INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY TRENDS

Vol. 6 No. 02 (2026)

e.ISSN: 2409-5222

https://ijictt.ictt.info

MARL-Converge: Decentralized Multi-Agent Reinforcement Learning for Failure-Aware Routing Convergence

David Clarke

Department of Computer Science, Rutgers University, New Brunswick, NJ, USA.

Mei Huang

Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ, USA.

Jonas Eriksen

School of Communication and Information, Rutgers University, New Brunswick, NJ, USA.

Abstract

Routing failures demand rapid control-plane convergence, yet uniformly aggressive detection and flooding increases signaling load and can induce unstable parameter oscillations. This paper presents MARL-Converge, a decentralized multi-agent reinforcement learning framework in which every router selects a local conservative, balanced, or aggressive control profile from router-local and one-hop telemetry. A parameter-shared dueling Double Deep Q-Network is trained with shared experience and a global-dominant team reward, while execution remains decentralized. A failure-aware action mask prevents routers without local evidence from blindly escalating to the aggressive profile. We implement a reproducible discrete-time link-state convergence proxy with independent topology views, stochastic failure detection and link-state advertisement diffusion, shortest-path recomputation, and forwarding-loop/drop checks. Evaluation comprises 600 matched episodes on the real Abilene and held-out AARNet topologies from the Internet Topology Zoo. Relative to a static balanced profile, MARL-Converge reduces convergence steps by 19.0% and modeled packet loss by 83.6%; the 6.1% overhead reduction is directionally favorable but not significant at the 0.05 level. Relative to a local heuristic, it is 9.3% faster, uses 14.0% fewer control units, and makes 56.4% fewer profile switches. It is 7.6% slower than always-aggressive control but cuts overhead by 35.4%. These results establish a measured speed-stability-efficiency trade-off without claiming physical-router timing equivalence.

Keywords *:decentralized control, failure-aware routing, multi-agent reinforcement learning, routing convergence, safe action masking*

I. INTRODUCTION

Fast routing convergence is a prerequisite for dependable cloud, enterprise, and wide-area services. After a link or router failure, neighboring devices must detect the event, propagate updated topology information, recompute paths, and stabilize forwarding state. OSPF formalizes this process through distributed link-state databases and shortest-path computation [2], whereas BGP exchanges inter-domain reachability under policy constraints [3]. Operators can accelerate failure detection with mechanisms such as Bidirectional Forwarding Detection (BFD) [4] and protect traffic locally through IP Fast Reroute (FRR) [5]. Nevertheless, the end-to-end convergence process still exposes a fundamental trade-off:

conservative control reduces routine signaling, while aggressive control shortens detection and dissemination at the cost of higher control load and potentially unstable reactions. Classical convergence studies documented delayed BGP recovery [1], the factors governing sub-second IGP convergence [8], OSPF stability limits [9], and the unintended effects of route-flap damping [10]. These works motivate careful timer and control-plane tuning, but a single configuration cannot be optimal for every topology, failure location, and transient event rate. The recently proposed DRL-Adapt framework provides an important demonstration that event-driven deep reinforcement learning can adapt routing parameters and improve the speed-overhead balance in large networks [7]. Its event-stream formulation, safety constraints, and positive simulation evidence establish a strong foundation for learning-based convergence control. The next architectural question is how to preserve that adaptivity when a global controller is unavailable, delayed, or undesirable. A routing failure is inherently local at first and becomes network-wide only through protocol dissemination. This observation favors a multi-agent formulation: each router acts on local evidence, while all routers share the objective of rapidly reaching a consistent state. However, naive independent learning suffers from non-stationarity, and unconstrained decentralized escalation can recreate the control storm that adaptive control is meant to avoid. MARL-Converge addresses both issues through parameter sharing and a failure-aware action mask. Homogeneous router agents use the same compact Q-network, which makes the number of learned parameters independent of topology size. Training pools transitions across routers and uses a team-dominant reward; deployment requires only local and one-hop telemetry. The contributions are fourfold. First, routing convergence is formulated as a cooperative partially observed control problem with independent router topology views and decentralized actions. Second, a parameter-shared dueling Double DQN is combined with one-hop telemetry and a hard failure-aware action mask that blocks blind aggressive escalation. Third, a transparent link-state convergence proxy is implemented in executable Python, including stochastic detection, queued LSA processing, controlled flooding, SPF recomputation, route-change accounting, and forwarding drop/loop checks. Fourth, matched-seed experiments on two real Internet Topology Zoo networks, synthetic scale stress tests, paired statistical tests, and safety/telemetry stress analyses quantify both gains and trade-offs. All reported numbers are generated by the supplied code and raw CSV files; convergence is measured in simulator decision steps rather than seconds.

