offering a highly extensible and protocol-rich environment tailored to modern networking research. FREKit enables the generation and evaluation of diverse data planes and supports advanced risk-aware analysis by natively computing TeaVaR and Flexile metrics, making it especially suitable for studies focused on resilience and congestion-aware behavior in complex networks.

II. System Architecture Overview

FREKit models computer networks using a pipe-based abstraction, assuming known traffic matrices as input. The core architecture consists of: **(1)** mechanisms to load or generate topologies and demands; **(2)** routing and path computation layers with support for various protocols and custom algorithms; **(3)** an extensible protection computation framework, which supports the implementation of industry-standard mechanisms such as MPLS, RSVP, and Fast Reroute (FRR); **(4)** a probabilistic framework that efficiently identifies the most likely failure scenarios, avoiding the resource-intensive overhead of exhaustive scenario enumeration; **(5)** a modular, distributed simulation engine capable of modeling link failures and capturing the effects of dynamic rerouting; **(6)** multiple

978-3-903176-74-4 ©2025 IFIP

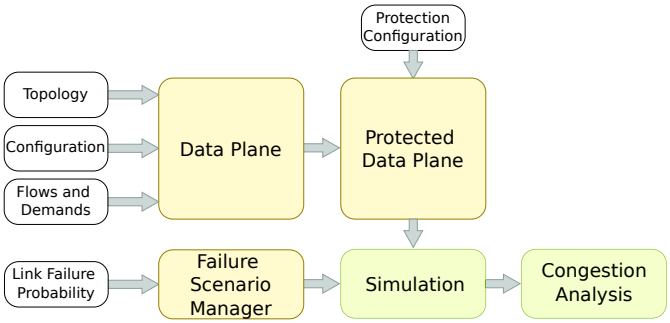

Fig. 1. FREKit architecture overview and modules.

congestion models, including elastic and inelastic congestion; and **(7)** Value-at-Risk (VaR) based metrics, which summarize experiments across multiple failure scenarios. Figure 1 illustrates the architecture and main components of FREKit.

The system is implemented in Python, using *NumPy* for fast numerical operations and *NetworkX* for graph-based topology and implementation of routing models. Parallel execution is supported via Python's multiprocessing package, and simulation state reuse is optimized with dynamic programming techniques. The following subsections detail the intended process for FREKit.

A. Network Initialization and Flow Definition

The typical workflow begins with generating or loading a weighted network topology, along with a set of Segment Routing *flows*, each defined by a source and destination router, a specified traffic demand, and potentially multiple segments. FREKit then synthesizes forwarding rules for user-defined protocols, enabling multi-protocol routing and the construction of a functional *data plane*. It supports a range of protocols, including IPv4, IPv6, MPLS, SR-MPLS, and SRv6. As a result, FREKit seamlessly manages diverse forwarding structures, such as hash tables for MPLS and trie-based lookups for IP routing. The synthetic rules are created based on k-shortest paths and may be extended to use advanced routing protocols. If a router knows multiple routes towards a destination, a *preference* attribute controls the outgoing traffic load balancing.

B. Fast Reroute Mechanisms

Next, the data plane is augmented to include FRR protection mechanisms. This is implemented by passing the data plane to a class that contains the specific methods for each protocol. The FRR entries take lower precedence in the routing tables and are only executed in case the intended outgoing link is unavailable. Furthermore, FREKit supports a hierarchy of precedence of FRR rules, which allow each router to react to multiple *local* link failures. Currently implemented protection mechanisms include MPLS RSVP FRR [11], RMPLS [12], and TI-LFA (draft 21) [5].

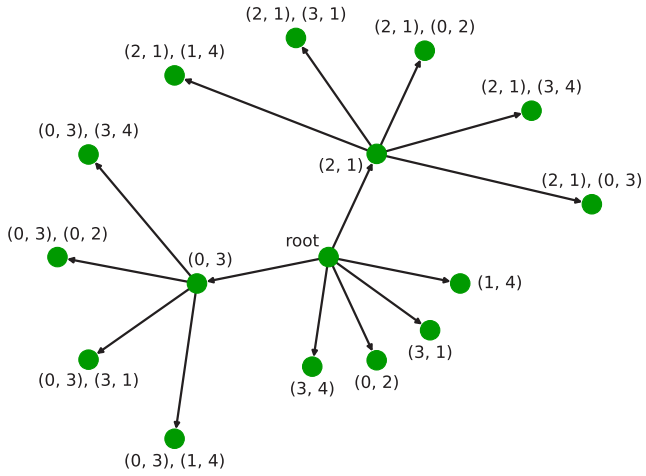

Fig. 2. Failure Scenario Manager tree structure.

