INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY TRENDS

Vol. 6 No. 02 (2026)

e.ISSN: 2409-5222

https://ijictt.ictt.info

DRL-Based OSPF Convergence Optimization: From Simulation to Network Emulation

Jonah Desai [2], Elijah Phillips[1], Seojun Miller [3*x]

[1] *Luddy School of Informatics, Computing, and Engineering, Indiana University Bloomington, Bloomington, USA*

[2] *Department of Computer Science, University of Texas at Austin, Austin, USA*

[3]*Tandon School of Engineering, New York University, New York, USA*

Email: s03819341@nyu.edu

Abstract

Deep reinforcement learning (DRL) has been proposed to tune OSPF convergence timers online, but such agents are trained and evaluated in simulators whose fidelity to real routing stacks is rarely checked. We take DRL-based OSPF convergence optimization from simulation to network emulation. An FRRouting testbed running production ospfd/zebra daemons in Linux namespaces is paired with a protocol-level simulator at three fidelity levels (idealized, FRR-faithful, trace-calibrated) that share scenarios, observations, actions and metrics. Calibration identifies 4.4 ms carrier-loss detection, 2.2 ms per-hop flooding and about 1.0 s of configuration-push skew. On real ISP topologies and traffic matrices, the idealized simulator misestimates the per-episode cost by 62.7% on average and the restoration time of hard failures by a factor of 66, whereas the calibrated simulator stays within 12.6%. We re-implement the DRL-Adapt agent, use its Markov decision process as the common control formulation, and extend it with a feasibility-aware action-branching design. Deployed on the testbed, the re-implemented DRL-Adapt agent lowers the cost by 31.6% (trained in the idealized simulator) and 32.9% (trained in the calibrated simulator) relative to RFC default timers, a gain comparable in magnitude to the convergence-time reduction originally reported for DRL-Adapt. The agent nevertheless converges to a near-static operating point, and a static configuration found by searching the calibrated simulator reduces the cost by 32.7%, within the scenario-to-scenario variability of both DRL-Adapt variants. Our results indicate that DRL-Adapt's improvement over default timers transfers from simulation to a production routing stack, and that, under realistic protocol constraints, calibrated configuration search captures most of the benefit of adaptive timer control on the evaluated topologies.

Keywords *: OSPF, routing convergence, deep reinforcement learning, network emulation, FRRouting, sim-to-real transfer, protocol timers.*

1. Introduction

Link-state interior gateway protocols carry most intra-domain traffic, and OSPF [1] is the most widely deployed of them. When a link or a router fails, the time OSPF needs to detect the failure, flood the new topology and recompute forwarding tables bounds how long traffic is black-holed. That time is governed by a handful of protocol timers: the Hello and Dead intervals that reveal a silent neighbor, the throttle that paces router-LSA origination, and the back-off that schedules shortest-path-first (SPF) computations [2], [3]. Measurements showed that sub-second convergence is attainable with careful tuning [4], but also that aggressive timers make adjacencies fragile when control packets are lost on congested or degraded links [5], [6]. Because failures in operational backbones are frequent, short-lived, often correlated through shared risk and concentrated on a few flapping links [7], [8], no single setting is best at all times. This tension makes timer tuning a natural target for learning-based control. Reinforcement learning was applied to packet routing early on [9] and, with deep function approximation, to routing and traffic engineering [10], [11]. Recently, DRL-Adapt [12] cast routing-convergence optimization as a Markov decision process in which a dueling double deep Q-network with prioritized replay adjusts Hello and Dead intervals from an event-stream view of the network, and for large-scale fully meshed networks it was reported to shorten fault convergence by 32.7% and control-plane overhead by 24.3% relative to static configurations. Its event-stream state, multi-objective reward and stabilized actuation give a complete formulation of online timer control, which we adopt as the common formulation of this study. Like most studies of deep reinforcement learning (DRL) for networking, however, that evaluation stays inside the simulator used for training, so whether its gains survive a production routing stack is an open question. That step is more than a formality. The reality gap that complicates sim-to-real transfer in robotics [13], [14] has a direct analogue in routing: an agent exploits exactly the dynamics its simulator encodes. A timer-centric simulator in which every failure waits for the Dead interval rewards ever faster Hellos. A production OSPF stack, in contrast, detects carrier loss within milliseconds, accepts only a discrete lattice of timer values, discards Hellos whose timers disagree with its own, and paces LSAs and SPF runs in ways that dominate convergence once detection is fast. Whether policies learned in simulation remain useful, or even safe, when they drive a real routing daemon is therefore an empirical question. This paper answers it for OSPF. We build an emulation testbed in which every router runs the production FRRouting [15] ospfd and zebra daemons in its own Linux network namespace, together with a protocol-level simulator offered at three fidelity levels that share failure scenarios, observations, actions and metrics with the testbed. Using real ISP topologies and traffic matrices from the Internet Topology Zoo and Rocketfuel [16], [17], as packaged by REPETITA [18], we make four contributions.