II. RELATED WORK

A. Routing Convergence and Failure Recovery

Link-state protocols seek a consistent topology database across routers, but detection delays, advertisement pacing, processing limits, and transient forwarding dependencies determine the observed convergence interval. OSPF [2] specifies the distributed database and SPF behavior, while BFD [4] separates rapid path-failure detection from the routing protocol. FRR mechanisms [5], [6] reduce packet loss during global reconvergence by installing local repair paths, but they do not remove the need for stable network-wide state. Empirical and analytical work has shown that convergence can be delayed by path exploration [7], that sub-second IGP recovery requires coordinated optimization across detection, flooding, SPF, and FIB stages [8], and that aggressive reactions may interact with stability mechanisms in undesirable ways [9], [10]. MARL-Converge complements these protocol mechanisms: it does not replace SPF or FRR, but selects an abstract local control profile that governs the responsiveness-cost trade-off in the modeled convergence process.

B. Learning-Based Networking and Multi-Agent Control

Q-routing embedded a reinforcement learner at each switching node and demonstrated the value of local adaptive decisions [23]. Subsequent work applied deep reinforcement learning to resource management [24], traffic engineering [25], software-defined routing [26], and model-free network control [30]. RouteNet showed that graph neural models can generalize network performance prediction across topology and routing conditions [27], while knowledge-defined networking articulated a broader architecture for combining telemetry, analytics, and control [28]. DRL-Adapt specifically advances convergence optimization by learning event-driven routing-parameter adjustments [7]. MARL-Converge extends this line toward decentralized execution and a topology-size-independent shared policy. Cooperative multi-agent reinforcement learning is commonly modeled through Markov games [12] or decentralized partially observable Markov decision processes [13]. Centralized-training/decentralized-execution methods include value factorization in QMIX [17], multi-agent actor-critic learning [18], and counterfactual credit assignment [19]. MAPPO demonstrates that carefully implemented shared-policy optimization can be competitive across cooperative benchmarks [20], and fully decentralized actor-critic methods formalize learning over networked agents [21]. Our setting differs in three practical respects: router agents are homogeneous, the action set is small and discrete, and execution must remain inexpensive. We therefore use a parameter-shared dueling Double DQN [14]–[16] with a global-dominant reward rather than a centralized critic or mixing network. Safe reinforcement learning motivates the use of externally enforced constraints when exploration or deployment can violate operational requirements [22].

III. PROBLEM FORMULATION

A. Decentralized Link-State Model

Let G=(V,E) be an undirected connected network before failure. A failure removes either one non-bridge link or one non-articulation router so that post-failure reachability remains meaningful. Each active router i maintains a local topology view G_i(t). Immediately after the physical failure, most G_i(t) are stale; incident routers must detect the failure, informed routers must process and flood an LSA, and each newly informed router recomputes its shortest-path next hops. Convergence is reached at the first decision epoch T for which every active router has incorporated the failure and no queued or in-flight LSA remains. This operational definition avoids assigning fabricated wall-clock seconds to the abstract simulator. At decision epoch t, router i observes a 20-dimensional vector o_i(t). The vector contains local activity, normalized degree and betweenness, incident-failure evidence, whether the failure is known or queued, recent incoming and local update indicators, normalized episode time, four one-hop aggregates, normalized degree density, a one-hot encoding of the previous profile, cumulative route-change and control-load summaries, and the fraction of active routers not yet informed. No router identifier or full global adjacency matrix is supplied. One-hop aggregates are computed from protocol-neighbor telemetry, which is available without a centralized actor.

B. Actions, Team Objective, and Safety

Each active router selects a_i(t) from three profiles: conservative, balanced, and aggressive. The profiles abstract a bundle of operational choices affecting failure-detection probability, queued-LSA processing, flooding pace, and periodic control cost. They are intentionally not labeled as specific vendor timer values. The goal is to minimize the convergence interval and transient forwarding impairment without paying the always-aggressive control cost or inducing excessive profile oscillation.

$$J(\pi) = E_{\pi}\left[\sum_t \gamma^t r_t \right] \quad (1)$$

The global reward combines newly informed routers, the unresolved fraction, packet drops and loops observed on sampled flows, per-step control cost, and profile switching. A small local term rewards a router when it first incorporates the failure and charges its own control profile. The implemented reward is 85% global and 15% local, preserving cooperative pressure while retaining a direct local learning signal.

$$r_t = 2.8\,P_t - 0.55\,U_t - 1.35\,L_t - 0.12\,C_t - 0.04\,S_t \quad (2)$$

A hard action mask enforces the principal safety rule. The aggressive profile is available only after local evidence exists: the router is incident to the failure, has a queued LSA, received a recent update, or already knows the failure. A failed router is forced to the conservative profile. Thus, safety does not depend on the learned Q-values. The mask limits blind network-wide escalation while still permitting a rapid reaction along the actual dissemination frontier.