C. Failure Scenario Modeling

The next stage involves configuring probabilistic failure scenarios. FREKit either loads or generates per-link failure probabilities, defaulting to the Weibull-distribution-based model proposed by TeaVaR [9]. A *Scenario Manager* is then constructed as a tree structure representing subsets of possible link failures. The root node corresponds to the failure-free scenario, and each child node incrementally introduces an additional failed link. Figure 2 illustrates the concept of the Scenario Manager.

Traversing an edge in this structure corresponds to introducing one additional link failure, which simplifies the incremental computation of scenario probabilities. This structure also enables efficient simulation: since most flows remain unaffected by a single additional failure, previously computed data can be reused. The tree expands in a breadth-first manner until the cumulative probability of its nodes exceeds a configurable threshold. This approach significantly reduces the scenario space while preserving accuracy, as committed scenarios are assumed to result in complete traffic failure.

III. Simulation Procedure

The simulation procedure in FREKit consists of multiple stages, as described in the following sections.

A. Per-flow packet forwarding simulation

In the initial stage, a failure scenario is selected. For each flow, a packet is instantiated at its source router(s) with the appropriate protocol headers. The packet's state is defined by the combination of its current header contents and router.

The simulator proceeds on each step by forwarding the packet from one router to the next, using the outgoing link and header rewriting rules mandated by the forwarding rules. On each step, the packet state is updated. Should multiple equally valid forwarding rules be found, the packet is cloned along each option to traverse all possible paths. The preference value of each option is stored on each packet. Should no rules

be valid, the router looks for lower priority – fast rerouting rules – and repeats the procedure until it finds a matching rule that it can use.

B. Packet Termination and Packet State Transition Graph

A packet forwarding procedure terminates if it reaches an intended target node, has no valid matching forwarding rule, or if its Time-To-Live reaches zero. The flow simulation ends when all of its packets have finished. When the flow simulation has finished, the trajectory of each packet is used to compute the graph of all the packet state transitions (PSTG). The multiplicatively-aggregated preference observed in the state transition is registered on the respective edge of the PSTG. To illustrate this, consider a packet that traverses two routers that load balance their forwarding rules with preferences 0.6 and 0.2 before arriving at the target node. That packet starts with a load value of 1.0, then 0.6, and finally 0.12.

C. Backpropagation and Load Function Generation

The PSTG of the flow is used to create a function that computes the traffic load of each traversed link up to the target node. This is done by backpropagating the load from the last transition before the target node. The loads are multiplied by the preferences when traffic is split, and added when they meet in the same transition. The load induced by each transition is expressed by the recursive relationship of Eq.1 and illustrated in Figure 3, which represents the PSTG and the load function generation basic computations. For a topology of l links, we define $\mathbf{p} \in [0,1]^l$ as the loss fraction experienced by each link. Let $L(y,z)$ be the induced load by the transition from packet state y to packet state z. Then, for a given state x with incoming transitions from states $\{a_1,\ldots,a_n\}$ and outgoing states $\{b_1,\ldots,b_m\}$, Eq.1 represents the load in the transition from x to b_j for some $j \in [1,m]$, where w_k and p_k for $k \in [1,m]$ are the outgoing link preference and packet loss, respectively.

$$L(x,b_j) = \underbrace{\left(\sum_{i=1}^{n} L(a_i,x)\right)}_{\substack{\text{incoming load}\\ \text{at } x}} \underbrace{\left(\frac{w_j}{\sum_{k=1}^{m} w_k}\right)}_{\substack{\text{normalized}\\ \text{outgoing}\\ \text{preference}}} \underbrace{(1-p_j)}_{\substack{\text{link}\\ \text{delivery rate}}} \quad (1)$$

To avoid recomputing the backpropagation stage, traffic load functions are cached and held as valid while the set of failed and traversed links remains intact.

IV. Congestion Computation

Since each packet transition corresponds to a specific topology link, link load can be directly computed from these transitions. In the final stage, once all flow-specific PSTGs are generated, their link load functions – scaled by actual flow demands – are aggregated to determine both absolute and relative link utilization. FREKit supports three distinct congestion computation models:

Overshoot – the largest relative utilization ratio over all links, even if it exceeds 1.0.

Inelastic – iterates over possible values of the link packet loss vector $\mathbf{p}$, following a fixed point algorithm. The rationale is to introduce sequential link losses until no utilization goes above its link bandwidth. Represents the traffic losses that follow in a short time span, before upper-layer congestion control mechanisms can kick in.

