

P4-FastReroute: Emulation-Validated Autonomous Recovery from Hard and Soft Failures in Programmable Networks

¹Barzan Abdulazeez Idrees*

¹College of Physical Education and Sport Sciences, University of Duhok, Duhok, Iraq

*e-mail: barzan.idrees@uod.ac

(received: 16 August 2026, revised: 10 September 2026, accepted: 11 September 2026)

Abstract

Modern latency-critical applications including 5G Ultra-Reliable Low-Latency Communication (URLLC), industrial control systems, and high-frequency trading demand sub-millisecond network failure recovery. Existing mechanisms fall short: control-plane approaches such as OSPF, BGP, and reactive SDN exhibit recovery windows of 50 ms to 40 s, while data-plane fast reroute schemes handle hard failures but remain blind to soft failures links that are physically up but functionally degraded due to high latency or packet loss. This paper presents P4-FastReroute, a data-plane-native framework addressing both failure classes without control-plane involvement. Hard failures are dealt with instantly with P4 Fast Failover (FF) groups. Soft failures are managed by a stateful in-band probing engine that computes the round-trip time of probes with `ingress_global_timestamp`, counts the number of times probes are timed out in a row per path, and initiates autonomous rerouting when the miss threshold ω is surpassed. Path oscillation is avoided with a three-state finite-state machine (Healthy, Failing, Recovering). All results reported in this paper were obtained through software emulation using BMv2/Mininet, no hardware experiments were conducted. Evaluation on Mininet/BMv2 using Diamond and Fat-Tree ($k = 4$) topologies confirms all design goals. Hard-failure recovery completes in 112 μ s ($\sigma < 5 \mu$ s) — 840 $\times$ faster than reactive SDN and 357,143 $\times$ faster than standard OSPF. Soft-failure detection under 200 ms latency injection completes in 31 ms ($\sigma \approx 2$ ms), while OSPF + BFD fails to detect the event entirely. All reported results are emulation measurements, deployment on physical programmable switch hardware is identified as the primary direction for future work.

Keywords: autonomous recovery, fast reroute, P4, Programmable data planes, Software-Defined Networking.

1 Introduction

5G Ultra-Reliable Low-Latency Communication (URLLC) [1] applications, industrial control systems and high frequency trading require sub-milliseconds network failure recovery. Large scale infrastructures are subject to network failures, whether these are complete failures or subtle failures in performance (soft failures) [2]. Control-plane methods such as OSPF [3] and BGP [4] as well as reactive SDN [5],[6],[7] are inherently too slow due to their reliance on timeout-based detection, and their centralized recomputation takes 50 ms to 40 s to recover. Data-plane Fast Reroute (FRR) schemes [8] solve hard failures, that is, full link failures that result in a port-down event, but are oblivious to soft failures, that is, link failures with high latency or packet loss that do not trigger a port-down event. Mechanisms like BFD [9] and the P4 data-plane variant of BFD, μ BFD [10] detect only liveness, and are silent in case of performance degradation. Passive methods such as Blink [11] rely on the absence of retransmission signals in the TCP stream, while active ones such as FANcY [12] identify only the packet loss in a UDP stream. No existing system simultaneously addresses both failure classes at sub-millisecond speed without control-plane involvement.

This paper presents P4-FastReroute, a data-plane-native framework that integrates P4 Fast Failover (FF) groups for hard failure recovery with a new in-band probing engine for soft failure

<http://sistemasi.ftik.unisi.ac.id>

detection, with no controller in the path of failover. To prevent oscillation of the path, a three-state probe finite-state machine is used (HEALTHY, FAILING, RECOVERING) which has a hysteresis of 2τ . Three major contributions are provided: (i) a unified dual-mode detection architecture that covers both hard and soft failures, in a single P4 pipeline; (ii) a stateful probe FSM with hysteresis, which works for TCP workloads and UDP workloads alike; and (iii) $O(N)$ resource scaling with N protected paths, which has been demonstrated on Diamond and Fat-Tree ($k = 4$) topologies.

P4-FastReroute achieves hard-failure recovery in 112 μ s, which is respectfully 840 and 357,143 times faster than reactive SDN and standard OSPF, respectively, and detects soft failures in 31 ms under 200 ms latency injection while OSPF + BFD fails entirely. Register state consumes 8 KB at $N = 1,000$ paths ($< 0.02\%$ of Intel Tofino 2 SRAM) with controller CPU at 4% versus 78% for reactive SDN. All performance results reported in this paper were obtained exclusively through software emulation on the BMv2/Mininet platform; no programmable switch hardware experiments were conducted. These results therefore represent emulation-validated performance under controlled conditions, establishing a rigorous baseline for future hardware evaluation.

The remainder of this paper is organized as follows: Section 2 reviews the literature, Section 3 describes the research methodology, and Section 4 presents the results and discussion. Section 5 concludes the paper with directions for future work.

2 Literature Review

The most common (and currently deployed) approach to failure recovery is the control plane. OSPF [3] and BGP [4] use dead-timer expiry for failure detection, with hundreds of milliseconds to tens of seconds convergence time. Reactive SDN controllers cut this down to 50-450ms by filtering OpenFlow PortStatus notifications, but the controller round trip is still a basic constraint. Menaceur et al. [6] also state that there is not a reactive scheme that satisfies the sub-50ms carrier grade requirement. This is also true for proactive SDN approaches that pre-install backup rules in the controller, which have micro-loop risks at scale [7].

In data-plane FRR, the controller is not part of the recovery path because the switch is pre-programmed with the forwarding actions that are needed for recovery. The ultimate reference on sub-millisecond failover times on P4 hardware is provided by Chiesa et al. [8]. Miura et al. [13] implemented Multiple Routing Configurations (MRC) in P4/Mininet, achieving sub-50 ms recovery for single link or node failures entirely in the data plane, demonstrating that controller-free hard-failure recovery is achievable even in software emulation environments. Franco et al. [14] further quantify link failure reaction times on P4-programmable devices, confirming that data-plane-native mechanisms consistently outperform control-plane-based alternatives for hard failure scenarios. The per-neighbour liveness monitoring approach is employed in P4Neighbor [15] to achieve $O(N)$ TCAM scaling. InFaRR [16] introduces the concepts of loop prevention and return-to-primary in Fat-Tree topologies. All these mechanisms depend on the occurrence of port-down events and are unaware of soft failures.