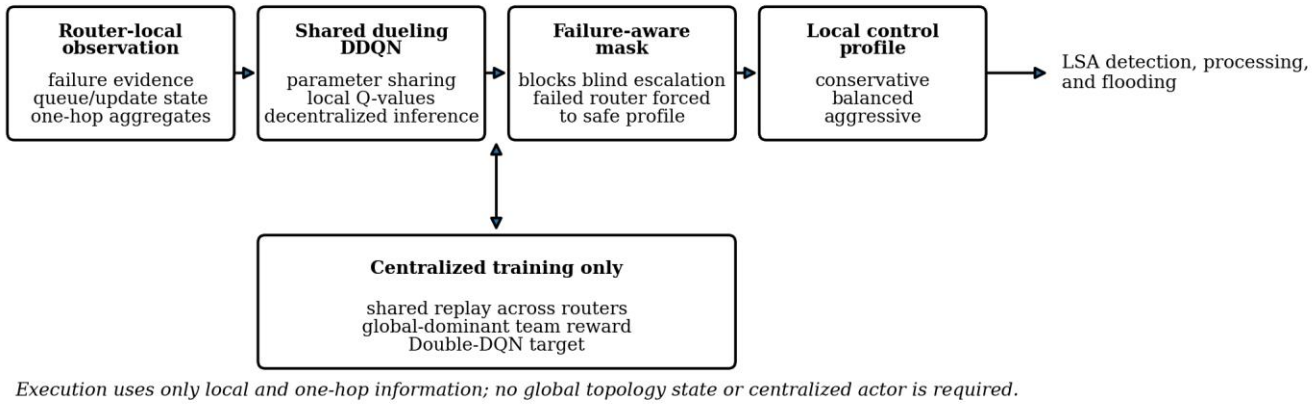


Fig. 1. MARL-Converge architecture. Shared training is centralized only during learning; each router executes the common policy from local and one-hop observations.

IV. MARL-CONVERGE METHOD

A. Parameter-Shared Dueling Double DQN

All active routers share one Q-network $Q(o,a;\theta)$. Parameter sharing exploits router homogeneity, pools experience from many agents, and keeps model size fixed as $|V|$ changes. The feature extractor uses two fully connected ReLU layers. Dueling heads separately estimate the local state value $V(o)$ and action advantages $A(o,a)$, then combine them as

$$Q(o,a;\theta) = V(o;\theta_v) + A(o,a;\theta_a) - \text{mean}_b\, A(o,b;\theta_a). \quad (3)$$

Double Q-learning reduces maximization bias by choosing the next action with the online network and evaluating it with a target network [15]. For transition (o,a,r,o',d), the target and Huber objective are

$$y = r + \gamma(1-d)\, Q(o', \arg\max_b Q(o',b;\theta); \bar{\theta}), \quad (4)$$

$$L(\theta) = E[\text{Huber}(y - Q(o,a;\theta))]. \quad (5)$$

Training uses a shared uniform replay buffer, AdamW, gradient clipping, a target-network copy every 300 optimizer updates, and epsilon-greedy exploration decayed from 1.0 to 0.04. This implementation deliberately avoids claiming prioritized replay because the supplied code samples uniformly. At execution, epsilon is zero and each router independently chooses the highest-valued action permitted by its mask.

B. Event Dynamics and Forwarding Validation

Each episode begins with a centrality-biased link or node failure. Incident routers detect the event stochastically according to their profile. An informed router floods at most one LSA per directed adjacency; delivery occurs in the next decision epoch, and the receiving router may need additional epochs to process the queued update. Upon first learning the failure, a router removes the failed component from $G_i(t)$, recomputes a single-source shortest-path tree, and records every next-hop change. This produces independent transient forwarding tables rather than instantly replacing all routes with the post-failure optimum. Forwarding quality is measured on sampled source-destination flows. A flow is traced through the next hop selected by the current local table at each visited router. A missing next hop, an inactive next hop, or an edge removed from the physical graph counts as a drop; revisiting a router counts as a loop. Packet-loss and loop rates are normalized by the number of hop checks,

making them comparable across episodes. These are modeled forwarding indicators rather than packet-level measurements from physical routers.

TABLE I. IMPLEMENTED PROFILE PARAMETERS

Profile	p_detect	p_process	p_flood	Hello cost
Conservative	0.22	0.35	0.30	0.35
Balanced	0.52	0.70	0.72	0.85
Aggressive	0.90	0.98	1.00	2.10

Table I lists the exact simulator constants. They encode a monotone speed-cost relationship and are exposed in env.py. The experimental claim is therefore about policies over these profiles in the stated proxy, not about a universal mapping from a profile to OSPF or BGP seconds.

C. Shared Training and Decentralized Execution

Every active router contributes one transition to a common replay buffer after each environment step. A transition stores the current local observation, the executed profile, the router reward, the next observation, the terminal flag, and both current and next action masks. Mini-batches of 128 transitions are sampled uniformly every third environment step. The online network uses AdamW with learning rate 3×10^−4, discount factor 0.97, Huber loss, and a gradient-norm limit of 5. The target network is copied every 300 optimizer updates, and the replay capacity is 120,000 transitions. These values are fixed in the released scripts rather than selected after observing the test results. The implemented network has 9,924 trainable parameters: a 20-to-64-to-64 shared feature extractor, a 32-unit value head, and a 32-unit advantage head with three outputs. Consequently, the learned parameter count is independent of the number of routers. At deployment, router i computes one forward pass and applies its local mask; no centralized state vector, joint action enumeration, or online inter-router gradient exchange is required. The total inference work per decision epoch is therefore linear in the number of active routers, while the one-hop aggregates are obtained from the same neighbor state that supports link-state dissemination.