(i) A simulation-to-emulation methodology for OSPF control: an FRRouting testbed and a protocol-level simulator with three fidelity levels that share failure scenarios, a 42-dimensional observation, FRR-feasible actions and a traffic-weighted disruption metric, together with a trace-driven calibration procedure. (ii) A fidelity study showing that an idealized timer-centric simulator misestimates per-episode cost by 62.7% on average and hard-failure restoration time by a factor of 66, whereas the FRR-faithful and calibrated levels stay within about 12.6%. (iii) A re-implementation of the DRL-Adapt agent and an emulation study of its transfer, which shows that its gain over default timers carries over to the real stack while the learned policy settles on a near-static operating point. (iv) A feasibility-aware action-branching agent with knob-specific stabilization and confidence gating, together with the finding that neither learned policy improves on a static configuration obtained by searching the calibrated simulator beyond scenario-to-scenario variability, which locates the value of simulation in calibrated configuration search rather than in online adaptation.

The rest of the paper is organized as follows. Section II reviews related work, Section III presents the testbed, the simulator, the calibration procedure and the agents, Section IV describes the experimental setup, Section V reports the results, and Section VI discusses limitations and concludes.

II. Related Work

A. OSPF Convergence and Timer Tuning

The components of link-state convergence, namely failure detection, LSA origination and flooding, SPF computation and FIB update, were measured and modeled by Francois *et al.* [4], and the many refinements that followed are surveyed in [2]. The cost side of fast timers is equally well documented: Hellos lost under congestion cause spurious adjacency failures and route flaps [5], and sub-second Hello timers increase routing load and flaps in a detailed simulation of a large ISP network [6]. Implementations therefore throttle LSA origination and back off SPF computations, an approach now standardized for interoperability [3], whereas dedicated liveness protocols such as BFD [19] move fast detection out of the IGP. Gray failures, i.e., partial and intermittent losses that evade binary failure detectors, are a known weakness of timer-based detection [20]. These works configure timers statically; we ask whether a learned policy can adapt them online on a real implementation.

B. Learning-Based Routing and Network Control

Q-routing introduced reinforcement learning to packet forwarding [9]; deep reinforcement learning has since been used for routing and traffic engineering [10], [11], and graph neural networks serve as fast network models for optimization [21]. Our agents build on DQN [22] and its double [23], dueling [24] and prioritized-replay [25] extensions, and on action branching [26] for factored action spaces. Learning-based controllers of this kind are predominantly trained and evaluated in simulation, which is precisely the gap studied here. Uncertainty-aware predictive elastic scheduling offers an informative systems-control example of trading rapid adaptation against resource overhead under changing workloads, a useful consideration when assessing routing-control policies [30].

C. Network Emulation and Sim-to-Real Transfer

Container-based emulation runs unmodified network software in lightweight namespaces [27] and has become a standard vehicle for reproducible experiments [28]. Network digital twins extend this idea to data-driven models that mirror a production network for what-if analysis [29]. In robotics, domain randomization [13] and dynamics randomization [14] are established remedies for the simulation-to-reality gap. We transpose them to routing: the simulator is calibrated against the emulator, and the identified parameters are randomized during training. Risk-aware digital-twin validation of multi-tenant network services provides a closely related experimental perspective on controlled scenario replay, operational constraints, and reliability-focused evaluation [31].

III. System Design

Fig. 1 summarizes the workflow. Failure scenarios are generated once and replayed identically in the simulator and in the emulator. Agents are trained only in simulation, the emulator supplies calibration traces and serves as the deployment target, and both environments compute the same observations and metrics from the same kinds of evidence, namely adjacency events, FIB updates and OSPF packet counters.

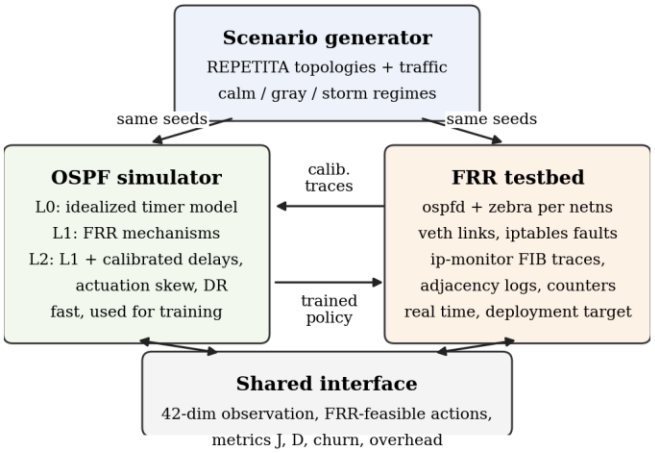


Figure 1Simulation-to-emulation workflow. Scenarios, observations, actions and metrics are shared; calibration traces flow from the FRR testbed to the simulator, trained policies flow back to the testbed.

A. Emulation Testbed

Each router is an FRRouting 8.4.4 instance (zebra and ospfd) confined to its own network namespace. Links are veth pairs configured as OSPF point-to-point interfaces with the IGP weights of the topology, and every router advertises a /32 loopback that serves as traffic endpoint. Three fault types are injected. A *hard* failure brings an interface down, so both ends observe carrier loss. A *silent* failure drops all packets in both directions with iptables rules while the interfaces stay up, so only the Dead interval can reveal it. A *gray* link drops packets at random, alternating between loss bursts and quieter periods. Since the guest kernel offers no netem queueing discipline, losses are produced with the iptables statistic match and link propagation delay is not emulated. Forwarding state is recorded with microsecond time-stamps by one ip-monitor process per namespace, adjacency transitions are read from ospfd logs, and OSPF packet counts from iptables counters. Reconfigurations are pushed to all routers concurrently through vtysh. Before each episode the testbed waits until all adjacencies are Full and no route has changed for 5 s; because a throttled LSA from the preceding reconfiguration can still arrive afterwards, the FIB state at the start of an episode is taken as the snapshot updated with every route event logged before the start, which affected one of the 70 emulated episodes.