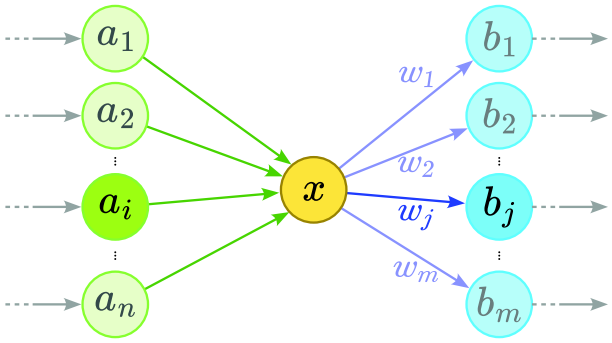

Fig. 3. Packet state transition graph and load computation function.

Elastic – iteratively throttles down or up the demand of each flow until no link is congested. Approximates the traffic losses that result after the sources and destinations are able to scale down traffic to fit.

In the elastic and inelastic congestion models, the loss for each flow under each failure scenario is defined as the fraction of traffic that fails to reach its destination relative to the original demand, resulting in a loss matrix of size {number of flows $\times$ number of scenarios} for each data plane. The simulation progresses through failure scenarios efficiently by reusing cached link load functions whenever their preconditions remain valid. To accelerate computation, simulations across different scenarios are parallelized across persistent worker processes, balancing caching benefits with multicore performance. Once the loss matrices are computed, FREKit derives Value-at-Risk-based metrics, including TeaVaR and Flexible style metrics, summarizing the performance of the specific protection mechanism under evaluation.

V. Congestion-aware Fast-ReRoute Research

We are using FREKit to develop and evaluate congestion-aware FRR mechanisms across multiple protocol stacks using Segment Routing. These mechanisms aim to make local decisions during link failures while mitigating post-failure congestion, employing SR-MPLS or SRv6 to pre-install alternate forwarding entries for rapid activation upon failure detection.

FREKit quantifies the impact of traffic redistribution following rerouting, by simulating probabilistic failure scenarios and evaluating per-scenario congestion using both elastic and inelastic models. This approach enables the estimation of the performance of both traditional and novel fast reroute mechanisms, whether they account for congestion or operate in a congestion-agnostic manner. By computing Value-at-Risk metrics, FREKit identifies the protection schemes that minimize traffic loss and transient overloads. It highlights cases where traditional FRR strategies cause severe link oversubscription,

leading to the design of multipath-aware protection strategies that spread rerouted traffic across disjoint backup segments.

ACKNOWLEDGMENT

This work has been supported by the FWF Austrian Science Fund, Project reference I-6653, as part of the EU CHIST-ERA-2022-SPiDDS-02 project *GRAPHS4SEC*.

REFERENCES

[1] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41 (4), 2011, pp. 350–361.

[2] M. Chiesa, A. Kamisinski, J. Rak, G. Retvari, and S. Schmid, "A survey of fast-recovery mechanisms in packet-switched networks," *IEEE Communications Surveys and Tutorials (COMST)*, 2021.

[3] T. Holterbach, E. C. Molero, M. Apostolaki, A. Dainotti, S. Vissicchio, and L. Vanbever, "Blink: Fast connectivity recovery entirely in the data plane," in *Proc. USENIX NSDI*, 2019, pp. 161–176.

[4] K.-T. Foerster, J. Hirvonen, Y.-A. Pignolet, S. Schmid, and G. Tredan, "On the feasibility of perfect resilience with local fast failover," in *Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2021.

[5] A. Bashandy, S. Litkowski, C. Filsfils, P. Francois, B. Decraene, and D. Voyer, "Topology independent fast reroute using segment routing," Internet Engineering Task Force, Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-21, February 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rtgwg-segment-routing-ti-lfa/21/

[6] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Failure protection in layered networks with shared risk link groups," *IEEE Network*, vol. 18, no. 3, pp. 36–41, 2004.

[7] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. of ICST SIMUTools Workshop*, 2008.

[8] J. Vanerio, S. Schmid, M. K. Schou, and J. Srba, "Mpls-kit: An mpls data plane toolkit," in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*. IEEE, 2022, pp. 49–54.

[9] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "Teavar: striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 29–43. [Online]. Available: https://doi.org/10.1145/3341302.3342069

[10] C. Jiang, Z. Li, S. Rao, and M. Tawarmalani, "Flexile: meeting bandwidth objectives almost always," in *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 110–125. [Online]. Available: https://doi.org/10.1145/3555050.3569119

[11] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to RSVP-TE for LSP tunnels," Internet Requests for Comments, IETF, RFC 4090, May 2005.

[12] S. Schmid, M. K. Schou, J. Srba, and J. Vanerio, "R-mpls: recursive protection for highly dependable mpls networks," ser. CoNEXT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 276–292. [Online]. Available: https://doi.org/10.1145/3555050.3569140