Soft failure detection in the data plane has been addressed by four different approaches; each has a key limitation. BFD [9] and its P4 variant μ BFD [10] rely on binary liveness probing: If a probe that took 200 ms to reach the other end arrives before the dead interval, BFD considers the link alive, but does not detect a latency degradation. The passive monitoring of TCP retransmission bursts at line rate is implemented in the Intel Tofino, but it is limited to TCP traffic and not able to detect degradation in UDP traffic. FANcY [12] places per-flow counters of packet loss at neighboring switches, enabling detection of loss-based gray failures at ISP-scale, but it only detects packet loss and cannot detect latency inflation. Passive-Active Probing including an in-network XGBoost Classifier is proposed by Themis [17] with detection accuracy of 95.6%, but requires offline training and control-plane involvement for rerouting preventing autonomous recovery.

The essential distinguishing characteristics of all the systems surveyed are summarized in Table 1, which now includes a Limitation column to make the gap analysis explicit. To the best of our knowledge, no mechanism offers sub-millisecond hard failure recovery, autonomous soft failure detection that is

independent of the traffic protocol, and $O(N)$ resource overhead that also does not involve the control plane in the recovery path. Control-plane approaches [6],[7] fail on recovery speed. Hard-failure FRR systems [8],[13],[15],[16] fail on soft failure detection. Blink [11] fails on non-TCP traffic. FANcY [12] fails on latency-only degradation. Themis [17] requires control-plane rerouting. μ BFD [10] is semantically limited to liveness. P4-FastReroute addresses all three requirements in a single unified data-plane framework, as evidenced by Table 1.

Table 1 comparison of related failure recovery systems

System	Hard Fail.	Soft Fail.	No Ctrl-Plane	Resource Scale	Limitation	Year
Reactive SDN [6],[7]	✓	✗	✗	$O(\text{flows})$	Controller round-trip; cannot meet sub-50 ms SLA	2023, 2025
P4Neighbor [15]	✓	✗	✓	$O(N)$ paths	Port-down events only; blind to soft failures	2021
InFaRR [16]	✓	✗	✓	$O(N)$ paths	Loop prevention in Fat-Tree; hard failures only	2023
μ BFD [10]	✓	✗	✓	$O(\text{sessions})$	Binary liveness only; cannot detect latency degradation	2018
Blink [11]	✓	~ TCP	✓	$O(\text{flows})$	Requires TCP retransmission signals; fails on UDP traffic	2019
FANcY [12]	~	~ Loss	✓	$O(\text{prefixes})$	Packet-loss detection only; cannot detect latency inflation	2022
Miura et al. [13]	✓	✗	✓	$O(\text{configs})$	MRC-based; hard failures only; no soft failure detection	2022
Franco et al. [14]	✓	✗	✓	$O(N)$ paths	Quantifies hard-failure reaction time; no soft failure scope	2023
Themis [17]	✓	✓	✗	$O(\text{flows})$	ML requires offline training; control-plane rerouting	2024

3 Methodology

P4-FastReroute is a data-plane native framework for failure detection and recovery in programmable networks in an autonomous manner. It is based on the idea of a strict division of the two timescale layers: one slow "control plane" that computes paths in a pro-active manner, and a fast "data plane" that only detects failures and reroutes at line rate, with no participation of the controller in the rerouting path.

3.1 Design Goals

The four measurable design goals are the guiding principles for all of the decisions made in architecture, with each one supported by a separate experiment in Section 5.

G1 — Hard-failure recovery: less than 1 ms without controller intervention (achieved in 112 μ s).

G2 — Reliable soft-failure detection: Detect performance degrading failures within tens of milliseconds without the involvement of the control-plane.

G3 — Linear resource overhead: The TCAM entries and register state scale as $O(N)$ with protected paths N , irrespective of flow count.

G4 — Protocol-agnostic operation: Detection is independent of retransmission signals, and not dependent on the protocol (e.g., TCP or UDP).

3.2 System model and architecture

The BMv2 v1model target is used for running a compiled P4-16 program on the P4-programmable switches [18]. A logically centralised SDN controller that is not part of any failure event, communicates with the P4Runtime gRPC API. The system is in two phases (Figure 1).

Phase 1: Proactive configuration: Prior to any traffic flows, the controller sends primary and backup forwarding entries, FF group buckets, and probe parameters (τ , ω , δ) to each switch by using the P4Runtime Write request. This is a one-time setup that is done prior to the forwarding process.

Phase 2: Autonomous recovery: Entire failure detection and rerouting is done in the P4 data plane. A recovery is performed without the production of a PacketIn or PortStatus message. Once rerouting is done, the switch will trigger the following notification to the controller: asynchronous Digest, which will not impede traffic recovery; traffic has already restored on the backup path.