B. Simulator and Fidelity Levels

The simulator is an event-driven model of OSPF on the same topologies. Its protocol-faithful level, L1, mirrors the mechanisms of FRR's ospfd: per-interface Hello timers with independent phases; Hellos are discarded when the Dead interval, or outside fast-hello mode the Hello interval, differs from the receiver's configuration; running inactivity timers are not rescheduled when timers change; adjacencies progress through Init, 2-Way and Full, with DBD retransmission after the RxmtInterval when exchange packets are lost, and a re-initializing neighbor tears down its peer through a sequence-number mismatch; router-LSA origination is throttled, receivers discard instances that arrive within MinLSArrival of the previous one, which forces a retransmission; SPF runs follow FRR's throttle with an initial delay and a hold time that grows with a back-off multiplier up to a maximum; and timers are restricted to FRR's lattice of integer seconds or fast-hello mode with a 1 s Dead interval. Hard failures are detected through carrier loss.

Level L0 is an idealized, timer-centric abstraction of the kind commonly used to train timer-tuning agents: every failure is detected by Dead-interval expiry, LSA origination, flooding, SPF and FIB updates are instantaneous, timers are continuous, and reconfiguration is atomic. Level L2 is L1 with processing delays, actuation skew and packet multipliers identified from emulation traces.

C. Trace-Driven Calibration

L2 is identified from a dedicated set of emulation runs whose scenarios are disjoint from the test scenarios. For isolated hard failures under configurations without SPF delay, the time from failure to the first FIB

update of a router grows linearly with the router's hop distance to the failed link: the slope yields the per-hop flooding delay and the intercept the sum of carrier detection, origination, SPF and FIB-installation delays, while carrier detection itself is measured from ospfd's neighbor-kill time-stamps. Route-install spacing, DBD exchange time and the duration of the concurrent configuration push, which bounds actuation skew, are measured directly. The per-LSA and per-adjacency packet multipliers are fitted to emulated OSPF packet counts. Table I lists the identified values. During training the delays are scaled by log-uniform factors in [0.5, 2] and the actuation skew by uniform factors in [0.5, 1.5], a form of dynamics randomization.

Table 1IDENTIFIED L2 PARAMETERS (FROM 177 HOP-DELAY AND 42 CARRIER SAMPLES)

Parameter	**L1 nominal**	**L2 identified**
Carrier-loss detection	0	4.4 ms
Flooding delay per hop	0	2.2 ms
SPF + FIB install (sum)	0	20.7 ms
Route install spacing	0	3.1 µs
DBD exchange	0	0.8 ms
Actuation skew (max)	0	1.01 s
Packets per LSA per hop	2	2.5
Packets per adjacency setup	6	4

With the identified multipliers, simulated OSPF packet counts deviate from the emulated ones by 5.1% on average over the calibration runs.

D. Control Problem

We adopt the Markov decision process and the learning machinery of DRL-Adapt [12] as the common formulation of every learned agent in this study, and extend them only where the real stack requires it. A centralized agent acts every $\Delta = 5$ s. Its state concatenates four blocks, $s_t = [f_{event}, f_{topo}, f_{proto}, f_{link}]$. The event block is DRL-Adapt's event-stream encoding: the window $[t - \tau, t]$ with $\tau = 15$ s is divided into $k = 6$ bins, each counting adjacency-down events, FIB updates and adjacency-up events, followed by window aggregates (update rate, affected destinations, inter-arrival variance). The topology block contains DRL-Adapt's localized vulnerability

$$V_{topo} = \sum_{v \in V} b(v) e^{-\lambda\, d(v,f)} \qquad (1)$$

computed over nodes within two hops of recently failed adjacencies ($b(v)$ is normalized betweenness, $d(v,f)$ the hop distance, $\lambda = 0.7$), plus static graph descriptors. The protocol block holds the current Hello, Dead, SPF delay, SPF hold and LSA throttle values (log-normalized), their change since the previous decision and the time since the last change. The link block is our addition: for each link e we estimate the Hello loss

$$\ell_e = 1 - \frac{n_e}{2W/h} \qquad (2)$$

from the OSPF packets received on both ends during the last $W = 20$ s, with h the Hello interval, and summarize it by its mean, its maximum and the fractions of links above 0.3 and 0.95. A loss near one indicates a silent failure that has not yet been detected, intermediate values a gray link. All 42 features are computed identically from emulator logs and counters and from simulator events.