V. EXPERIMENTAL DESIGN

A. Topologies, Training, and Evaluation

Real-network tests use Abilene (11 routers, 14 links) and AARNet (19 routers, 24 links), transcribed from the official Internet Topology Zoo GraphML records [11]. Abilene appears once every four training episodes. AARNet is never used in training and is therefore a zero-shot topology test. The remaining training episodes use seeded Watts-Strogatz graphs with 20 or 30 nodes and Barabasi-Albert graphs with 24 nodes. Two independent shared policies are trained for 300 episodes with seeds 7 and 11; evaluation alternates the two models across matched random seeds. The principal evaluation

contains 2 topologies × 2 failure types × 5 methods × 30 matched seeds = 600 complete episodes. Failure types are single-link and single-router failures. Link selection excludes bridges when possible and favors high edge betweenness with probability 0.60; router failures exclude articulation points when possible and are sampled in proportion to normalized betweenness plus a small floor. Each episode has a 38-step timeout and 45 sampled flows. All methods receive identical topology, failure type, and random seed within a matched comparison.

B. Baselines and Metrics

The baselines are Static-Conservative, Static-Balanced, Static-Aggressive, and Local-Heuristic. Static policies apply one profile to every active router, subject to the failed-router safety projection. The heuristic selects aggressive control at routers with incident, queued, or recent-update evidence; informed routers outside that frontier use balanced control, and all others use conservative control. It is a strong decentralized hand-crafted comparator because it has direct access to the same local evidence used by the learned policy. Metrics are convergence steps, modeled control units, packet-loss rate, loop rate, route changes, and profile switches. Lower is better for every reported metric. Paired t-tests compare MARL-Converge against each baseline across 120 matched observations (2 topologies × 2 failure types × 30 seeds). The use of matched seeds reduces variance from stochastic detection and failure location. A separate scale stress test uses fixed held-out Watts-Strogatz graphs with 20, 30, and 40 nodes, 20 seeds per failure type, and four methods. The experiment artifacts include raw rows, grouped summaries, p-values, trained weights, and the exact scripts used to generate them.

C. Reproducibility and Statistical Protocol

The main results are regenerated by run_real_experiments.py from two stored checkpoints. To avoid selecting the more favorable checkpoint, matched evaluation seeds alternate deterministically between seed-7 and seed-11 policies. Within each topology-failure-seed tuple, every method receives a newly constructed environment with the same seed, so the selected failed component and stochastic event stream are aligned as far as the method-dependent actions permit. Means are reported with sample standard deviations. The significance values are two-sided paired t-tests over the 120 aligned observations and are reported without multiplicity correction; they should therefore be read as transparent comparative diagnostics rather than confirmatory family-wise inference. The recorded environment used Python 3.13.5, PyTorch 2.10.0 (CPU), NetworkX 3.6.1, NumPy 2.3.5, pandas 2.2.3, and SciPy 1.17.0. The package includes the exact GraphML-derived edge lists used for Abilene and AARNet, fixed synthetic-graph seeds, raw episode-level CSV files, grouped summaries, statistical outputs, and plotting scripts. No performance number is manually inserted into a results file; the paper-generation script reads the saved CSV summaries directly for the main tables.

VI. RESULTS

A. Overall Speed-Overhead Trade-off

Table II summarizes all 120 episodes per method. MARL-Converge reaches convergence in 9.32±2.64 decision steps. Compared with Static-Balanced, this is a 19.0% reduction (paired $p=1.00\times10^{-14}$), while the packet-loss rate falls from 0.4667% to 0.0767% (83.6%, $p=3.19\times10^{-12}$). Control use declines by 6.1%, but this difference is not statistically significant at 0.05 ($p=0.096$); it is reported as directional rather than conclusive. Compared with Local-Heuristic, MARL-Converge is 9.3% faster ($p=1.11\times10^{-7}$), uses 14.0% fewer control units ($p=5.92\times10^{-5}$), reduces loop rate by 15.5% ($p=3.73\times10^{-4}$), and makes 56.4% fewer profile switches. Static-Aggressive is the fastest method at

8.66 steps. MARL-Converge is 7.6% slower (p=2.31×10^−6), but it reduces control use by 35.4% (p=3.87×10^−24), packet loss by 4.0% (p=0.0146), and loops by 6.0% (p=0.00116). This is the intended operating point: the learned policy does not attempt to win convergence speed at any cost. The heuristic has a slightly lower packet-loss rate than MARL-Converge (0.0696% versus 0.0767%, p=0.0044), an absolute difference of 0.0072 percentage points; this unfavorable result is retained to avoid selective reporting.