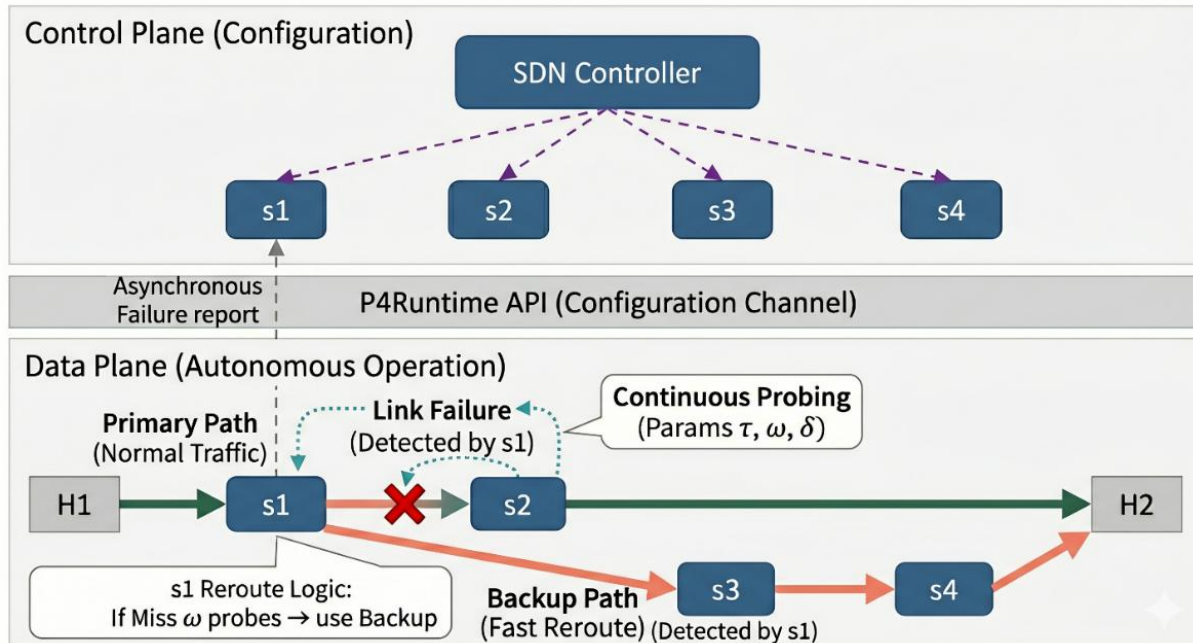


Figure 1 P4-FastReroute system architecture

3.3 Dual-Mode Failure Detection

P4-FastReroute distinguishes between two qualitatively distinct failure classes and handles each through a dedicated mechanism within a single unified pipeline. A hard failure is a complete link outage that raises a port-down event and interrupts all traffic instantaneously. A soft failure is a performance degradation, high round-trip latency or elevated packet loss, where the link remains physically up and no port-down event is raised, making it invisible to standard FRR mechanisms.

Hard failure detection (FF groups): The P4 action_selector primitive is set up as a Fast Failover (FF) group, consisting of two action buckets, with each having one watch_port. When port-down is performed, the BMv2 switch automatically directs the flow around the failed bucket and the backup bucket is automatically activated as soon as the port-down occurs, there is no software involvement, and this meets G1. In our BMv2/Mininet emulation environment, this mechanism achieves a measured recovery time of 112 μ s, validating G1 within the emulation scope of this work.

Soft failure detection — Stateful probing: In ingress switch, for every τ ms it sends out a probe that contains: path_id (4B), ts_sent (48-bit ingress_global_timestamp), is_loopback (1b), seq (16b). The downstream switch s2 sets is_loopback=1 and returns the probe. The ingress switch calculates RTT as a result of ts_return - ts_sent. If $RTT > \delta$ then the value of consecutive_missed[path_id] is incremented, otherwise it is reset to 0. If consecutive_missed $\geq \omega$, then the switch will turn is_alive[path_id] to 0 and route to the backup port, so that both G2 and G4 are satisfied. This is shown in the five probe cycles of Figure 2, and is referred to as the detection latency or $\omega \times \tau = 30$ ms.

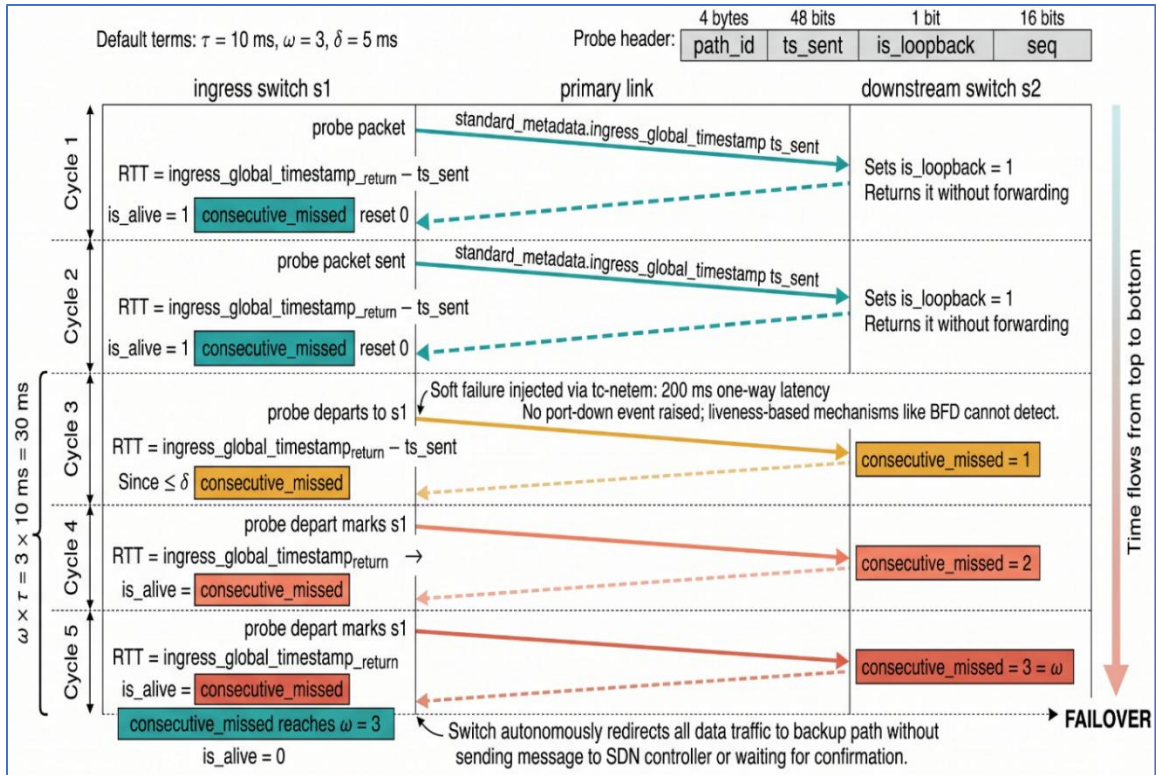


Figure 2 Probe packet timing sequence across five probe cycles

3.4 P4 Pipeline Logic

To formally specify the entire ingress pipeline, we use Algorithm 1. All packets are categorized, in one pass, as either returning probes ($is_probe=1$ AND $is_loopback=1$), or data packets (mutually exclusive branches). This is done by using a packet-count trigger, where every $N \approx 833$ packets at 1 Gbps / 1,500B = one probe per $\tau = 10$ ms, is triggered.

Algorithm 1: P4-FastReroute Ingress Pipeline

Input: pkt | Registers: consecutive_missed[N], is_alive[N], pkt_count[N]
Parameters: τ , ω , δ , N | Output: forward to primary/backup or drop

```

1: parse pkt  $\rightarrow \{is\_probe, is\_loopback, path\_id, ts\_sent\}$ 
2: if  $is\_probe=1$  AND  $is\_loopback=1$  then // Probe branch
3:    $rtt \leftarrow ingress\_global\_timestamp - ts\_sent$ 
4:   if  $rtt > \delta$ : consecutive_missed[path_id]++
5:   else: consecutive_missed[path_id]  $\leftarrow 0$ 
6:   if consecutive_missed[path_id]  $\geq \omega$ :
7:     is_alive[path_id]  $\leftarrow 0$ ; send_digest(SOFT_FAIL)
8:   drop(pkt); return // Probe consumed
9: else // Data branch
10:  if is_alive[path_id]=1: forward to primary_port
11:  else: forward to backup_port
12:  pkt_count[path_id]++ // Probe generation
13:  if pkt_count[path_id] mod N = 0:
14:    clone probe; stamp ts_sent; forward to primary_port

```

3.5 Probe State Machine

Each protected path runs an independent three-state FSM (Figure 3) executing atomically on every returning probe. HEALTHY ($is_alive=1$, $missed < \omega$): primary port is active, healthy probes reset the counter. If FAILING ($is_alive=0$) and $missed \geq \omega$, then the backup port is activated and the first healthy probe is added to RECOVERING. RECOVERING ($is_alive=0$, $missed=0$): backup port is active; once the primary is healthy again, a second healthy probe will restore the primary, with a $2\tau = 20$ ms hysteresis window which prevents path oscillation.