The reward follows DRL-Adapt's multi-objective structure, balancing disruption, control overhead and stability:

$$r_t = -(\alpha \frac{D_t}{D} + \beta \frac{O_t}{\tilde{O}} + \eta \frac{F_t}{F}) \qquad (3)$$

where O_t counts OSPF packets sent, F_t counts FIB updates (route churn) and D_t is the traffic-weighted disruption

$$D_t = \int_t^{t+\Delta} \sum_{s,d} w_{sd}(1 - \varphi_{sd}(\tau)) c_{sd}(\tau) d\tau \qquad (4)$$

with demand weights w_{sd} from the traffic matrix, $c_{sd}(\tau)$ indicating that d is physically reachable from s, and $\varphi_{sd}(\tau)$ the fraction of the demand that the current FIBs deliver, splitting equally over ECMP next hops and counting branches that enter a failed link or a loop as lost. Disruption is thus black-holed demand integrated over time. The normalizers are per-topology means per interval: $\bar{D}$ and $\bar{F}$ under the RFC default timers and $\bar{O}$ under a 1 s/4 s Hello/Dead setting, which plays the role of DRL-Adapt's acceptable-overhead threshold. We use $\alpha = 1$, $\beta = 0.05$ and $\eta = 0.1$, fixed before any agent was trained, and evaluate every policy by the episode cost J, the mean of $-r_t$ over decision intervals.

E. Agents

DRL-Adapt agent. We re-implemented the agent from its description: a one-dimensional convolution (8 filters, kernel 3) over the temporal event bins whose output is concatenated with the remaining features, a shared trunk of two 128-unit ReLU layers, and dueling value and advantage streams (64 units each) combined with max-based aggregation

$$Q(s,a) = V(s) + A(s,a) - max_{a'}A(s,a') \quad (5)$$

trained with double-Q targets, proportional prioritized replay with importance-sampling weights, a Huber loss and gradient clipping. Its actions are DRL-Adapt's multiplicative adjustments of the Hello and Dead intervals by factors in {0.5, 0.75, 1, 1.25, 1.5} (25 joint actions), bounded to [0.5, 10] s and [2, 40] s, with the Dead interval projected to at least three Hello intervals and a stabilization interval between changes. On L1, L2 and the testbed, the continuous values are projected onto FRR's timer lattice.

Feasibility-aware branching agent. Our agent keeps the encoder, trunk, value stream and max-based aggregation but replaces the joint action space by four advantage branches, one per protocol knob, defined directly on FRR-feasible value ladders: the detection pair (Hello, Dead) in {(0.25, 1), (0.33, 1), (0.5, 1), (1, 3), (1, 4), (2, 8), (3, 12), (5, 20), (10, 40)} s, the SPF initial delay in {0, 10, 50, 100, 200, 500} ms, the SPF hold time in {10, 50, 100, 200, 500, 1000} ms, and the LSA throttle in {0, 100, 250, 500, 1000, 2500, 5000} ms. MinLSArrival is tied to the LSA throttle (capped at 1 s), so that neighbors never discard an instance the originator is allowed to send. Each branch selects a move in {−2, −1, 0, +1, +2} along its ladder, and branch targets are averaged as in action-branching DQN [26]:

$$y = r + \frac{\gamma}{B}\sum_{b=1}^{B} Q_b^-(s', argmax_a Q_b(s', a)) \quad (6)$$

with each branch regressing $Q_b(s, a_b)$ towards y. Two actuation rules reflect what the emulator taught us. First, the stabilization interval (10 s) applies only to Hello/Dead changes, because both ends of an adjacency must agree on these timers, whereas SPF and LSA throttles are router-local and may change at every decision. Second, a small switching cost (0.03 per Hello/Dead change) is added to the training reward to discourage needless reconfiguration; it is not part of the evaluation metric. The agent is trained in L2 with dynamics randomization.

Both agents use the same budget: 2500 training episodes of 120 s (23 decision steps per episode, 57,500 transitions in total), discount 0.95, Adam with learning rate 10^{-4}, mini-batches of 64, a replay memory of 60,000 transitions, target updates every 500 steps, ε-greedy exploration decaying from 1 to 0.05 over 60% of training, prioritization exponent 0.6 and importance-sampling exponent annealed from 0.4 to 1. Training uses Abilene, AT&T and GÉANT and starts half of the episodes from a random configuration.

IV. Experimental Setup

Topologies and traffic. Table II lists the four topologies. Abilene and AT&T are used in emulation and simulation, GÉANT completes the training set, and Exodus is never seen during training. IGP weights and traffic matrices come from REPETITA [18]; the matrices weight the disruption metric (4).

Table 2TOPOLOGIES

Topology	Nodes	Links	Diam.	Use
Abilene (Topology Zoo)	11	14	5	train, sim, emu
AT&T MPLS (Topology Zoo)	25	56	5	train, sim, emu
GÉANT 2010 (Topology Zoo)	37	56	7	train, sim
Exodus AS3967 (Rocketfuel)	79	147	10	unseen, sim

Scenarios. An episode lasts 120 s and consists of four 30 s segments, the first calm and the others drawn uniformly from three regimes. In a *calm* segment two single-link failures, hard or silent with equal probability, last 6–15 s each. In a *gray* segment one link alternates for the whole segment between loss bursts (85% loss, mean duration 2 s) and quieter periods (5% loss, mean 4 s), while one silent failure occurs elsewhere. In a *storm* segment two links of the same router fail 50–500 ms apart and recover after 5–12 s, emulating a shared-risk event, while another link flaps three to five times. Failure locations favor the 25% of links with the highest edge betweenness with probability 0.6, mirroring the failure-placement model used in the evaluation of DRL-Adapt [12]. Scenario seeds are disjoint across uses: 2001–2002 for calibration, 5000–5019 for reward normalizers, 7000–7003 for the static-configuration search, 10^5 and above for training, and 1001–1008 for testing, of which 1001–1003 are replayed on the testbed.