TABLE II. OVERALL RESULTS ACROSS 120 MATCHED EPISODES PER METHOD

Method	Conv. steps	Control units	Loss (%)	Loops (%)	Switches
Static-Conservative	26.06±7.83	385.0±188.7	0.7496	0.5485	14.5
Static-Balanced	11.50±3.34	410.1±199.6	0.4667	0.4121	0.0
Static-Aggressive	8.66±2.33	595.5±291.2	0.0800	0.4734	14.4
Local-Heuristic	10.27±2.97	447.8±204.5	0.0696	0.5267	38.1
MARL-Converge	9.32±2.64	384.9±221.3	0.0767	0.4450	16.6

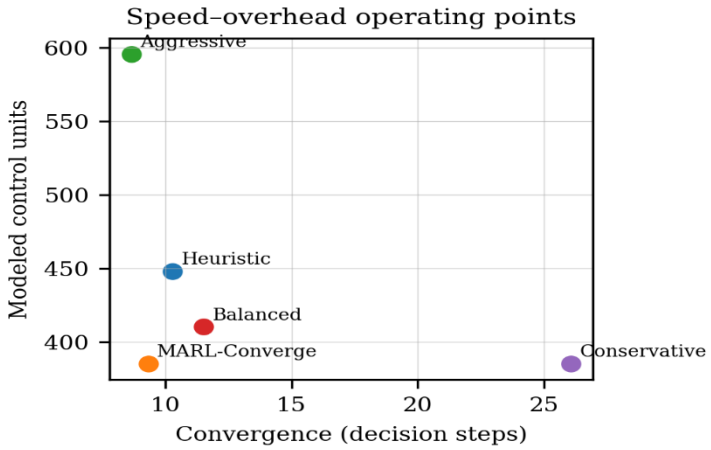


Fig. 2. Mean speed-overhead operating points across both real topologies and failure types.

TABLE III. SELECTED PAIRED COMPARISONS (TWO-SIDED, UNCORRECTED)

Comparator	Metric	Δ MARL	p
Balanced	Convergence	-19.0%	1.00e-14
Balanced	Packet loss	-83.6%	3.19e-12
Aggressive	Convergence	+7.6%	2.31e-06
Aggressive	Control	-35.4%	3.87e-24
Local-Heuristic	Convergence	-9.3%	1.11e-07
Local-Heuristic	Control	-14.0%	5.92e-05

B. Failure-Type and Zero-Shot Topology Results

Figure 3 separates the four real-network scenarios. On Abilene link failures, MARL-Converge requires 8.43±1.38 steps versus 9.70±1.86 for Static-Balanced and 8.73±2.05 for the heuristic. On Abilene router failures it reaches 7.00±1.11 steps, close to the heuristic at 7.13±1.20 and slower than always-aggressive control at 6.50±0.86. The held-out AARNet is more difficult for every method. MARL-Converge still reduces balanced convergence from 14.10 to 11.20 steps for link failures (20.6%) and from 13.20 to 10.63 steps for router failures (19.4%). This zero-shot result indicates that the identity-free observation and shared policy transfer beyond the topology encountered during training.

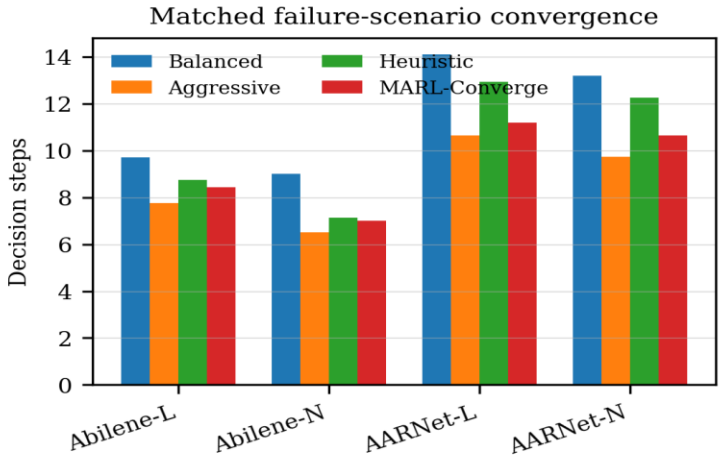


Fig. 3. Mean convergence by real topology and failure type. L denotes link failure and N denotes router failure.

C. Scale Stress Test

The scale stress test in Fig. 4 is not presented as a physical large-router validation; it tests whether the shared policy can be executed without resizing its network. Across fixed held-out graphs with 20, 30, and 40 nodes, MARL-Converge records 8.65, 9.68, and 9.68 mean steps. Static-Balanced records 9.75, 11.45, and 11.08, corresponding to MARL reductions of 11.3%, 15.5%, and 12.6%. Always-aggressive control remains faster but uses substantially more modeled control. The shared network input remains 20 features and three actions at every scale; inference complexity grows linearly only because one forward pass is made per active router.