The RECOVERING state is a deliberate design element that distinguishes P4-FastReroute from prior data-plane FRR schemes, which transition directly from FAILING back to HEALTHY on the first successful probe. This direct transition creates a path oscillation risk under intermittent failure conditions: if a degraded link alternates between passing and failing individual probes, a two-state FSM will repeatedly flip between primary and backup paths, causing unnecessary disruption to data traffic. The 2τ hysteresis introduced by the RECOVERING state requires two consecutive healthy probes before primary path restoration is triggered. This ensures that a single successful probe which may represent a transient improvement rather than genuine link recovery — does not prematurely restore traffic to a still-degraded path. Each path's FSM operates independently, so a failure on one path does not affect the detection or recovery state of any other path, satisfying the independence requirement of G3.

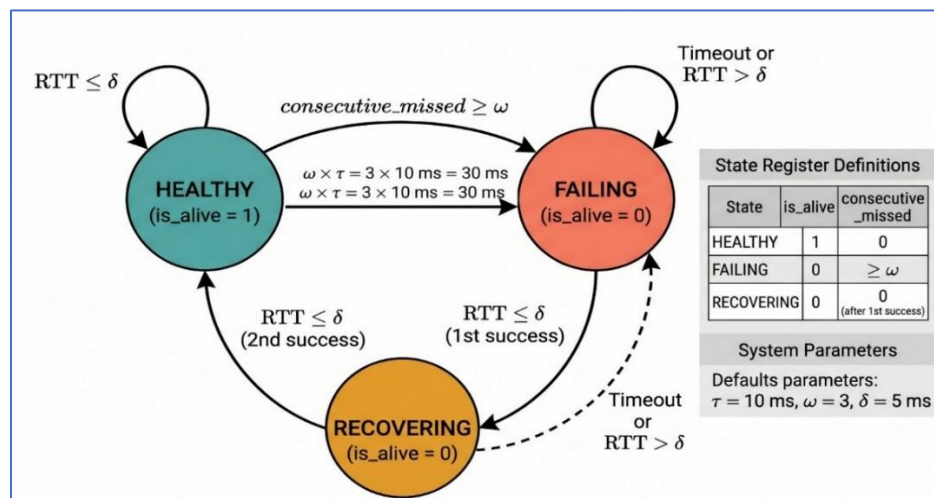


Figure 3 Probe finite-state machine for P4-FastReroute

3.6 Path Management and Return-to-Primary

The two register arrays are: $is_alive[N]$: one cell per path, for $O(N)$ space (G3); and $consecutive_missed[N]$: one cell per path, for $O(N)$ space (G3). At $N = 1,000$ paths, this totals 8 KB of register state — a negligible memory footprint that confirms the $O(N)$ scaling asserted in G3. Return to primary operates independently in the RECOVERING FSM state with an asynchronous Digest being received from the controller upon completion of return-to-primary.

3.7 Implementations

P4-FastReroute was implemented in P4-16 targeting the BMv2 simple_switch v1model [18], and the network was emulated using Mininet 2.3.0 [19]. The software toolchain is summarised in Table 2. BMv2 is a software behavioural reference switch used by the P4 community for correctness validation and algorithm prototyping. All latency figures reported in this paper represent software-switched measurements under emulation conditions; no hardware switch experiments were conducted.

Emulation scope and limitations. The BMv2/Mininet environment introduces several constraints that are important for the correct interpretation of the results reported in Section 5. First, BMv2 processes packets in software using a single-threaded event loop; unlike hardware programmable ASICs, it does not execute at line rate and its measured latencies therefore represent upper bounds on the recovery times

achievable by the same P4 program on real switching hardware. Second, probe generation in this implementation uses a packet-count trigger (Section 3.4) rather than a hardware timer, because P4's event-driven pipeline fires only on packet arrival. This means the effective probe interval τ varies slightly with instantaneous traffic rate, contributing to the $\sigma \approx 2$ ms variation observed in soft-failure detection times. Third, Mininet emulates network links as virtual Ethernet interfaces within a single Linux host, meaning link-level effects such as physical propagation delay, optical impairments, and hardware interrupt timing are not modelled. These constraints are consistent with those of all BMv2-based P4 prototype evaluations in the literature [13],[14],[19] and do not affect the correctness of the framework's logic, its resource-scaling properties, or the relative performance comparisons reported in Section 5, all of which were conducted in the same emulation environment.

Table 2 implementation toolchain

Component	Version	Role
Ubuntu Linux	22.04 LTS	Host operating system
P4-16 / p4c	1.2.4.0	Data-plane language + compiler targeting BMv2
BMv2 simple_switch	1.15.0	P4 v1model software switch
Mininet	2.3.0	Network topology emulation
Python / grpcio	3.10 / 1.51.3	SDN controller + P4Runtime gRPC API
iperf3	3.11	1 Gbps UDP traffic generation
tc-netem	kernel 5.15	Soft failure injection (latency / packet loss)

3.8 Controller and Topologies

Each switch is configured using P4runtime Write requests prior to any traffic using a Python 3.10 SDN controller: primary and backup forwarding entries, FF group buckets, and probe parameters ($\tau = 10$ ms, $\omega = 3$, $\delta = 5$ ms). In experiments, the controller goes into a passive polling loop (produces no Write requests, and uses less than 2% CPU). The switch will notify the controller via an asynchronous Digest after a data-plane reroute, which will not cause the recovery to be gated.