Baselines. Static configurations are *Default* (RFC 2328 and FRR defaults: 10 s/40 s, SPF 0/50/5000 ms, LSA throttle 5 s), *Fast* (1 s/4 s, default throttles), *Aggressive* (0.25 s/1 s fast-hello, SPF 0/10/1000 ms, no LSA throttle), *Conservative* (5 s/20 s, SPF 50/200/5000 ms), *Damped* (1 s/4 s, SPF 200/1000/10000 ms; simulation only) and *Tuned* (0.33 s/1 s, SPF 50/200 ms, LSA throttle 100 ms). *Best-static* is the best of 81 FRR-feasible settings, nine detection pairs crossed with three SPF and three LSA throttle settings, found by grid search in the calibrated simulator on separate scenarios. Adaptive baselines are a rule-based *Heuristic* that shortens detection after failures and lengthens it when Hello loss indicates a lossy link (simulation only), and the DRL-Adapt agent trained in L0 (the idealized simulator) or in L2. All adaptive policies start from the Fast configuration.

Metrics. Besides the episode cost J we report disruption D (black-holed demand × seconds per episode), the restoration time of each failure (until all demand it disrupted is delivered again), route churn (FIB updates for loopback destinations), OSPF packets sent, spurious adjacency drops (an adjacency going down while its link is physically up) and the number of reconfigurations.

Platform. All experiments run on a single-vCPU Firecracker virtual machine with 4 GB of memory and Linux 6.18. Emulated episodes run in real time; the simulator and the NumPy implementation of the agents run on the same CPU, where training one agent takes about ten minutes.

V. Results

A. Simulator Fidelity

Table 3FIDELITY WITH RESPECT TO THE FRR TESTBED (30 TEST EPISODES, 5 STATIC CONFIGURATIONS, 2 TOPOLOGIES)

Level	err. J	err. D	err. churn	err. pkts	ρ_J	hard rest. (s)
L0 idealized	62.7%	165.7%	52.5%	26.8%	0.57	2.284
L1 FRR-faithful	12.1%	50.2%	15.9%	8.5%	0.90	0.000
L2 calibrated	12.6%	50.4%	16.0%	9.5%	0.90	0.026
FRR testbed	–	–	–	–	–	0.035

Errors are mean absolute relative errors per episode; ρ_J is the Spearman correlation of per-episode cost; the last column is the median restoration time of hard failures under the Fast configuration.

Every emulated static-configuration episode was replayed three times at each simulator level. Table III summarizes the agreement on the test scenarios. The idealized level L0 misestimates the episode cost by 62.7% and the disruption by 165.7% on average, and its per-episode costs correlate only weakly with the emulated ones ($\rho = 0.57$). The FRR-faithful level reduces the cost error to 12.1% and the error on OSPF packet counts to 8.5%. Calibration leaves these aggregate errors essentially unchanged (12.6% for L2), because episode cost is dominated by silent failures and route churn, but it fixes the timing of fast events. Fig. 2 shows the restoration-time distributions: hard failures are restored after a median of 35 ms on the testbed and 26 ms in L2, whereas L1 restores them instantaneously and L0, which must wait for the Dead interval, needs 2.28 s. Silent failures are reproduced well by all levels (median 3.38 s on the testbed and 3.18 s in L2 under Fast), since they are dominated by the Dead interval.

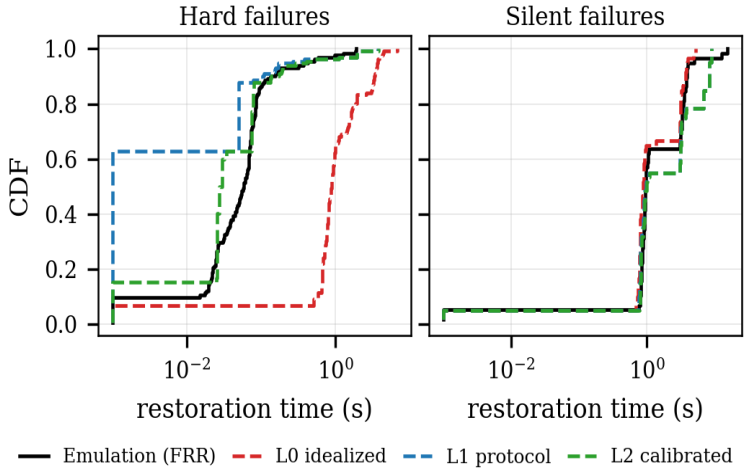


Figure 2. Restoration time of hard (left) and silent (right) failures on the testbed and at the three simulator levels (Fast, Aggressive and Tuned configurations, test scenarios).

Fig. 3 compares per-episode costs. L0 overestimates the cost of slow-detection configurations by up to a factor of two, because carrier-detected failures are charged a full Dead interval, and it is blind to SPF and LSA throttling: in the static grid search, all throttle settings at the best detection pair tie in L0 ($J = 0.68$), whereas L1 and L2 single out the 5 s LSA throttle. Two limitations remain visible. Spurious adjacency drops on gray links are reproduced poorly at every level (mean absolute error of 1.70 drops per episode for L2), and the ranking of the five hand-picked configurations is similar at all levels (Spearman 0.90, 0.80 and

0.80 for L0, L1 and L2), so fidelity matters more for absolute predictions and for knobs the idealized model ignores than for ordering coarse alternatives.

