technologies

MDPI

Article

Robust Offline Multi-Agent Reinforcement Learning for Latency-Aware SDN Path Control in 6G-Oriented Network Softwarization

Abzal E. Kyzyrkanov [1,*], Yedil S. Nurakhov [2], Zhenis Otarbay [1,3,*] and Danil V. Lebedev [2]

[1] School of Software Engineering, Astana IT University, Astana 020000, Kazakhstan
[2] Department of Computer Sciences, Faculty of Information Technologies, Al-Farabi Kazakh National University, Almaty 050040, Kazakhstan; y.nurakhov@gmail.com (Y.S.N.); dan-lebedev@mail.ru (D.V.L.)
[3] Department of Computer Science, School of Engineering and Digital Sciences, Nazarbayev University, Astana 010000, Kazakhstan
* Correspondence: kyzyrkanov.abzal@astanait.edu.kz (A.E.K.); zhenis.otarbay@nu.edu.kz (Z.O.)

Abstract

Future sixth-generation (6G)-oriented networks require programmable control that can adapt routing to latency and congestion without unsafe online exploration. This study evaluates offline multi-agent deep deterministic policy gradient (MADDPG) with behavior-adjusted training rewards for latency-aware path control in software-defined networking (SDN). Each traffic pair is modeled as an agent selecting one of three retained candidate paths, while centralized critics learn coordinated decisions from topology-specific Ryu–Mininet transition datasets. Nine policies are compared using ten paired seeds on fat-tree, mesh-grid, and WAN-corridors topologies under a deployed utilization–latency weighting of 0.60/0.40, together with flow-completion, latency, congestion, architectural-comparison, sensitivity, robustness, statistical, and controller-overhead analyses. The utilization-aware path heuristic achieves the strongest overall reward ranking. MADDPG is the strongest learned policy on fat-tree, is not significantly outperformed by any evaluated policy on mesh-grid, and remains statistically tied with completion-matched policies on WAN-corridors. Behavior adjustment is topology-dependent rather than uniformly beneficial. The exported policy requires approximately 52 µs per joint decision, whereas complete control-loop timing is dominated by network-statistics polling. These results support offline multi-agent SDN control as a competitive, low-overhead option when interpreted jointly with topology structure, flow completion, and strong heuristic baselines.

Keywords: software-defined networking; offline reinforcement learning; multi-agent reinforcement learning; MADDPG; traffic engineering; path control; latency-aware routing; 6G network softwarization

Check for updates

Academic Editors: Sotirios K. Goudos and Panagiotis Sarigiannidis

Received: 23 June 2026
Revised: 27 July 2026
Accepted: 28 July 2026
Published: 30 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Sixth-generation (6G)-oriented networks are expected to combine connectivity, computation, artificial intelligence (AI), sensing, and service-aware control within programmable infrastructures [1–3]. In this paper, *6G-oriented* refers primarily to the softwarized transport, edge, and service-network layers that support heterogeneous and latency-sensitive applications rather than to the physical radio interface itself. Edge intelligence, tactile communication, industrial automation, autonomous systems, and distributed sensing strengthen the need for control mechanisms whose decisions are executable, measurable, and safe under changing network conditions [4–7].

Although the present experiments isolate the programmable path-control layer, reliable higher-layer control ultimately depends on information obtained from the underlying communication system. This coupling is particularly relevant to near-field extremely large-scale multiple-input multiple-output (XL-MIMO) systems, where channel estimation under low signal-to-noise ratio (SNR) conditions remains challenging and enhanced polar-domain estimation methods are being developed [8]. Integrating physical-layer estimates with SDN path control is outside the scope of this study; the Ryu–Mininet environment is instead used as a reproducible abstraction of the programmable transport and edge-control layer.

Software-Defined Networking (SDN) provides the architectural basis for this form of adaptive control by separating forwarding from logically centralized decision-making [9,10]. OpenFlow exposes programmable flow-table operations [11], allowing a controller to collect network measurements, select paths, and install forwarding rules. This capability has supported adaptive traffic engineering in both research systems and operational wide-area networks, including B4 and SWAN [12–14]. In the present study, Ryu provides the OpenFlow controller framework and Mininet provides executable hosts, links, and switches [15,16]. SDN is therefore not merely the evaluation platform; it is the mechanism through which learned path decisions are executed in a closed control loop.

Path selection becomes non-trivial when several feasible routes share network resources. Multi-rooted data-center fabrics such as VL2 and fat-tree expose substantial path diversity [17,18], and dynamic traffic-engineering systems have shown that this diversity can be used to respond to changing demands [19,20]. In lower-diversity networks, however, additional hops may increase latency without producing sufficient congestion relief. The evaluation therefore includes a path-diverse fat-tree, a lower-diversity mesh-grid, and a larger WAN-corridors topology to examine whether learned control remains competitive under structurally different routing opportunities.

Reinforcement learning (RL) is a natural candidate for adaptive SDN path control because it treats control as a sequential decision problem: an agent observes the current state, selects an action, and improves its policy according to a reward signal that reflects the quality of the resulting behavior. In routing, this framing is attractive because a path choice can affect not only the current packet delay, but also later congestion, bottleneck utilization, and path stability. Deep deterministic actor–critic methods such as Deep Deterministic Policy Gradient (DDPG) provide one foundation for learning control policies from experience [21], while multi-agent actor–critic methods extend this idea to systems in which several decision-makers interact through a shared environment [22]. More broadly, machine learning and deep reinforcement learning have been studied for networking tasks such as routing optimization, traffic engineering, resource management, and quality-of-service prediction [23–26]. However, the same interaction that makes RL powerful also makes it difficult to deploy in networks. A controller that explores directly in a live SDN environment may install poor forwarding rules, overload shared links, increase round-trip time, or create transient packet loss before the policy improves. These risks are part of the broader real-world RL challenge, where safety, limited exploration, non-stationarity, and evaluation under constrained data are central obstacles [27]. Offline RL addresses this issue by learning from previously collected transitions without requiring additional online exploration during training [28,29].

Offline learning, however, introduces its own unresolved problem. A learned policy can select actions that are weakly represented or absent in the logged dataset, causing the value function to extrapolate beyond reliable evidence. This difficulty has been studied in fixed-batch and offline RL, where behavior-constrained and conservative methods aim to reduce extrapolation error and stabilize value learning under distribution shift [30–32]. For SDN, this problem is not abstract: an unsupported action corresponds to a real path-

selection decision that may alter load, latency, and forwarding behavior in the emulated or operational network. Network softwarization and slicing research emphasizes programmable, service-aware control for future networks [33,34], and knowledge-defined networking similarly argues for combining SDN, analytics, and machine learning to support data-driven control decisions [35]. Yet these perspectives do not by themselves resolve how an offline learned controller should coordinate multiple simultaneous traffic-pair decisions over shared links while remaining close enough to the logged behavior to avoid unsupported actions. This is the specific gap addressed in the present study.