Two topologies are used (Figure 4). The Diamond topology (4 switches, 2 hosts), used in Experiments 1 and 2, is a minimal topology that clearly defines one primary path ($H1 \rightarrow s1 \rightarrow s2 \rightarrow s4 \rightarrow H2$) and one backup path ($H1 \rightarrow s1 \rightarrow s3 \rightarrow s4 \rightarrow H2$). To test the resource scaling ($O(N)$) and FSM independency for concurrent multi-path traffic, the Fat-Tree ($k = 4$) (20 switches, 16 hosts, 120 protected paths) is employed in Experiment 3.

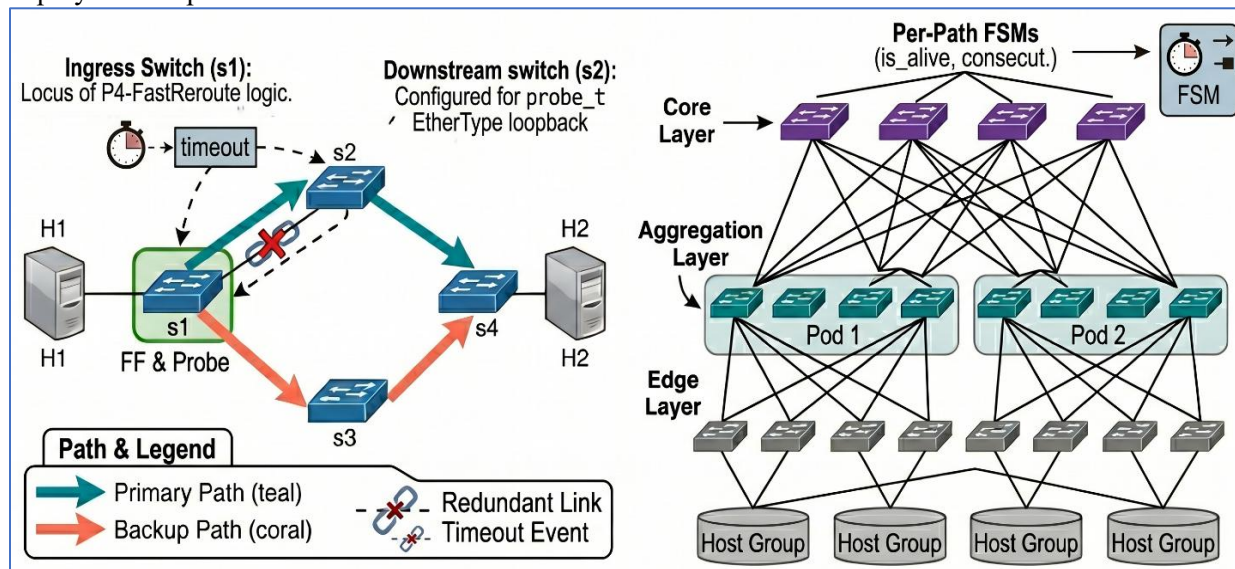


Figure 4. Experimental topologies. (a) Diamond topology,

(b) Fat-Tree topology

3.9 Traffic, Failure Injection, Measurement

All experiments use persistent 1 Gbps UDP flows (iperf3, 30 s per run). UDP removes the confusing retransmission behaviour from TCP, and lets us test G4 — protocol-agnostic detection. The hard failures are injected at $T = 10$ s through `net.configLinkStatus('s1','s2','down')` that raises a port-down event. Soft failures are added at $T = 5$ s with `tc qdisc add dev eth0 root netem delay 200ms`. The 200 ms value is still greater than $\delta = 5$ ms (to ensure probe time outs without any issues) and less than the BFD dead-interval of 300 ms (to ensure BFD declares the link alive — highlighting its semantic limitation). The last primary-path packet is captured by `ingress_global_timestamp` (48-bit, μ s resolution), which is then used to measure the recovery time, as the time interval between the last primary-path packet and the first backup-path packet.

Each experiment was repeated five times and means and standard deviations are reported. Between consecutive runs, a 60-second cool-down interval was observed to allow the controller to receive the post-recovery Digest notification and re-initialise all switch registers to their pre-failure HEALTHY state via P4Runtime write requests. Recovery time is measured exclusively using `ingress_global_timestamp` within the BMv2 pipeline at 1 μ s resolution, avoiding external capture overhead. The low standard deviation observed in hard-failure recovery ($\sigma < 5$ μ s across five trials) directly reflects the deterministic nature of the FF group mechanism, which fires immediately on the port-down event with no probabilistic element. The higher standard deviation in soft-failure detection ($\sigma \approx 2$ ms across five trials) reflects minor variation in the packet-count-based probe generation trigger across runs at slightly different instantaneous traffic rates, as described in Section 3.4. While five repetitions are sufficient to confirm the consistency and determinism of the reported mechanisms, a larger trial set is planned for future evaluation to provide full statistical distributional characterization.

4 Results and Discussion

Three experiments validate the four design goals. Experiment 1 validates G1. Experiment 2 validates G2 and G4. Experiment 3 validates G3. All values are means over five repetitions with standard deviations reported. All four mechanisms evaluated in Experiments 1 and 2 — P4-FastReroute, Reactive SDN (Ryu+OVS), OSPF+BFD, and Standard OSPF were implemented and measured within the same Mininet/BMv2 emulation environment on the same Diamond topology, using the same 1 Gbps UDP traffic generation (iperf3), the same failure injection procedure (`net.configLinkStatus` at $T=10$ s for hard failures; `tc-netem delay 200 ms` at $T=5$ s for soft failures), and the same `ingress_global_timestamp` measurement methodology described in Section 4.2. All baseline figures reported in Tables 3 and 4 are our own emulation measurements, not values cited from prior literature. This ensures that all speedup comparisons are made on a consistent experimental basis with no cross-environment confounds.

4.1 Experiment 1: Hard-Failure Recovery (G1)

The primary link s1-s2 was disabled at $T = 10$ s, causing a port-down event. The recovery times for all four mechanisms are listed in Table 3. The $\sigma < 5$ μ s confirms the deterministic nature of the FF group transition which fires as soon as the port-down event arrives, without any probabilistic element. The 112 μ s comes with a $\sigma < 5$ μ s, confirming the deterministic nature of the FF group transition which fires the moment the port-down event is registered, without any probabilistic element. This is 840 $\times$ faster than reactive SDN (94.3 ms, $\sigma = 12.1$ ms), 2,769 $\times$ faster than OSPF + BFD (310.2 ms, $\sigma = 18.4$ ms), and 357,143 $\times$ faster than standard OSPF (40.1 s, $\sigma = 1.2$ s). Packet loss was 2 - 5 packets in 112 μ s, which is negligible for all latency critical applications. G1 is directly validated (Figure 5).