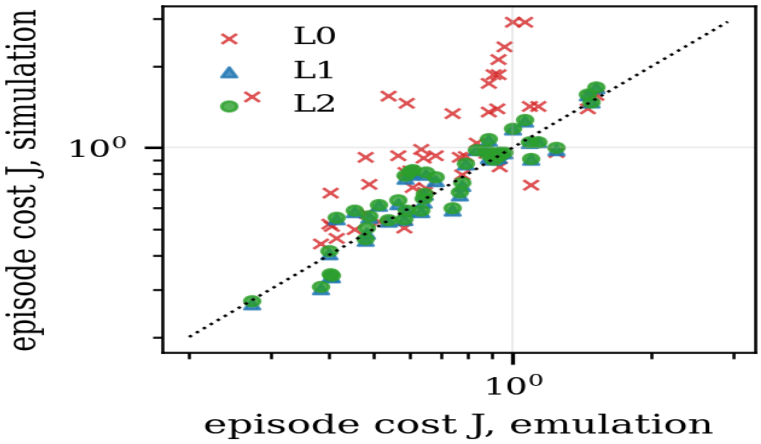


Figure 3. Per-episode cost J in simulation versus emulation for the three simulator levels (calibration and test episodes). The dotted line marks equality.

B. Deployment on the FRR Testbed

Table 4RESULTS ON THE FRR TESTBED (MEAN OVER THREE TEST SCENARIOS)

Policy	J Abil.	J AT&T	D Abil.	D AT&T	Churn	Pkts (k)	Spur.
Default	0.874	0.999	8.11	1.37	289	3.3	1.0
Fast	0.792	0.714	4.94	0.52	834	12.4	6.2
Conservative	0.962	1.043	8.93	1.42	316	4.0	2.3
Aggressive	**0.683**	0.706	1.65	0.17	1104	39.0	12.0
Tuned	0.802	0.674	3.72	0.20	1030	30.0	11.3
Best-static	0.775	**0.486**	3.69	0.13	862	21.5	17.0
DRL-Adapt (L0)	0.792	0.490	4.10	0.13	854	20.6	16.2
DRL-Adapt (L2)	0.762	0.494	3.48	0.14	883	20.9	15.5
Branching (L2)	0.872	0.599	4.82	0.31	775	28.1	9.2

J: episode cost (lower is better); D: black-holed demand × s; churn, OSPF packets and spurious adjacency drops are averaged over both topologies. Bold: lowest J per topology.

Table IV reports the testbed results. The lowest cost is obtained by Aggressive on Abilene (J = 0.683) and by Best-static on AT&T (J = 0.486). Averaged over both topologies and relative to the RFC defaults, the DRL-Adapt agent trained in the idealized simulator lowers the cost by 31.6%, the same agent trained in the calibrated simulator by 32.9%, our branching agent by 21.5%, and the Best-static configuration by 32.7%. The DRL-Adapt gains over default timers are consistent in magnitude with the 32.7% convergence-time reduction that its authors reported against static configurations in simulation [12]; the metric here is the traffic-weighted episode cost J rather than convergence time, but the comparison shows that the re-implemented agent carries its benefit from simulation onto a production routing stack. Relative to Best-static, the DRL-Adapt agent trained in L0 is 1.7% worse, the one trained in L2 0.4% better, and the branching agent 16.6% worse. The paired per-scenario differences of the DRL-Adapt agents (L0, L2) from Best-static are +0.017 ± 0.010 and −0.013 ± 0.021 on Abilene, and +0.004 ± 0.023 and +0.008 ± 0.017 on AT&T (mean ± s.d. of J), i.e., within the scenario-to-scenario variability, whereas the branching agent is

consistently worse (+0.096 ± 0.050 and +0.112 ± 0.027). Abilene is the noisier topology: the static Aggressive setting wins there on average, but its paired difference to Best-static is −0.092 ± 0.241, and on AT&T it is 0.219 worse.

The learned policies ran on the testbed without manual intervention: DRL-Adapt's stabilization interval kept Hello/Dead changes rare, so its actuation design carried over to the real stack with only the projection onto FRR's timer lattice. However, the DRL-Adapt agents behave almost statically: within the first decisions they move to a 1 s Dead interval (fast-hello mode with 0.5 s Hellos, the detection setting that the static search also selects) and keep it during 91.0% (L0-trained) and 95.8% (L2-trained) of all decision steps, with 1.5 and 1.0 Hello/Dead changes per episode; they keep FRR's default 5 s LSA throttle because they cannot change it. The branching agent reconfigures 21.0 times per episode, 9.3 of which change the Hello/Dead pair, and keeps the 5 s LSA throttle 66.7% of the time. Fig. 4 shows one AT&T episode.