Prior research spans several related but distinct control settings. Online RL-based SDN routing can adapt through continued environment interaction, but exploratory actions may alter forwarding behavior and degrade live-network performance. Single-agent offline RL reduces this exploration risk but does not explicitly represent simultaneous per-traffic-pair decisions coupled through shared links. General multi-agent reinforcement-learning studies have demonstrated coordinated control for communication problems such as generalized wireless medium access control (MAC) protocols and massive machine-type random access [36,37], but they do not address executable offline SDN path installation. Recent project-level studies have also examined kernel-based reinforcement learning and log-driven proximal policy optimization for adaptive SDN control [38,39]. The contribution of the present work is not a new reinforcement-learning algorithm. It is the integrated formulation and evaluation of offline multi-agent SDN path control using topology-specific logged data, per-pair actors, centralized critics, behavior-adjusted training rewards, fixed-size candidate-path catalogues, and live Ryu–Mininet forwarding-rule installation.

This study addresses the above problem through offline Multi-Agent Deep Deterministic Policy Gradient (MADDPG) with behavior-adjusted training rewards for SDN candidate-path control. Each traffic pair is represented as an agent and selects one of $K = 3$ paths retained by a pruned breadth-first candidate-generation procedure. The controller then installs the resulting forwarding decisions through the SDN control loop. The practical specificity of the method is deliberately narrow: the learned controller does not synthesize arbitrary routes at inference time, but chooses among executable path alternatives that can be installed by the controller and evaluated using measured network behavior. A behavior-deviation adjustment is applied once through the training reward, discouraging actor outputs that depart substantially from the logged action representation without adding a supervised behavior-cloning term to the actor loss. This design differs from purely simulated routing learners because the policy is evaluated as an executable control component in a Ryu–Mininet loop, using both scalar reward and physical measurements. It also differs from heuristic path selection because it can learn topology- and load-dependent trade-offs among utilization, latency, and path cost rather than applying a fixed rule such as always choosing the shortest path.

Accordingly, the aim of this study is to determine whether offline MADDPG with behavior-adjusted training rewards can provide competitive, latency-aware SDN path control when routing decisions are constrained to executable candidate paths and evaluated in a live controller–emulator loop. The study is guided by six research questions. First, how does MADDPG compare with eight learning-based and heuristic baselines, including Conservative Q-Learning (CQL), Batch-Constrained Q-learning (BCQ), and the utilization-aware path heuristic (Util-aware SP), across fat-tree, mesh-grid, and WAN-corridors topologies? Second, how do topology structure and flow-completion differences explain the observed reward, utilization, and round-trip time (RTT) outcomes? Third, what roles are played by the centralized critic, the per-pair multi-agent decomposition, the behavior-adjusted training reward, and the selected learning hyperparameters? Fourth, are the principal conclusions stable under alternative utilization–latency scalarizations and

hop/switch-penalty multipliers applied to the same deployed trajectories? Fifth, how does the controller respond to bursty and mixed traffic, link-down and packet-loss conditions, and an Equal-Cost Multi-Path (ECMP) forwarding reference? Finally, what are the isolated policy-inference cost, complete SDN control-loop timing, and computational scaling as the numbers of agents and candidate paths increase?

The principal conclusion is deliberately bounded. Util-aware SP achieves the strongest overall reward ranking, placing first on fat-tree and WAN-corridors and second on mesh-grid. MADDPG is nevertheless the strongest learned policy on fat-tree, no evaluated policy significantly outperforms it on mesh-grid, and it remains statistically tied with the completion-matched policy group on WAN-corridors. The mechanism analysis shows that the fat-tree advantage of Util-aware SP is primarily associated with lower bottleneck utilization at comparable completion, RTT, and path length, whereas WAN-corridors requires reward comparisons to be qualified by delivered traffic. Behavior adjustment is also topology-dependent rather than uniformly beneficial. The study therefore supports offline MADDPG as a competitive and executable learned controller, but not as a universal replacement for topology-aware routing heuristics. Its relevance to 6G lies in the AI-assisted network-softwarization layer, where learned decisions must be assessed jointly with topology, reliability outcomes, and complete controller overhead.

The remainder of the paper is organized as follows. Section 2 describes the Ryu–Mininet control architecture, the three evaluated topologies, the offline datasets and candidate-path construction, the reward formulation, the offline MADDPG training procedure, the eight comparison policies, the evaluation metrics, and the auxiliary traffic, fault, ECMP, timing, and scalability experiments. Section 3 reports the primary reward and rank comparisons, flow completion, RTT and utilization measurements, architectural-variant and hyperparameter analyses, reward and penalty sensitivity, routing behavior, paired statistics, auxiliary traffic, fault, and ECMP tests, controller overhead, scalability, and operating-envelope results. Section 4 interprets the topology-dependent mechanisms and implications for 6G-oriented network softwarization, and Section 5 summarizes the bounded conclusions and future work.

2. Materials and Methods

This section describes the experimental system, learning formulation, and evaluation protocol used to assess offline multi-agent reinforcement learning for SDN path control. The methodology has four main parts. First, logged SDN interactions are collected from a Ryu–Mininet environment and converted into an offline transition dataset. Second, an MADDPG controller with behavior-adjusted training rewards is trained using centralized training with decentralized execution, where each traffic pair acts as a routing agent over a fixed-size candidate-path action space. Third, the trained policy and eight comparison policies are deployed in live SDN evaluations on the fat-tree, mesh-grid, and WAN-corridors topologies under matched topology, offered-traffic profile, and seed conditions. Finally, the resulting trajectories are analyzed through reward re-scoring, architectural-variant and behavior-adjustment analyses, hyperparameter sensitivity, policy-behavior metrics, flow-completion analysis, paired statistics, and controller-overhead measurements. Auxiliary experiments evaluate bursty and mixed traffic, link-down and packet-loss conditions, ECMP forwarding, full control-loop timing, policy-decision scalability, and the MADDPG operating envelope across three offered-traffic profiles. All methodological details reported below correspond to the implementation used to generate the results in Section 3.

2.1. SDN Control-Loop Architecture

The experimental system follows a software-defined networking architecture in which forwarding decisions are made by a logically centralized control plane and then installed in the data plane through flow rules. This design is consistent with the OpenFlow model of programmable forwarding control [11]. In the implementation used in this study, the control plane is built around the Ryu SDN framework [15], while the data plane is emulated in Mininet [16]. Mininet provides the controlled live-emulation environment in which the path-control policies are evaluated under matched topology, offered-traffic profile, and seed conditions.