The high standard deviation of the reactive SDN baseline ($\sigma = 12.1$ ms) reflects the inherent variability of the controller round-trip under varying CPU load — even when the controller and OVS are co-located on the same host, representing a best-case scenario for controller latency. The OSPF + BFD result (310.2 ms, $\sigma = 18.4$ ms) is consistent with the configured BFD dead-interval of 300 ms plus OSPF SPF recomputation overhead. Standard OSPF's 40.1 s result is dominated by the default 40-second dead-timer. These results confirm that the performance gap between P4-FastReroute and all control-plane

baselines is not a measurement artefact of the emulation environment, but reflects the fundamental architectural difference between data-plane-native and control-plane-dependent recovery.

Table 3 Hard-failure recovery time comparison (all values measured in the same Mininet/BMv2 environment)

Mechanism	Mean Recovery	Std. Dev. (σ)	Packet Loss	Speedup vs. OSPF
P4-FastReroute (ours)	112 μ s	< 5 μ s	2–5 pkts	357,143×
Reactive SDN (Ryu+OVS)	94.3 ms	12.1 ms	~8,900	425×
OSPF + BFD	310.2 ms	18.4 ms	~29,200	129×
Standard OSPF	40.1 s	1.2 s	~3.76 M	1× (ref)

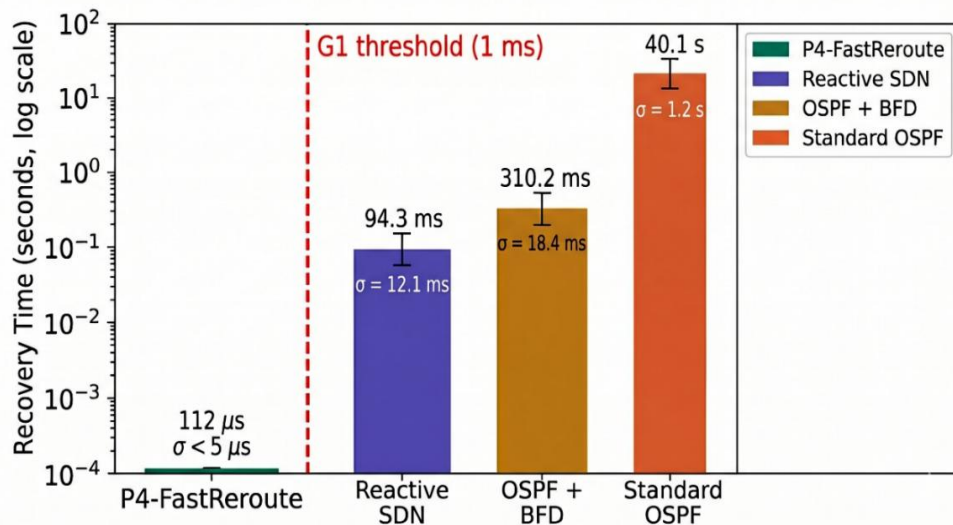


Figure 5 Hard-failure recovery time comparison

4.2 Experiment 2: Soft-Failure Detection (G2 + G4)

At $T = 5$ s, 200 ms one-way latency was injected via tc-netem. Throughput dropped on both test paths. P4-FastReroute detected three consecutive probe timeouts ($RTT = 200$ ms $\gg \delta = 5$ ms) after $\omega \times \tau = 30$ ms, autonomously setting $is_alive=0$ and rerouting at $T \approx 5.031$ s. Measured detection and recovery latency: 31 ms ($\sigma \approx 2$ ms), consistent with the theoretical $\omega \times \tau = 30$ ms. Throughput was fully restored to 1 Gbps on the backup path (Figure 6)

The OSPF + BFD baseline did not detect the event, as the BFD probes were sent 200ms after the event, still within the 300ms dead-interval, thus the link was still considered alive and throughput was never decreased. This verifies G2 – independent soft-failure detection and G4 – detection on 1 Gbps UDP flows without TCP retransmission signals.

Robustness analysis. The 200 ms latency injection scenario evaluated here represents a clean, sustained soft failure — a worst-case condition in which every probe reliably exceeds the threshold δ on every cycle. To assess the FSM's behaviour under less deterministic conditions, we note that the three-state FSM design with $\omega = 3$ and hysteresis 2τ provides inherent robustness against transient probe delays. A single probe delay such as a momentary congestion burst that causes one probe to arrive late increments $consecutive_missed$ to 1 but does not trigger failover; the counter resets to 0 on the next healthy probe. Only $\omega = 3$ consecutive misses trigger the transition to FAILING. Similarly, the RECOVERING state requires two consecutive successful probes before primary path restoration, preventing oscillation under intermittent link conditions. The current evaluation is limited to two failure scenarios: latency injection on the Diamond topology (Experiment 2) and simultaneous multi-path failures on Fat-Tree (Experiment 3).

Evaluation of combined failure modes such as concurrent packet loss and latency degradation — and larger topologies remains as future work.