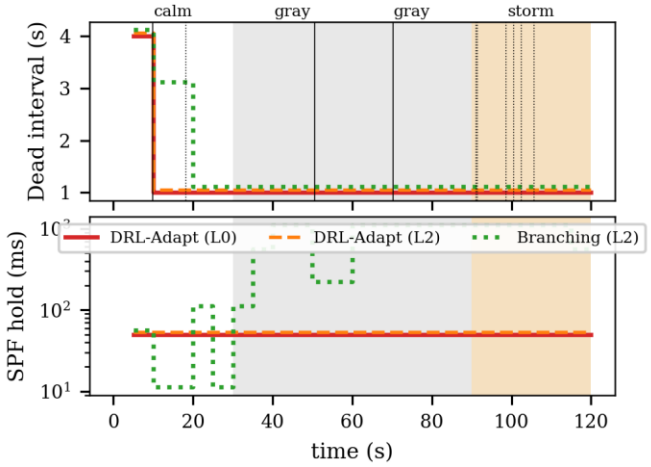


Figure 4Dead interval and SPF hold time selected on the FRR testbed during one AT&T test episode (seed 1002); lines are offset slightly for visibility. Shading marks gray and storm segments; vertical lines mark hard (dotted) and silent (solid) failures.

C. Simulation at Scale and Ablations

Table 5EPISODE COST J IN THE CALIBRATED SIMULATOR (8 TEST SCENARIOS PER TOPOLOGY)

Policy	**Abil.**	**AT&T**	**GÉANT**	**Exodus***
Default	1.179	1.445	1.261	1.538
Fast	0.651	0.729	0.696	0.806
Tuned	0.611	0.632	0.682	0.703
Best-static found in L0	0.674	0.628	0.647	0.705
Best-static	**0.534**	**0.489**	**0.499**	**0.515**
Heuristic	1.035	1.314	1.770	2.080
DRL-Adapt (L0)	0.585	0.522	0.558	0.641
DRL-Adapt (L2)	0.590	0.522	0.523	0.550
DRL-Adapt (L2), seed 2	0.590	0.522	0.523	0.550
Branching (L2)	0.682	0.605	0.773	0.650
Branching (L2), seed 2	0.726	0.691	0.782	0.770
Branching (L1)	1.359	1.719	3.033	5.003
Branching (L2, no DR)	0.743	0.615	0.589	0.661
Branching, Best-static start	0.678	0.587	0.580	0.606
Branching, gated	0.695	0.608	0.589	0.592

*Exodus (79 routers) is not used for training or for the static search. Bold: lowest J per topology.

Table V extends the comparison to GÉANT and to the larger, unseen Exodus topology. The lowest cost is achieved by Best-static on all four topologies (mean J = 0.509). The static configuration selected in the idealized simulator differs from Best-static only in its LSA throttle (100 ms instead of 5 s, an arbitrary choice among ties in L0) and costs 0.663 on average, 30.2% more, which isolates the value of a simulator that models LSA pacing; the L1 search selects the same configuration as L2. Training the DRL-Adapt agent in the calibrated instead of the idealized simulator lowers its mean cost from 0.577 to 0.546, mostly on the larger topologies, but it stays above Best-static. Calibration matters far more for the branching agent. Trained in the uncalibrated L1 simulator, where reconfiguration is instantaneous, it changes the Hello/Dead pair 8.8 times per AT&T episode; under the measured push skew, these changes repeatedly break adjacencies through timer mismatches, and its mean cost rises to 2.779. The rule-based Heuristic suffers from the same effect (8.9 changes per AT&T episode, mean cost 1.550). Trained in L2, the branching agent reaches 0.687 on the training topologies, 0.649 without dynamics randomization and 0.733 with a second training seed, so the differences between its variants are of the order of the seed variability. Starting it from Best-static lowers its mean cost from 0.677 to 0.613, and DRL-Adapt's confidence gate with $\delta_Q = 0.3$, selected on validation scenarios, leaves it essentially unchanged at 0.621. Neither variant reaches the static optimum.

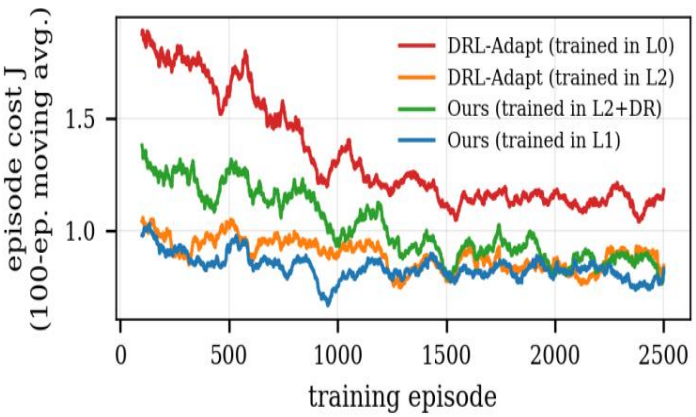


Figure 5Training curves (episode cost in the respective training simulator, 100-episode moving average; costs are not comparable across simulators).

Fig. 5 shows that all agents learn quickly to leave the slow detection settings, which accounts for most of their gain over the defaults, and then plateau. The regime analysis explains why little headroom remains: per-regime optimal static settings differ, but under the 10 s stabilization interval and about 1 s of actuation skew an agent can switch only a few times per 30 s regime, while the 5 s LSA throttle already damps flapping adjacencies in gray and storm periods without any adaptation.

VI. Discussion and Conclusion

Limitations. The testbed runs on a single virtual CPU without link propagation delay, so processing delays and actuation skew are larger and propagation smaller than in a hardware network; calibration accounts for both, but absolute timing results are specific to this platform. Loss is produced by iptables rather than by a queueing model, gray failures are therefore idealized, and spurious adjacency drops remain the least faithful aspect of every simulator level. We evaluated two topologies on the testbed and three scenarios per topology, which limits statistical power; the simulator results over eight scenarios and four topologies point in the same direction. Finally, the agents act on network-wide timers; per-link or per-region control and BFD-based detection may open more room for adaptation.