The experimental environment should be interpreted as a programmable-network control testbed for the softwarized infrastructure layer of 6G-oriented systems. It does not emulate a complete 6G radio-access network. Rather, it isolates the path-control problem that can arise in programmable transport, edge, and service-network segments supporting latency-sensitive traffic. This abstraction allows the learning and deployment properties of the controller to be evaluated under controlled topology, offered-traffic profile, and seed conditions while preserving executable SDN forwarding behavior.

Figure 1 summarizes the control-loop architecture used throughout the study. The workflow has two stages. In the offline stage, Ryu–Mininet interactions generated under a behavior policy are logged and converted into an offline transition dataset. This dataset is then used to train an MADDPG controller with behavior-adjusted training rewards under centralized training with decentralized execution. In the deployment stage, the trained controller is exported as a compact path-selection policy and executed inside the closed-loop SDN evaluation.

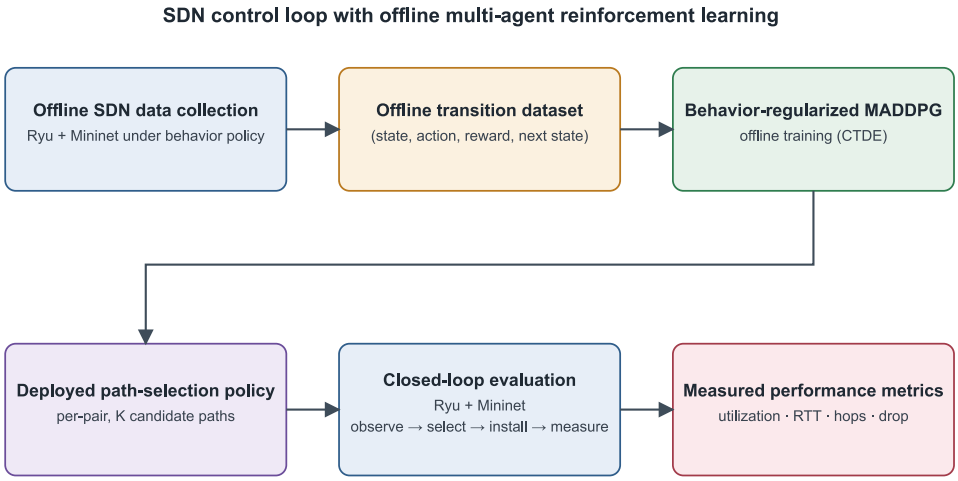


Figure 1. Offline training and closed-loop deployment workflow for the proposed SDN path-control policy. Logged SDN interactions are converted into an offline transition dataset, used to train an MADDPG controller with behavior-adjusted training rewards, and exported as a deterministic argmax path-selection policy for live Ryu–Mininet evaluation.

During live evaluation, each control step begins with a state observation collected by the SDN controller. The deployed policy then selects one candidate path for each traffic pair from the current $K = 3$ candidate catalogue. The selected paths are translated into forwarding decisions and installed in the emulated data plane through the controller. Traffic is then forwarded over the installed paths, and the resulting performance measurements are logged for analysis. The measured quantities include bottleneck utilization, RTT, hop count, selected-path changes, packet-drop information, and flow-completion outcomes, which are subsequently used for reward computation, policy-behavior analysis, completion qualification, and statistical evaluation.

The controller implementation was also instrumented to distinguish isolated policy inference from complete control-loop latency. For one control interval, the elapsed time is represented as

$$T_{\text{loop}} = T_{\text{poll}} + T_{\text{state}} + T_{\text{decision}} + T_{\text{install}} + T_{\text{barrier}} + T_{\text{other}},$$

where T_{poll} is the network-statistics polling time, T_{state} is the state-construction time, T_{decision} is the controller-side path-selection time, T_{install} is the forwarding-rule installation time, T_{barrier} is the OpenFlow barrier-synchronization time, and T_{other} contains the remaining controller and emulator operations within the measured interval. The isolated NumPy-policy benchmark measures only actor evaluation and candidate-path selection, whereas the full-loop measurement includes the surrounding monitoring and rule-update workflow.

This architecture separates learning from deployment. The policy is trained offline from previously logged transitions, so no exploratory reinforcement-learning actions are required during live evaluation. At the same time, deployment remains closed-loop: the trained controller observes the current network state, selects paths, installs forwarding rules, and receives new performance measurements at subsequent control steps. This separation is central to the experimental design because it allows learned path control to be evaluated in a live SDN emulator while avoiding online exploration inside the network under test.

2.2. Network Topologies and Traffic Setting

The first topology is a fat-tree fabric with $k = 4$, representing a path-diverse data-center-style network. Fat-tree networks were originally proposed as scalable commodity data-center architectures with multiple alternative paths between hosts [18]. This structure tests whether a path-control policy can exploit route diversity to reduce congestion rather than selecting alternatives uniformly.

The second topology is a 4×4 mesh-grid. It provides shorter feasible routes and less useful path diversity than the fat-tree, making shortest-hop routing a strong topology-specific reference. This setting tests whether learned controllers suppress unnecessary detours when alternative paths provide limited congestion relief.

The third topology is WAN-corridors, a larger configuration containing 26 switches and $N = 4$ controlled traffic pairs. Its candidate paths follow corridor-like structures in which alternative routes can remain dependent on shared network segments. This topology extends the evaluation beyond the two compact fabrics and tests the controller under longer paths, shared corridor bottlenecks, and a structurally different wide-area SDN setting.

The primary comparison uses the validated offered-traffic profile with ten paired seeds for each topology and policy. Policies evaluated with the same topology, offered-traffic profile, and seed form matched comparisons for the reward and statistical analyses. The separate operating-envelope experiment evaluates MADDPG only using a uniform 3 topologies $\times$ 3 offered-traffic profiles $\times$ 3 matched seeds design. The three nominal profiles are moderate-low, validated, and heavy: moderate-low denotes the lower configured offered-traffic profile, validated denotes the primary configuration used for the main policy comparison, and heavy denotes the highest configured profile. These labels describe traffic-generator settings rather than topology-normalized realized load levels; consequently, their realized utilization can differ across fabrics.

Figure 2 summarizes the evaluation and analysis pipeline. The topology, offered-traffic profile, seed, and policy configurations define the live Ryu–Mininet evaluations. The resulting trajectories contain selected paths, controller-side observations, and measured performance outcomes used for reward re-scoring, architectural-variant analysis, hyperparameter sensitivity, policy-behavior analysis, flow-completion qualification, paired

statistics, and controller-overhead evaluation. Separate three-seed auxiliary experiments examine bursty and mixed traffic patterns, link-down and packet-loss conditions, and an ECMP forwarding reference.