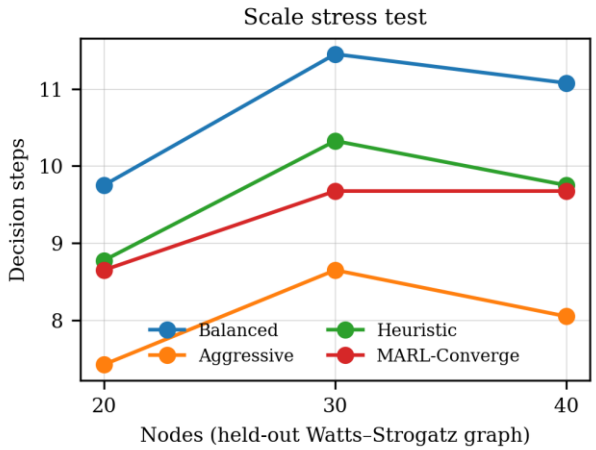


Fig. 4. Held-out synthetic scale stress test, averaged over link and router failures.

D. Telemetry and Safety Stress Analyses

Two stress analyses isolate the operational role of one-hop telemetry and the action mask. A separately trained No-Neighbor-Telemetry policy removes observation dimensions 9–12. Its convergence is similar to the full policy (8.89 versus 9.09 steps on a fresh matched seed set), but profile switches increase from 16.45 to 36.11, a 119.5% rise, and modeled packet loss increases by 11.6%. The result suggests that one-hop aggregates primarily stabilize coordination rather than determine raw speed. The Mask-Disabled stress test evaluates the full trained policy in an environment that permits every active router to choose aggressive control even without evidence. Convergence improves from 9.09 to 8.22 steps, but switches rise by 117.6%, control use rises by 3.9%, and packet loss rises by 14.0%. Because the policy was trained with the mask, this is a deployment stress test rather than a retrained ablation. The result nonetheless clarifies the mask's purpose: it deliberately gives up some speed to suppress network-wide profile churn and blind escalation.

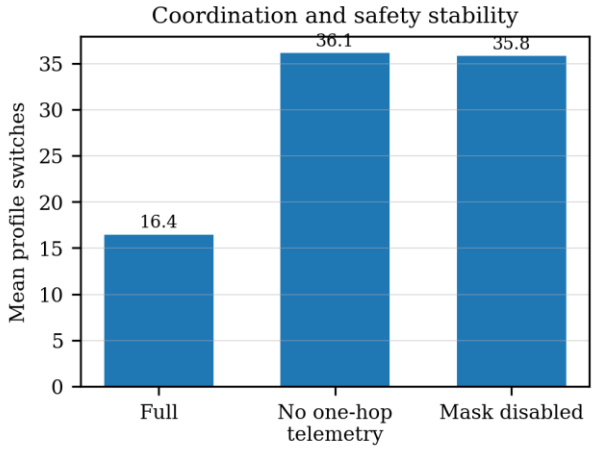


Fig. 5. Mean profile switches in the full method and telemetry/safety stress variants.

E. Learning Behavior and Checkpoint Use

Figure 6 shows the recorded seed-11 training trajectory using a 25-episode moving average. Mean convergence decreases from 10.64 steps in the first 25 episodes to 8.44 in the final 25, while mean episode reward rises from 1.08 to 2.56 and mean modeled control falls from 827.9 to 569.7 units. The curve remains non-monotone because every fourth episode changes graph family and topology size, exploration remains active at ε=0.04, and failures are resampled. The purpose of the plot is therefore to show learning progress under a changing curriculum, not a smooth optimization benchmark.

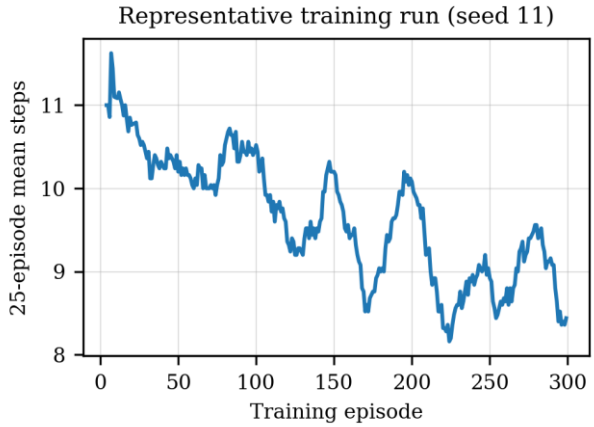


Fig. 6. Representative seed-11 training run; the line is a 25-episode moving average of convergence steps.

Both independent checkpoints are retained. The principal evaluation alternates them across matched seeds rather than selecting the checkpoint with the best validation outcome. This conservative use of checkpoints reduces the risk that the reported aggregate reflects a single fortunate initialization, although two training seeds are insufficient for a full characterization of learning variance. Raw per-episode evaluation rows preserve the checkpoint alternation through the deterministic seed order.