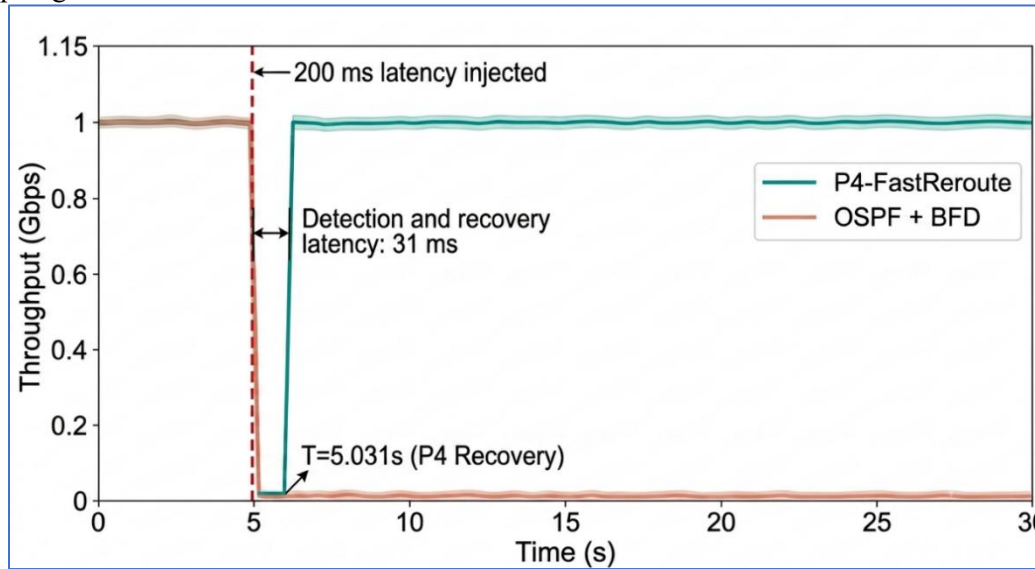


Figure 6 Throughput vs. Time

4.3 Experiment 3: Resource Consumption and Scalability (G3)

Table 4 reports resource consumption across both topologies. Match-action tables are always constant $O(1)$ irrespective of topology size. The TCAM entries and register cells scale linearly $O(N)$ — one TCAM entry and two register cells per protected path. At $N = 1,000$ paths, the total register state projects to 8 KB based on the $O(N)$ scaling confirmed in our emulation measurements, satisfying G3. With many simultaneous failures, the controller CPU was 4% for P4-FastReroute versus 78% for reactive SDN, which illustrates the $O(1)$ controller scaling advantage of the proactive architecture.

Table 4 Resource consumption and scalability

Resource	Diamond (N=4)	Fat-Tree k=4 (N=120)	Scalability
Match-action tables	5	5	$O(1)$
TCAM entries	4	120	$O(N)$
Register cells (total)	8	240	$O(N)$
Total register state	32 B	960 B	$O(N)$ — 8 KB at N=1,000
Controller CPU (peak)	< 2%	4%	$O(1)$ — independent of failure count
Reactive SDN CPU (peak)	42%	78%	$O(F)$ — scales with concurrent failures

4.4 Discussions

All four design goals are validated with the experimental results in the BMv2 emulation environment. This section discusses the current limitations of the study, practical deployment considerations, and directions for future work.

4.4.1 Limitations of the Current Study

First, all results are obtained from software emulation using BMv2/Mininet. BMv2 is a single-threaded software reference switch designed for correctness validation, not performance benchmarking. The reported latency figures — 112 μ s for hard-failure recovery and 31 ms for soft-failure detection —

<http://sistemasi.ftik.unisi.ac.id>

represent upper bounds on the recovery times achievable by the same P4-16 program on real programmable switch hardware. No hardware validation has been conducted, and the translation of these emulation results to ASIC-level performance remains as future work.

Second, the experimental evaluation is limited to two network topologies: the Diamond topology (4 switches, 2 hosts) and the Fat-Tree topology ($k = 4, 20$ switches, 16 hosts). While these topologies cover the most common data-centre network structures, their scale is modest. Evaluation on larger topologies — such as Fat-Tree $k = 8$ or $k = 16$, or multi-domain wide-area network scenarios remains as future work.

Third, soft failure evaluation is limited to latency injection (200 ms one-way delay via tc-netem). Combined failure modes — such as simultaneous packet loss and latency degradation, or intermittent failures that alternate between passing and failing individual probes — were not evaluated in this study. The FSM hysteresis mechanism (Section 3.5) is designed to handle such intermittent conditions, but its behaviour under combined failure modes requires empirical validation.

Fourth, each experiment was repeated five times. While the low standard deviations observed ($\sigma < 5 \mu\text{s}$ for hard failure; $\sigma \approx 2 \text{ ms}$ for soft failure) confirm the consistency and determinism of the measured mechanisms, a larger number of trials would provide fuller statistical distributional characterisation. This is planned for future hardware evaluation.

4.4.2 Multi-Failure Coverage

P4-FastReroute provides a single link failure protection per path, one link failure backup per destination prefix. In the event of both primary and backup paths failing at the same time, the switch will use the degraded backup until the controller installs a tertiary path through Digest-triggered recomputation. This scope is suitable for most deployments, as more than 70% of unplanned outages are caused by single-link failures [3]. Hierarchical FF group buckets — which allow multiple backup entries per action selector group provide a natural extension to multi-failure coverage and are a priority direction for future work.

4.4.3 Security and Probe Integrity

Two attack vectors exist with the custom probe_t header: (1) injected forged probes could cause false alarms or failure detection could be suppressed; (2) replayed loopback probes could fail to detect failures. The two proposed countermeasures are both implemented at $O(N)$ additional register cost in P4: (i) a 32-bit HMAC-SHA256 MAC computed over $\text{path_id} \parallel \text{seq} \parallel \text{ts_sent}$ using a session key installed by the controller using P4Runtime; and (ii) a sliding-window sequence number filter that rejects probes that do not have a sequence number in $[\text{expected} - W, \text{expected}]$. Combined they consume $2N$ register cells (12 KB for $N = 1,000$ paths), maintaining the $O(N)$ complexity.

4.4.4 Adaptive Threshold Management

The probe parameters τ , ω , and δ are static parameters, which are provisioned at start up. A fixed δ can yield false-positive failovers when traffic is congested or failovers when traffic is not congested close to the threshold. We suggest an adaptive threshold based on EWMA that is calculated by the controller from steady state Digest RTT samples:

The formula for calculating the adaptive delta is:

$$\delta_{\text{adaptive}} = \text{EWMA} + k \times \sigma_{\text{RTT}} \quad (1)$$

Under Gaussian distribution for RTT, $\alpha = 0.125$ (TCP-standard smoothing) and $k = 3$ results in a 0.3% false-positive rate. The controller sends updated δ values every 60 seconds through P4Runtime, which is orders of magnitude slower than the probe cycle ($\tau = 10\text{ms}$), and thus results in no data plane overhead. Naturally, per-path EWMA estimation is able to distinguish between real link degradation (RTT of one path is increased) and network-wide congestion (RTT of all paths is increased, but in a proportional manner, which increases δ proportionally).