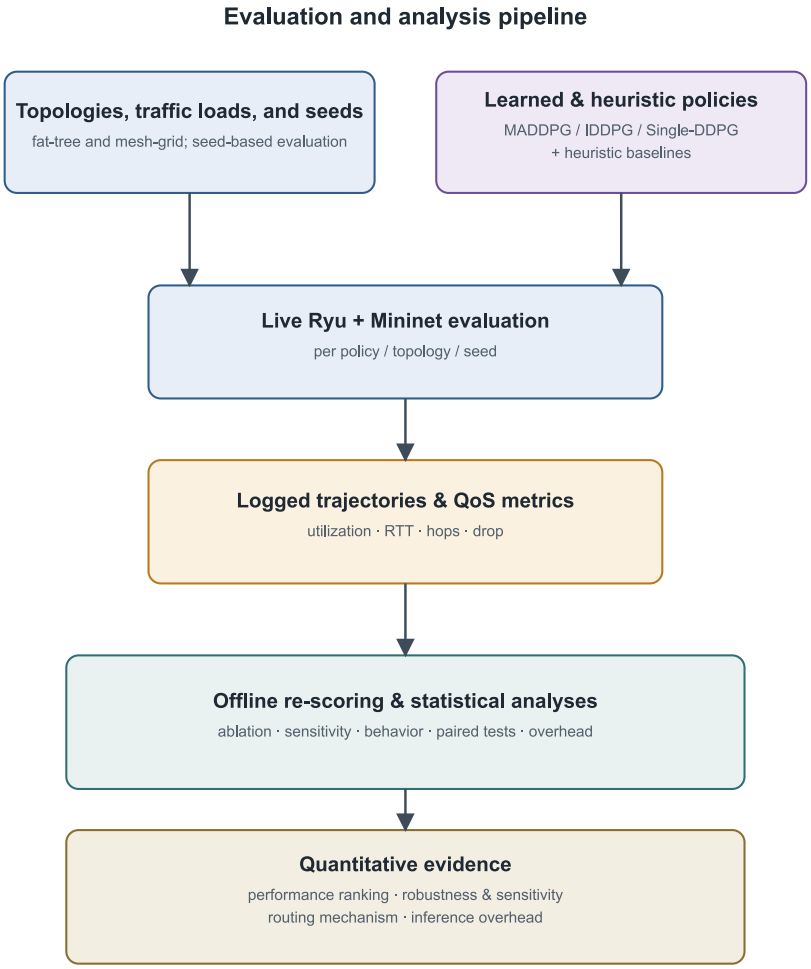


Figure 2. Evaluation and analysis pipeline. Topologies, offered-traffic profiles, seeds, and path-control policies define the live Ryu–Mininet evaluations. Logged trajectories and measured outcomes are used for reward re-scoring, architectural-variant analysis, sensitivity analysis, policy-behavior and flow-completion analysis, paired statistical testing, and controller-overhead evaluation. Auxiliary experiments examine bursty and mixed traffic, link-down and packet-loss conditions, and ECMP forwarding.

Table 1 summarizes the topology and offered-traffic coverage. The ten-seed validated-profile evaluation is used for the main nine-policy comparison, RTT and behavior measurements, flow-completion analysis, and paired statistics. The uniform three-seed MADDPG grid is reported separately as a single-method operating-envelope analysis and does not replace the primary comparison. The bursty, mixed-traffic, fault, and ECMP conditions are auxiliary three-seed evaluations reported in their corresponding results subsection.

The primary policy set is identical across the three topologies and contains nine controllers. The proposed method is compared with two architectural variants. Independent Deep Deterministic Policy Gradient (IDDPG) trains the traffic-pair agents independently without centralized joint-action evaluation, whereas Single-DDPG replaces the per-pair multi-agent decomposition with one learned controller. CQL and BCQ provide additional offline reinforcement-learning baselines. The heuristic set contains Round-robin, Shortest-hops, Min-switch-cost, and Util-aware SP. Util-aware SP selects the retained candidate path with the lowest currently observed bottleneck utilization. This policy set compares

centralized multi-agent control with architectural variants, offline reinforcement-learning baselines, and topology-specific routing heuristics while preserving the same $K = 3$ action cardinality and candidate-generation procedure.

Table 1. Topology and offered-traffic coverage. The main comparison evaluates nine policies with ten paired seeds per topology; the operating-envelope analysis evaluates MADDPG with three seeds per offered profile.

Topology and Structural Role	Main Comparison	MADDPG Operating Envelope
Fat-tree ($k = 4$); path-diverse fabric	Validated profile; 9 policies; 10 paired seeds	Moderate-low, validated, and heavy; 3 seeds per profile
Mesh-grid (4×4); lower-diversity fabric	Validated profile; 9 policies; 10 paired seeds	Moderate-low, validated, and heavy; 3 seeds per profile
WAN-corridors ($N = 4$, 26 switches); corridor-like network	Validated profile; 9 policies; 10 paired seeds	Moderate-low, validated, and heavy; 3 seeds per profile

2.3. Offline Dataset and Logged Transition Structure

The learning problem is constructed from logged SDN controller interactions rather than from online exploration during evaluation. Each logged sample corresponds to one control step in the Ryu–Mininet environment and records the observed network state, the joint candidate-path action, the per-agent training rewards, the subsequent network state, and supplementary measured reward ingredients. We represent each transition as

$$\tau_t = (s_t, \mathbf{a}_t, \mathbf{r}_t, s_{t+1}), \qquad \mathbf{r}_t = (r_{1,t}, \ldots, r_{N,t}),$$

where s_t is the full controller-side state observation at control step t, $\mathbf{a}_t$ is the joint vector of logged discrete candidate-path indices selected for the traffic-pair agents, $\mathbf{r}_t$ is the per-agent reward vector used during MADDPG training, and s_{t+1} is the next full state observation. During MADDPG training, the logged action indices are converted to the bipolar critic-action representation described in Section 2.6. The scalar controller reward R_t defined in Section 2.5 is reconstructed from the logged reward ingredients and is used for episode-level evaluation, reward re-scoring, and policy ranking. A separate offline dataset is generated for each topology under the validated offered-traffic profile using the same random behavior policy and three collection seeds. At each control step, the behavior policy selects one of the $K = 3$ candidate paths for each traffic-pair agent, the joint selection is installed by the controller, and the resulting measurements and next observation are recorded. This procedure produces 2067 transitions for fat-tree, 1833 for mesh-grid, and 2725 for WAN-corridors. Policy training samples only from these fixed topology-specific datasets and performs no interaction with the Ryu–Mininet environment or online exploration.