VII. DISCUSSION

MARL-Converge's advantage over Static-Balanced and the heuristic arises from selective temporal escalation. Incident and newly informed routers can choose aggressive behavior while unaffected routers avoid its recurring cost. Parameter sharing lets the same learned mapping respond to different local roles: a high-centrality incident router, a queued boundary router, and an already stabilized router receive different observations even though they use the same weights. The policy therefore

approximates a moving convergence frontier rather than applying one global mode. The comparison with Static-Aggressive is equally important. If the only objective were the minimum number of decision steps in this proxy, always-aggressive control would be preferred. The learned policy instead occupies a lower-overhead point with modestly slower convergence. This measured trade-off is more operationally credible than claiming dominance on every metric. Likewise, the nonsignificant overhead difference against Static-Balanced and the small packet-loss disadvantage against the heuristic are explicitly reported. The statistically supported gains are convergence, overhead versus the heuristic and aggressive baselines, loop reduction versus the heuristic and aggressive baseline, and profile-switch reduction versus the heuristic. The approach is compatible in concept with existing protocol safeguards. A production actuator could map the three profiles to bounded timer, pacing, or CPU-budget configurations, retain BFD and FRR as independent mechanisms, and roll back to a known-safe baseline when monitoring thresholds are exceeded. DRL-Adapt's staged deployment and safety-oriented design provide a positive precedent for this broader operational path [7]. A practical deployment should first run in advisory mode and validate recommendations against FRRouting or vendor-specific behavior before enabling automated changes.

A. Deployment Path and Operational Guardrails

A realistic deployment should separate recommendation, enforcement, and rollback. In a shadow phase, the policy would consume telemetry and emit profiles without changing router configuration; operators would compare these recommendations with daemon logs and maintenance events. A bounded-actuation phase would then permit only prevalidated profile transitions, rate-limit repeated changes, and require minimum evidence persistence. Finally, an automatic rollback rule would restore the balanced profile when convergence exceeds a deadline, control CPU crosses a threshold, telemetry becomes stale, or routers disagree on the failure epoch. This design keeps the learned component outside the protocol correctness boundary: the router still enforces legal configuration ranges and preserves its existing FRR/BFD mechanisms. The present action mask is a simple instance of such an external guardrail. It is intentionally independent of neural confidence, and its stress test exposes the cost of removing it. Stronger deployments could add constrained policy optimization, formal verification of the discrete profile transition graph, and distribution-shift monitors over observation statistics. These mechanisms would not make the learned policy infallible, but they would ensure that uncertainty or out-of-distribution behavior degrades to a known baseline rather than to unconstrained escalation.

B. Interpreting the Learned Control Pattern

The aggregate action mix provides a compact behavioral explanation of the learned operating point. Across the 120 MARL episodes, 56.6% of router decisions use the balanced profile, 25.4% use conservative control, and only 18.0% use aggressive control. The local heuristic is more polarized: 41.1% balanced, 34.6% conservative, and 24.4% aggressive. Thus, MARL-Converge does not obtain its speed by approximating always-aggressive control. Instead, it spends more time in the middle profile and invokes aggressive behavior selectively along the evidence frontier.

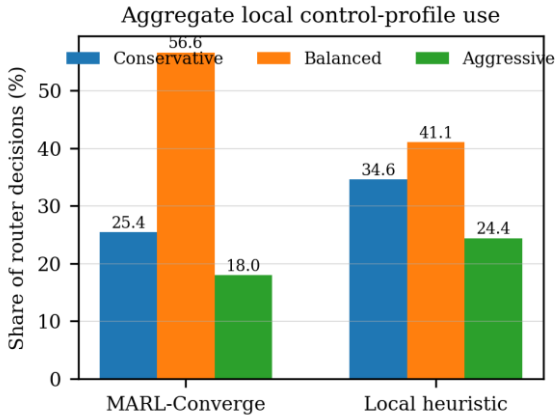


Fig. 7. Aggregate router-level action mix over the 120 matched real-topology episodes for each adaptive method.

The action mix also changes with scenario difficulty. Aggressive decisions account for 23.1% of actions on Abilene router failures but only 15.8% on AARNet link failures, where the larger topology makes indiscriminate escalation more expensive. Profile-switch distributions reinforce this interpretation: MARL-Converge has a median of 16 switches per episode and an interquartile range of 13–20, compared with a heuristic median of 36.5 and interquartile range of 26.75–50.25. Route-change totals are identical across methods in this proxy because each informed router recomputes once toward the same final unweighted shortest paths; the controllable quantities are therefore dissemination timing, transient forwarding quality, signaling cost, and profile churn rather than the final path set.

C. Limitations and Threats to Validity

The experiments use a discrete-time stochastic link-state proxy, not Mininet, ns-3, FRRouting, or physical routers. Decision steps and modeled control units cannot be converted to seconds or bytes without calibration. The profile constants are abstract, and the simulator omits vendor-specific timing, CPU scheduling, LSA retransmission details, weighted links, BGP policy exploration, and simultaneous correlated failures. Real-topology evaluation is limited to two modest networks, although AARNet is held out and synthetic stress tests extend to 40 nodes. Single failures are selected so the remaining network is connected, which excludes partition recovery. These limitations bound the claim to reproducible comparative evidence within the supplied environment. Training uses two random seeds and 300 episodes per seed. Evaluation uncertainty is reduced through 30 matched seeds per real topology and failure type, but broader training-seed sweeps could reveal additional policy variance. The telemetry stress variant was retrained, whereas the mask-disabled condition was not; the latter should be interpreted as a safety stress test rather than a causal learning ablation. Future work should integrate FRRouting/Mininet or ns-3, calibrate profile actions to concrete protocol parameters, model multi-failure and cascading events, add constrained or distributionally robust objectives, and compare against value-factorized MARL on larger heterogeneous networks.