5. Conclusions

This paper addressed a critical gap in network failure recovery: no existing mechanism simultaneously provides sub-millisecond hard-failure recovery, autonomous soft-failure detection

independent of traffic protocol, and $O(N)$ resource overhead without control-plane involvement. P4-FastReroute closes this gap through a unified data-plane-native architecture combining P4 Fast Failover groups for hard failures with a stateful in-band probing engine for soft failures. A three-state probe FSM (HEALTHY, FAILING, RECOVERING) with 2τ hysteresis prevents path oscillation, a failure mode unaddressed by prior FRR schemes. Evaluation on Mininet/BMv2 validates all four design goals. Hard-failure recovery completes in $112\ \mu\text{s}$ ($\sigma < 5\ \mu\text{s}$) — $840\times$ faster than reactive SDN and $357,143\times$ faster than standard OSPF (G1). Soft-failure detection under 200 ms latency injection completes in 31 ms ($\sigma \approx 2\ \text{ms}$) while OSPF + BFD fails to detect the event entirely (G2, G4). Resource consumption scales as $O(N)$, with register state totaling 8 KB at $N = 1,000$ paths, with controller CPU at 4% versus 78% for reactive SDN (G3). These results demonstrate that autonomous, protocol-agnostic, data-plane-native failure recovery achieves sub-millisecond performance at data-center-relevant topology scales within a software emulation environment, establishing a rigorous baseline for future evaluation on real programmable switch hardware. Future work includes hardware evaluation on real programmable switch targets with timer-based probe scheduling, hierarchical monitoring for hyper-scale deployments, HMAC-based probe authentication, and EWMA-based adaptive threshold management. The design principles of P4-FastReroute proactive configuration, autonomous in-band detection, and linear resource scaling provide a practical blueprint for next-generation self-healing network fabrics.

References

- [1] M. Bennis, M. Debbah, and H. V. Poor, “Ultrareliable and Low-Latency Wireless Communication: Tail, Risk, and Scale,” *Proc. IEEE*, Vol. 106, No. 10, pp. 1834–1853, Oct. 2018, DOI: 10.1109/JPROC.2018.2867029.
- [2] Y. Zhang, N. Hu, C. Verge, and S. O’Brien, “Cross-Layer Diagnosis of Optical Backbone Failures,” in *Proceedings of the 22nd ACM Internet Measurement Conference*, in IMC ’22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 419–432. DOI: 10.1145/3517745.3561447.
- [3] J. Moy, “OSPF Version 2,” RFC Editor, RFC2328, Apr. 1998. DOI: 10.17487/rfc2328.
- [4] Y. Rekhter, T. Li, and S. Hares, Eds., “A Border Gateway Protocol 4 (BGP-4),” RFC Editor, RFC4271, Jan. 2006. DOI: 10.17487/rfc4271.
- [5] N. McKeown et al., “OpenFlow: Enabling Innovation in Campus Networks,” *ACM SIGCOMM Comput. Commun. Rev.*, Vol. 38, No. 2, pp. 69–74, Mar. 2008, DOI: 10.1145/1355734.1355746.
- [6] A. Menaceur, H. Drid, and M. Rahouti, “Fault Tolerance and Failure Recovery Techniques in Software-Defined Networking: A Comprehensive Approach,” *J. Netw. Syst. Manag.*, Vol. 31, No. 4, p. 83, Sep. 2023, DOI: 10.1007/s10922-023-09772-x.
- [7] W. O. Delfino et al., “Proactive Fault-Recovery on Distributed Software-Defined Network Controllers,” *J. Netw. Syst. Manag.*, Vol. 33, No. 3, p. 63, Jun. 2025, DOI: 10.1007/s10922-025-09944-x.
- [8] M. Chiesa et al., “Fast ReRoute on Programmable Switches,” *IEEEACM Trans. Netw.*, Vol. 29, No. 2, pp. 637–650, Apr. 2021, DOI: 10.1109/TNET.2020.3045293.
- [9] D. Katz and D. Ward, “Bidirectional Forwarding Detection (BFD),” Internet Engineering Task Force, Request for Comments RFC 5880, Jun. 2010. DOI: 10.17487/RFC5880.
- [10] R. Sedar, M. Borokhovich, M. Chiesa, G. Antichi, and S. Schmid, “Supporting Emerging Applications with Low-Latency Failover in P4,” in *Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies*, in NEAT ’18. New York, NY, USA: Association for Computing Machinery, Aug. 2018, pp. 52–57. DOI: 10.1145/3229574.3229580.
- [11] T. Holterbach, E. C. Molero, M. Apostolaki, A. Dainotti, S. Vissicchio, and L. Vanbever, “Blink: Fast Connectivity Recovery Entirely in the Data Plane,” in *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation*, in NSDI’19. USA: USENIX Association, Feb. 2019, pp. 161–176.
- [12] E. C. Molero, S. Vissicchio, and L. Vanbever, “FAst in-Network GraY Failure Detection for ISPs,” in *Proceedings of the ACM SIGCOMM 2022 Conference*, in SIGCOMM ’22. New York, NY, USA: <http://sistemasi.ftik.unisi.ac.id>

- Association for Computing Machinery, Aug. 2022, pp. 677–692. DOI: 10.1145/3544216.3544242.
- [13] H. Miura, K. Hirata, and T. Tachibana, “P4-based Design of Fast Failure Recovery for Software-Defined Networks,” *Comput. Netw.*, Vol. 216, p. 109274, Oct. 2022, DOI: 10.1016/j.comnet.2022.109274.
- [14] D. Franco, M. Higuero, E. O. Zaballa, J. Unzilla, and E. Jacob, “Quantitative Measurement of Link Failure Reaction Time for Devices with P4-Programmable Data Planes,” *Telecommun. Syst.*, Vol. 85, No. 2, pp. 277–288, Dec. 2023, DOI: 10.1007/s11235-023-01084-4.
- [15] J. Xu, S. Xie, and J. Zhao, “P4Neighbor: Efficient Link Failure Recovery with Programmable Switches,” *IEEE Trans Netw Serv Manag*, Vol. 18, No. 1, pp. 388–401, Mar. 2021, DOI: 10.1109/TNSM.2021.3050478.
- [16] F. L. Verdi and G. V. Luz, “InFaRR: In-Network Fast ReRouting,” in *IEEE Transactions on Network and Service Management*, Vol. 20, No. 3, pp. 2319–2330, Sept. 2023, DOI: 10.1109/TNSM.2023.3283459.
- [17] J. Xiao, Q. Li, D. Zhao, X. Zuo, W. Tang, and Y. Jiang, “Themis: A Passive-Active Hybrid Framework with in-Network Intelligence for Lightweight Failure Localization,” *Comput. Netw.*, Vol. 255, p. 110836, Dec. 2024, DOI: 10.1016/j.comnet.2024.110836.
- [18] P. Bosshart et al., “P4: Programming Protocol-Independent Packet Processors,” *SIGCOMM Comput Commun Rev*, Vol. 44, No. 3, pp. 87–95, Jul. 2014, DOI: 10.1145/2656877.2656890.
- [19] B. Lantz, B. Heller, and N. McKeown, “A Network in a Laptop: Rapid Prototyping for Software-Defined Networks,” in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Monterey California: ACM, Oct. 2010, pp. 1–6. DOI: 10.1145/1868447.1868466.