The controller state comprises a shared network-summary block and candidate-path feature blocks. The shared block contains aggregate utilization, traffic-control, latency, and previous-action information. Each candidate-path block contains a feasibility indicator, bottleneck and mean path utilization, normalized hop count, and normalized switch cost. For N traffic pairs, K candidate paths per pair, a shared-block width $G = 7$, and five features per candidate, the full state supplied to each centralized critic has dimension

$$d(s) = G + 5KN.$$

The decentralized observation supplied to agent i contains the shared block and only the candidate-path block associated with its own traffic pair:

$$d(o_i) = G + 5K.$$

With $N = 4$ and $K = 3$, the critic receives a 67-dimensional state and each actor receives a 22-dimensional observation.

Table 2 summarizes the observation layout, transition fields, reward ingredients, and supplementary per-flow records used by the offline learning and analysis pipeline. The scalar reward is not treated as an opaque value. Instead, the underlying reward ingredients are logged so that the reward can be recomputed during analysis. This enables the ex post reward-weight and hop/switch-penalty sensitivity experiments reported later in the paper.

Table 2. State and transition structure used for offline learning.

Component	Contents	Role
Shared network block	Aggregate utilization, traffic-control, latency, and previous-action features	Common network context available to all agents
Candidate-path block	Feasibility, bottleneck utilization, mean utilization, normalized hop count, and normalized switch cost	Describes one retained routing candidate, including its feasibility state
Global state s_t	Shared block and candidate blocks for all traffic pairs	Input to the centralized critics during training
Local observation $o_{i,t}$	Shared block and candidate blocks for traffic pair i	Input to actor i during training and deployment
Joint action $\mathbf{a}_t$	One candidate-path index per traffic pair	Determines the paths installed by the SDN controller
Per-agent reward vector $\mathbf{r}_t$	One base training reward per traffic-pair agent	Used to form the behavior-adjusted MADDPG training rewards
Reward ingredients	Utilization, drop-related, latency, and path-cost terms	Support scalar reward reconstruction and sensitivity analysis
Next state s_{t+1}	Subsequent state in the same representation	Completes the offline transition tuple

Missing or infeasible candidates are encoded as infeasible with maximal normalized path costs rather than as zero-valued paths, preventing padding from being interpreted as an attractive routing option.

Flow completion is derived from the per-flow traffic logs rather than from the per-step link-drop field. For each evaluation episode, the traffic-dispatch records are matched by timestamp to the episode timestamp interval in the evaluation trajectory. The associated per-flow records are then selected, and completion is computed as the fraction with successful process termination status equal to zero. Timestamp matching is required because the traffic-validity logs are append-only and can contain retries or separate launches under the same topology, policy, and seed identifiers.

For the main validated-profile evaluation, each topology is evaluated with ten paired seeds. Per-episode reward is computed after removing the initial 10% warm-up portion of the episode, and the reported policy score is the mean over the retained steps and then over seeds. This warm-up removal prevents the initial transient controller state from dominating the episode-level score.

Reward reconstruction from the logged ingredients was verified to numerical precision before the ex post sensitivity analyses. A data-hygiene check also identified four evaluation files containing a stale partial append followed by a complete rerun under the same run identifier. The analysis retained the final complete episode without overwriting the source files; this correction did not change any reported policy ranking. These checks support the analyses in Sections 3.7 and 3.8, which re-score fixed logged trajectories rather than collecting new Mininet runs.

2.4. Problem Formulation

The SDN path-control problem is formulated as a discrete candidate-path selection problem with multiple routing agents. Let $\mathcal{P} = \{1, \ldots, N\}$ denote the set of traffic pairs controlled by the SDN controller. Each traffic pair $i \in \mathcal{P}$ is represented by one routing agent. At each control step t, the controller observes the network state s_t, and each agent selects one candidate path for its corresponding traffic pair.

For each traffic pair i, the controller maintains a candidate-path set of fixed cardinality K,

$$\mathcal{C}_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,K}\}.$$

The controller constructs each candidate set from the topology discovered for the relevant data-collection or live-evaluation run. Each retained path connects the designated source and destination, is loop-free, and is distinct from the other candidates for the same traffic pair. Candidate generation uses a pruned breadth-first expansion and terminates when the first $K = 3$ complete paths admitted by that procedure have been found; no ranking over the complete feasible path set is performed. The retained catalogue is therefore not guaranteed to contain the K globally shortest paths. After truncation, the retained paths are sorted by hop count and node-identifier sequence to assign stable action indices within that catalogue. Because the reported runs used dynamic topology discovery without explicitly sorting adjacency insertion order before generation, exact catalogue membership can depend on topology-discovery order across separate runs. Nominal changes in utilization and latency affect path selection but do not regenerate the catalogue, whereas a detected topology change, such as link removal, can trigger candidate-path recomputation. All evaluated policies use the same generation procedure and the same $K = 3$ action cardinality, although independently discovered runs are not guaranteed to retain identical node sequences.

The local action of agent i is therefore a discrete candidate-path index,

$$a_{i,t} \in \{1, 2, \ldots, K\}.$$

The joint routing action at control step t is

$$\mathbf{a}_t = (a_{1,t}, a_{2,t}, \ldots, a_{N,t}),$$

and deployment maps this joint action to one selected path per traffic pair:

$$\pi_{\text{deploy}}(s_t) = \left(c_{1,a_{1,t}}, c_{2,a_{2,t}}, \ldots, c_{N,a_{N,t}}\right).$$

The selected paths are then installed by the SDN controller as single-path forwarding decisions for the current control interval.

The learning problem is defined over logged transitions

$$\tau_t = (s_t, \mathbf{a}_t, \mathbf{r}_t, s_{t+1}),$$

where s_t is the controller-side state observation, $\mathbf{a}_t$ is the joint candidate-path action, $\mathbf{r}_t = (r_{1,t}, \ldots, r_{N,t})$ is the per-agent reward vector used during offline training, and s_{t+1} is the next observed state. The behavior-adjusted per-agent rewards used by MADDPG are defined in Section 2.6. Policy performance is evaluated using the scalar controller reward R_t defined in Section 2.5, with the corresponding objective

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t R_t\right],$$

where T is the episode length and γ is the discount factor.