VIII. CONCLUSION

This paper introduced MARL-Converge, a decentralized failure-aware routing-convergence controller based on parameter-shared dueling Double DQN. Router agents use local and one-hop observations, share experience during training, and execute independently. A hard action mask prevents aggressive control in the absence of failure evidence. In 600 matched episodes on Abilene and held-out AARNet, the method reduced convergence steps by 19.0% relative to Static-Balanced and by 9.3% relative to a strong local heuristic, while using 14.0% fewer control units and 56.4% fewer profile switches than the

heuristic. Always-aggressive control remained 7.6% faster but required 35.4% more control. The supplied code, model weights, raw results, and statistical summaries make these trade-offs auditable. The next step is calibrated emulation with real routing daemons, not extrapolation of simulator steps into unmeasured deployment claims.

References

C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," in Proc. ACM SIGCOMM, 2000, pp. 175–187, doi: 10.1145/347059.347428.

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

Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006, doi: 10.17487/RFC4271.

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

A. Atlas and A. Zinin, "Basic specification for IP Fast Reroute: Loop-Free Alternates," RFC 5286, Sep. 2008, doi: 10.17487/RFC5286.

M. Shand and S. Bryant, "A framework for loop-free convergence," RFC 5715, Jan. 2010, doi: 10.17487/RFC5715.

B. Wang, Z. Wang, W. Zhao, F. Zhang, and W. Shang, "DRL-Adapt: Deep reinforcement learning for adaptive routing convergence optimization in large-scale networks," IEEE Open Journal of the Computer Society, vol. 7, pp. 849–863, 2026, doi: 10.1109/OJCS.2026.3687441.

P. François, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," ACM SIGCOMM Computer Communication Review, vol. 35, no. 3, pp. 35–44, 2005, doi: 10.1145/1070873.1070877.

A. Basu and J. G. Riecke, "Stability issues in OSPF routing," in Proc. ACM SIGCOMM, 2001, pp. 225–236, doi: 10.1145/964723.383077.

Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz, "Route flap damping exacerbates Internet routing convergence," in Proc. ACM SIGCOMM, 2002, pp. 221–233, doi: 10.1145/633025.633047.

S. Knight et al., "The Internet Topology Zoo," IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1765–1775, Oct. 2011, doi: 10.1109/JSAC.2011.111002.

M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in Machine Learning Proceedings 1994, 1994, pp. 157–163, doi: 10.1016/B978-1-55860-335-6.50027-1.

F. A. Oliehoek and C. Amato, A Concise Introduction to Decentralized POMDPs. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-28929-8.

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. Artificial Intelligence, vol. 30, no. 1, 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. Machine Learning, PMLR, vol. 48, 2016, pp. 1995–2003.

T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, “QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning,” in Proc. 35th Int. Conf. Machine Learning, PMLR, vol. 80, 2018, pp. 4295–4304.

R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 6379–6390.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in Proc. AAAI Conf. Artificial Intelligence, vol. 32, no. 1, 2018, pp. 2974–2982, doi: 10.1609/aaai.v32i1.11794.

C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of PPO in cooperative multi-agent games,” in Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 24611–24624.

K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully decentralized multi-agent reinforcement learning with networked agents,” in Proc. 35th Int. Conf. Machine Learning, PMLR, vol. 80, 2018, pp. 5872–5881.

J. García and F. Fernández, “A comprehensive survey on safe reinforcement learning,” Journal of Machine Learning Research, vol. 16, no. 42, pp. 1437–1480, 2015.

J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: A reinforcement learning approach,” in Advances in Neural Information Processing Systems, vol. 6, 1994, pp. 671–678.

D. Teng, M. Rhee, Y. Qin, B. Zi, and W. Liu, "SW-SpeedDLM: Sliding window speculative decoding for diffusion language models under long context constraints," Mathematics, 2026.

Z. Wang, J. S. Yang, W. Shang, and J. Ding, "FairPromote: Explainable and fairness-aware talent promotion prediction via adversarial debiasing and SHAP-based interpretation," IEEE Access, 2026.

D. Teng, "TEAS: Token- and Energy-Aware Autoscaling for Cost-Efficient LLM Serving," AI and Data Science Journal, 2025

F. Zhang, Z. Guo, J. Ding, J. Yang, and W. Liu, "Adaptive sensor fusion for robust perception in dense fog: A gated vision and LiDAR integration framework," Sensors, 2026.

B. Zi, "Cloud-native distributed systems for real-time payment intelligence," AI and Data Science Journal, vol. 1, no. 1, pp. 51-56, 2024.