Conclusion. We moved DRL-based OSPF convergence optimization from simulation to an FRRouting emulation testbed. An idealized timer-centric simulator misrepresents how a real stack detects hard failures and paces updates, and is blind to SPF and LSA throttling; an FRR-faithful simulator calibrated from emulation traces reproduces per-episode cost within about 12.6% and hard-failure restoration within

milliseconds. A re-implemented DRL-Adapt agent transfers to the real stack and improves substantially on default timers, but it does so by settling on a near-static operating point, and neither it nor our feasibility-aware branching agent improves on a static configuration found by searching the calibrated simulator beyond scenario-to-scenario variability; in simulation over four topologies, including an unseen one, that static configuration has the lowest cost. For practitioners, the most effective use of simulation is therefore calibrated configuration search, with learned online control reserved for settings that offer real headroom for adaptation. Our testbed, simulator, calibration traces and agents are designed to make such questions testable on real routing software.

REFERENCES

J. Moy, "OSPF version 2," IETF, RFC 2328, Apr. 1998, doi: 10.17487/RFC2328.

M. Goyal, M. Soperi, E. Baccelli, G. Choudhury, A. Shaikh, H. Hosseini, and K. S. Trivedi, "Improving convergence speed and scalability in OSPF: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 443–463, 2012, doi: 10.1109/SURV.2011.011411.00065.

B. Decraene, S. Litkowski, H. Gredler, A. Lindem, P. Francois, and C. Bowers, "Shortest path first (SPF) back-off delay algorithm for link-state IGPs," IETF, RFC 8405, Jun. 2018, doi: 10.17487/RFC8405.

P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 35–44, Jul. 2005.

A. Shaikh, A. Varma, L. Kalampoukas, and R. Dube, "Routing stability in congested networks: Experimentation and analysis," in *Proc. ACM SIGCOMM*, Stockholm, Sweden, 2000, pp. 163–174.

A. Basu and J. Riecke, "Stability issues in OSPF routing," in *Proc. ACM SIGCOMM*, San Diego, CA, USA, 2001, pp. 225–236.

F. Zhou, “Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders,” Innovation and Technology Studies, vol. 3, no. 1, pp. 27–41, 2026.

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

J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Adv. Neural Inf. Process. Syst. 6 (NIPS)*, 1993, pp. 671–678.

A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proc. 16th ACM Workshop Hot Topics Netw. (HotNets)*, Palo Alto, CA, USA, 2017, pp. 185–191, doi: 10.1145/3152434.3152441.

Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, 2018, pp. 1871–1879, doi: 10.1109/INFOCOM.2018.8485853.

J. Chen, X. Han, X. Zhang, and Y. Yang, “Exact concept-level explanations for early detection of financial misstatements,” Eurasia Journal of Science and Technology, vol. 5, no. 2, pp. 34–48, 2023.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, 2017, doi: 10.1109/IROS.2017.8202133.

X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Brisbane, QLD, Australia, 2018, doi: 10.1109/ICRA.2018.8460528.

FRRouting Project, "FRRouting (FRR) protocol suite," version 8.4.4. [Online]. Available: https://frrouting.org

S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011, doi: 10.1109/JSAC.2011.111002.

N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, USA, 2002, pp. 133–145.

S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable experiments for performance evaluation of traffic-engineering algorithms," arXiv:1710.08665, 2017.

D. Katz and D. Ward, "Bidirectional forwarding detection (BFD)," IETF, RFC 5880, Jun. 2010, doi: 10.17487/RFC5880.

P. Huang, C. Guo, L. Zhou, J. R. Lorch, Y. Dang, M. Chintalapati, and R. Yao, "Gray failure: The Achilles' heel of cloud-scale systems," in *Proc. 16th Workshop Hot Topics Oper. Syst. (HotOS)*, Whistler, BC, Canada, 2017, pp. 150–155, doi: 10.1145/3102980.3103005.

D. Teng, "TRIAD: Deadline- and reliability-aware scheduling for post-earthquake structural assessment workflows," World Journal of Engineering Research, vol. 3, no. 6, pp. 55–66, 2025.

V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, 2016, pp. 2094–2100, doi: 10.1609/aaai.v30i1.10295.

Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2016, pp. 1995–2003.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, 2016.

A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, 2018, doi: 10.1609/aaai.v32i1.11798.

B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM Workshop Hot Topics Netw. (HotNets)*, Monterey, CA, USA, 2010.

N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Nice, France, 2012, pp. 253–264, doi: 10.1145/2413176.2413206.

P. Almasan *et al.*, "Network digital twin: Context, enabling technologies, and opportunities," *IEEE Commun. Mag.*, vol. 60, no. 11, pp. 22–27, Nov. 2022, doi: 10.1109/MCOM.001.2200012.

X. Yue and R. Zhu, "An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study," World Journal of Management Science, vol. 2, no. 1, pp. 63–72, 2024.

J. Ding and D. Teng, "A Risk-Aware Digital Twin Framework for Release-Readiness Validation of Multi-Tenant Zero Trust SASE Systems," WEP Transactions on Computer Science and Intelligent Systems Research, vol. 13, pp. 247–259, 2026.