The multi-agent structure is important because the routing decisions for different traffic pairs are coupled through shared links. A path selected for one traffic pair can change the bottleneck utilization and latency experienced by another. Therefore, each agent executes a local path-selection decision, but training can use information about the joint state and joint action. This motivates the centralized-training/decentralized-execution formulation used by MADDPG: during training, each critic can evaluate coordinated joint actions, while during deployment, each traffic-pair agent contributes to the selected joint routing action through the learned policy.

This formulation keeps the deployed control action simple and SDN-compatible. During nominal operation, the controller does not synthesize arbitrary routes at each decision step; it uses the current observation to select among the retained candidates and installs the corresponding forwarding rules. Candidate paths are recomputed only when topology discovery reports a structural change. The learned and heuristic controllers therefore use the same candidate-generation procedure, action cardinality, and Ryu–Mininet control loop, while differing in how the current network state is mapped to a candidate-path index.

2.5. Reward and Cost Function

The controller is evaluated using a scalar reward constructed from measured network quantities. The reward is defined as the negative of a routing cost, so higher reward corresponds to better performance and zero is the best attainable value. At control step t, the cost is

$$C_t = w_u U_t + w_d D_t + w_l L_t + P_t,$$

and the reward is

$$R_t = -C_t = -(w_u U_t + w_d D_t + w_l L_t + P_t),$$

where U_t is the bottleneck-utilization term, D_t is the drop-related term, L_t is the normalized RTT-related loss, and P_t is the realized gated hop/switch penalty. The scalar reward R_t is used for episode-level evaluation, reward re-scoring, and policy ranking, whereas MADDPG training uses the corresponding per-agent rewards and their behavior-adjusted forms defined in Section 2.6. The quantity D_t is a per-step link-drop term derived from controller-side measurements; it does not measure end-to-end flow completion. Because $w_d = 0$ in the deployed reward, policy comparisons on WAN-corridors are interpreted jointly with the separately measured flow-completion rate.

The deployed reward used in the primary validated-profile experiments is parameterized by

$$w_u = 0.60, \qquad w_d = 0, \qquad w_l = 0.40.$$

This configuration assigns greater weight to bottleneck utilization than to normalized latency loss while retaining latency as a substantial component of the routing objective.

Table 3 summarizes the reward terms and their roles. The utilization term penalizes routing decisions that place traffic on highly loaded bottleneck links. The RTT-related loss penalizes high end-to-end latency. The per-step link-drop term is retained in the objective definition, although its weight is zero in the deployed configuration and it is not used as a proxy for end-to-end flow completion. The hop/switch penalty discourages unnecessary detours and path changes through a gated excess path-cost term.

Table 3. Reward and cost terms used for offline scoring and sensitivity analysis.

Term	Meaning	Role in Objective
U_t	Bottleneck utilization	Penalizes congested selected paths
D_t	Per-step link-drop term	Retained in objective; $w_d = 0$; does not measure flow completion
L_t	Normalized RTT-related loss	Penalizes high end-to-end latency
P_t	Realized hop/switch penalty	Penalizes gated excess path cost

For an episode of length T, the reported episode-level reward is computed after removing the initial warm-up portion of the trajectory. Let $\mathcal{I}$ denote the retained set of time steps after the 10% warm-up skip. The episode score is

$$\bar{R}_{\text{episode}} = \frac{1}{|\mathcal{I}|} \sum_{t \in \mathcal{I}} R_t.$$

Policy-level scores are then obtained by averaging episode scores across the evaluated seeds within each topology.

The same logged reward ingredients are also used for ex post objective sensitivity analysis. For a general utilization–latency trade-off with $w_d = 0$, the re-scored reward is

$$R_t(w_u, w_l) = -(w_u U_t + w_l L_t + P_t), \qquad w_l = 1 - w_u.$$

This re-scoring does not retrain the policies; it evaluates the same logged trajectories under alternative scalarizations of the measured quantities. Therefore, the sensitivity analysis tests whether the observed ranking is stable under alternative reward interpretations of the same deployed behavior.

The hop/switch penalty sensitivity is computed in the same ex post manner. Let λ denote a multiplier applied to the realized combined hop/switch penalty. The re-scored reward is

$$R_t(\lambda) = -(w_u U_t + w_d D_t + w_l L_t + \lambda P_t),$$

with $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$. The evaluated penalty multipliers are

$$\lambda \in \{0,\ 0.5,\ 1,\ 2\}.$$

The deployed reward corresponds to $\lambda = 1$. This formulation tests whether the qualitative conclusions depend on the exact magnitude of the realized hop/switch penalty while keeping the physical trajectories fixed.

Reward reconstruction from the logged ingredients was verified numerically before the ex post weight and penalty sensitivity analyses, which re-score fixed logged trajectories rather than collecting new SDN evaluation episodes.

2.6. Offline MADDPG with Behavior-Adjusted Training Rewards

The proposed controller uses Multi-Agent Deep Deterministic Policy Gradient (MADDPG) with centralized training and decentralized execution (CTDE). Deep Deterministic

Policy Gradient (DDPG) provides the deterministic actor–critic foundation in which a parameterized actor produces continuous action scores and a critic estimates their action value [21]. In the present implementation, these scores are converted to a discrete candidate-path decision by an arg max operation at deployment. This preserves a differentiable actor representation during training while producing an executable discrete SDN action. DDPG also provides a shared actor–critic basis for comparison among the implemented MADDPG, IDDPG, and Single-DDPG architectures. MADDPG extends DDPG through CTDE, allowing each traffic-pair agent to execute its own policy while its critic is trained using joint information from the multi-agent system [22]. This structure is appropriate because routing decisions are executed per traffic pair but remain coupled through shared links and congestion.

In this study, each traffic pair is modeled as one routing agent. Agent i has an actor $\mu_i(o_i;\theta_i^{\mu})$, where o_i denotes its local observation and θ_i^{μ} denotes the actor parameters. The actor produces a continuous K-dimensional action-score vector

$$\mathbf{u}_{i,t} = \mu_i(o_{i,t};\theta_i^{\mu}) \in [-1,1]^K.$$

For MADDPG training, the logged discrete candidate-path index $a_{i,t}$ is represented by the bipolar vector

$$b_{i,t,j} = \begin{cases} +1, & j = a_{i,t}, \\ -1, & j \neq a_{i,t}, \end{cases} \qquad j \in \{1,\ldots,K\}.$$

Thus, exactly one component of $\mathbf{b}_{i,t}$ equals $+1$, and the remaining components equal -1. This bipolar representation, rather than the integer index, is supplied as the logged action to the critic. During deployment, the continuous actor output is converted back to a discrete candidate-path index by

$$a_{i,t} = \arg\max_{k\in\{1,\ldots,K\}} \mu_i(o_{i,t};\theta_i^{\mu})_k.$$

The selected indices form the joint routing action and are mapped by the SDN controller to one installed path per traffic pair.

During centralized training, each agent has a critic that evaluates the global state and the joint continuous action representation. Let

$$\mathbf{u}_t = (\mathbf{u}_{1,t},\ldots,\mathbf{u}_{N,t}),$$

and denote the critic of agent i by

$$Q_i(s_t,\mathbf{u}_t;\theta_i^{Q}).$$

For logged transitions, the critic action input is the corresponding joint bipolar action vector $\mathbf{b}_t = (\mathbf{b}_{1,t},\ldots,\mathbf{b}_{N,t})$.

Because training is performed from a fixed dataset, the implementation discourages deviations from the logged behavior through the reward used in the Bellman target rather than through an additional supervised actor loss. For agent i, the behavior-deviation term is

$$d_{i,t} = \frac{1}{K}\sum_{j=1}^{K}\left(u_{i,t,j} - b_{i,t,j}\right)^2.$$

The reported models use the per-agent behavior-adjusted reward

$$r_{i,t}^{\mathrm{BR}} = r_{i,t} - \beta d_{i,t} = r_{i,t} - \frac{\beta}{K}\sum_{j=1}^{K}\left(u_{i,t,j} - b_{i,t,j}\right)^2.$$

The selected configuration uses $\beta = 0.1$, while the $\beta = 0$ sensitivity variant removes the behavior-deviation adjustment without changing the remaining MADDPG architecture.

The critic is trained using the standard mean-squared Bellman error

$$\mathcal{L}_{Q_i} = \mathbb{E}_{(s,\mathbf{b},\mathbf{r},s')\sim\mathcal{D}}\left[\left(Q_i(s,\mathbf{b};\theta_i^Q) - y_i\right)^2\right],$$

with the unprojected target

$$\widetilde{y}_i = r_i^{\mathrm{BR}} + \gamma Q_i'\left(s', \mathbf{u}'; \theta_i^{Q'}\right),$$

where

$$\mathbf{u}' = \left(\mu_1'(o_1'; \theta_1^{\mu'}), \ldots, \mu_N'(o_N'; \theta_N^{\mu'})\right).$$

Here, Q_i' and μ_i' denote the critic and actor target networks, respectively, and γ is the discount factor.

When target projection is enabled, the implementation defines the empirical interval

$$\mathcal{Y} = [q_{\min}, q_{\max}],$$

where

$$q_{\min} = \min\left(r_{\min}, \frac{r_{\min}}{1-\gamma}\right), \qquad q_{\max} = \max\left(r_{\max}, \frac{r_{\max}}{1-\gamma}\right),$$

and $r_{\min}$ and $r_{\max}$ are the empirical minimum and maximum of the per-agent reward array used in the corresponding training run. The projected target is then

$$y_i = \mathrm{proj}_{\mathcal{Y}}(\widetilde{y}_i).$$

When projection is disabled, $y_i = \widetilde{y}_i$. The interval is motivated by the discounted-return envelope for bounded per-step rewards under $0 \le \gamma < 1$.

Target projection was disabled for the primary fat-tree and mesh-grid MADDPG models and enabled for the primary WAN-corridors model. IDDPG and Single-DDPG used the same empirical projection rule on all three topologies, whereas CQL and BCQ did not use target projection. Projection settings were matched within each behavior-adjustment comparison: projection was disabled for both β variants on fat-tree and mesh-grid and enabled for both variants on WAN-corridors. The learning-rate, discount-factor, and hidden-width sensitivity runs on fat-tree did not use target projection.

The actor objective remains the standard deterministic MADDPG objective:

$$\mathcal{L}_{\mu_i} = -\mathbb{E}_{s\sim\mathcal{D}}\left[Q_i\left(s, \mathbf{u}_1, \ldots, \mu_i(o_i;\theta_i^{\mu}), \ldots, \mathbf{u}_N\right)\right].$$

No supervised behavior-cloning term is added to either the actor or critic loss; β influences learning only through the behavior-adjusted reward entering the Bellman target.

Table 4 reports the selected training configuration and the single-parameter alternatives evaluated in the sensitivity analysis. Each alternative changes only the indicated parameter while preserving the same offline dataset and 30,000-step training budget.

Algorithm 1 summarizes the training and deployment workflow. The key design choice is that learning is performed offline from logged Ryu–Mininet transitions, whereas deployment uses only deterministic path selection from the trained actors. Therefore, the deployed controller does not explore inside the live network. It observes the current state, selects one candidate path per traffic pair, installs the corresponding forwarding rules, and logs the resulting performance measurements for evaluation.

Table 4. Selected MADDPG training configuration and tested alternatives.

Parameter	Selected Value	Tested Alternatives
Learning rate	10^{-3}	2×10^{-4}, 5×10^{-3}
Discount factor γ	0.99	0.90, 0.995
Hidden units per layer	128	64, 256
Behavior-adjustment weight β	0.1	0
Training iterations	30,000	Not varied

Algorithm 1 Offline MADDPG with behavior-adjusted training rewards for SDN path control

Require: Offline transition dataset $\mathcal{D}$; traffic-pair agents $i = 1, \ldots, N$; candidate-path sets $\mathcal{C}_i$ with $K = 3$; behavior-adjustment weight β

Initialize actor networks μ_i, critic networks Q_i, and corresponding target networks
for each offline training iteration **do**
  Sample a mini-batch $(s, \mathbf{b}, \mathbf{r}, s')$ containing bipolar logged-action vectors and per-agent rewards
  Compute current actor outputs $\mathbf{u}_i = \mu_i(o_i)$ for the sampled states
  Compute $d_i = K^{-1}\|\mathbf{u}_i - \mathbf{b}_i\|_2^2$ and form $r_i^{\mathrm{BR}} = r_i - \beta d_i$
  Compute target actor-score vectors $\mathbf{u}'$ and form $\widetilde{y}_i$
  Set $y_i = \mathrm{proj}_{\mathcal{Y}}(\widetilde{y}_i)$ when target projection is enabled; otherwise set $y_i = \widetilde{y}_i$
  Update each centralized critic Q_i by minimizing $\mathcal{L}_{Q_i}$ using y_i
  Update each actor μ_i using the standard deterministic critic objective $\mathcal{L}_{\mu_i}$
  Update target networks
end for
Export the trained actors as the deterministic argmax candidate-score policy
for each live SDN control step **do**
  Observe the current controller-side network state s_t
  for each traffic pair i **do**
    Select $a_{i,t} = \arg\max_k \mu_i(o_{i,t})_k$
    Map $a_{i,t}$ to candidate path $c_{i,a_{i,t}}$
  end for
  Install the selected paths through the SDN controller
  Log measured network metrics for evaluation
end for

2.7. Baselines and Architectural Variants

The proposed controller is compared against learning-based and heuristic path-control policies. The learning-based variants and offline reinforcement-learning baselines test alternative learned formulations, while the heuristic baselines represent deployable routing strategies that do not require offline training. All policies operate over the same candidate-path abstraction: for each traffic pair, one path is selected from the retained $K = 3$ candidates and installed by the SDN controller.

Table 5 summarizes the nine policies used in the main ten-seed comparison. IDDPG trains each traffic-pair agent independently without centralized joint-action evaluation. Single-DDPG replaces the per-pair multi-agent decomposition with one learned controller for the joint path-control decision. These two methods provide implementation-level architectural comparisons with MADDPG; because target-projection settings are not matched on every topology, they are not interpreted as perfectly isolated one-factor ablations.

Conservative Q-Learning (CQL) and Batch-Constrained Q-learning (BCQ) provide additional offline reinforcement-learning references using the same topology-specific datasets, 22-dimensional local observations, and $K = 3$ candidate-action abstraction as the decentralized MADDPG actors. CQL applies conservative Q-value regularization with $\alpha_{\mathrm{CQL}} = 1.0$ and selects the maximum-valued candidate at deployment. BCQ uses a learned behavior-support model with relative-support threshold $\tau_{\mathrm{BCQ}} = 0.3$ and selects the highest-valued

supported candidate. Both use the raw logged rewards, without the MADDPG behavior-deviation reward adjustment or target projection. These baselines test whether the proposed multi-agent formulation remains competitive with established offline-RL alternatives under the same local observation and candidate-action abstraction.

Table 5. Policies used in the main validated-profile comparison.

Policy	Type	Purpose in Evaluation
MADDPG	Proposed method	Multi-agent controller with centralized critics and behavior-adjusted training rewards
IDDPG	Learning variant	Trains traffic-pair agents independently without centralized joint-action evaluation
Single-DDPG	Learning variant	Replaces the per-pair multi-agent decomposition with one learned controller
CQL	Offline-RL baseline	Applies conservative Q-value regularization to per-agent discrete candidate selection
BCQ	Offline-RL baseline	Restricts per-agent candidate selection using a learned behavior-support model
Round-robin	Heuristic baseline	Cycles through the retained candidates
Shortest-hops	Heuristic baseline	Selects the feasible retained candidate with minimum hop count
Min-switch-cost	Heuristic baseline	Selects the retained candidate with minimum switch cost
Util-aware SP	Heuristic baseline	Selects the retained candidate with minimum observed bottleneck utilization

The heuristic baselines provide non-learning reference points. Round-robin cycles through the retained candidates and therefore tests whether unconditioned path variation is sufficient. Shortest-hops selects the feasible retained candidate with minimum hop count and is expected to be strong when topology structure favors direct routes. Min-switch-cost selects the retained candidate with minimum switch cost and provides a second topology-aware heuristic. Util-aware SP selects the retained candidate with the lowest observed bottleneck utilization and provides a congestion-aware heuristic reference. These baselines are important because a learned SDN controller should be compared not only with other learned formulations but also with simple, deployable routing rules.

The behavior-adjustment sensitivity variant, denoted MADDPG $\beta = 0$, is not part of the nine-policy primary comparison. It preserves the MADDPG architecture but removes the behavior-deviation adjustment from the training reward. The comparison uses three matched seeds on fat-tree and mesh-grid and ten matched seeds on WAN-corridors. It is therefore used to assess the direction and magnitude of the behavior-adjustment effect rather than as an additional policy in the main ranking.

This comparison set addresses three questions. First, comparison with IDDPG and Single-DDPG evaluates alternative actor–critic architectures. Second, comparison with CQL and BCQ evaluates established offline-RL formulations under the same local state and candidate-action abstraction. Third, comparison with Round-robin, Shortest-hops, Min-switch-cost, and Util-aware SP tests whether learned control improves upon simple heuristic path-selection rules under the same SDN deployment conditions.

2.8. Evaluation Metrics and Statistical Analysis

The evaluation uses reward, physical network measurements, policy-behavior metrics, and paired statistical tests. The primary scalar metric is the episode-level reward defined in Section 2.5. Higher reward is better, and policy rankings are computed independently within each topology and evaluation setting. Because the reward is a scalarization of multiple measured quantities, the analysis also reports reward-independent outcomes, particularly end-to-end RTT, bottleneck utilization, and flow completion.

Flow completion measures the fraction of dispatched flows that terminate successfully:

$$\text{Completion} = \frac{\#\{\text{dispatched flows with exit code } 0\}}{\#\{\text{dispatched flows}\}}.$$

This metric is reported separately from the per-step drop-related reward term. It is used to qualify reward comparisons when policies carry different realized traffic volumes, particularly on WAN-corridors.

For each of the three topologies, the primary validated-profile comparison uses ten paired seeds for MADDPG and the eight comparison policies. A seed is paired across policies when they are evaluated under the same topology and offered-traffic profile. This pairing is used whenever seed-level inferential comparisons are reported. For a comparison policy b, the paired reward gap on seed i is defined as

$$\Delta_i^{(b)} = \bar{R}_i^{\text{MADDPG}} - \bar{R}_i^{b},$$

where $\bar{R}_i^{\text{MADDPG}}$ and $\bar{R}_i^{b}$ are the episode-level rewards of MADDPG and comparison policy b on the same seed. A positive value therefore indicates that MADDPG achieved a higher reward on that seed. The mean paired gap is

$$\bar{\Delta}^{(b)} = \frac{1}{n}\sum_{i=1}^{n}\Delta_i^{(b)},$$

where $n = 10$ for the main validated-profile paired comparisons.

The primary ten-seed MADDPG–comparison-policy reward analysis reports several complementary quantities rather than relying on a single p-value. A percentile bootstrap 95% confidence interval for the mean paired reward gap is computed from 10,000 resamples of the seed-level gaps [40]. The exact two-sided paired permutation test evaluates all 2^n sign assignments of the paired gaps [41]. An exact two-sided sign test is also reported, with zero differences excluded from the effective sample size. Finally, Cliff's δ is reported as a non-parametric dominance effect-size estimate [42]. These quantities are interpreted conservatively as evidence about effect direction, uncertainty, and seed-level consistency rather than as proof of universal dominance.

Additional paired tests are used for selected physical-metric, sensitivity, and auxiliary comparisons. Paired two-sided t-tests and paired Wilcoxon signed-rank tests are reported for flow completion, bottleneck utilization, RTT, and the ten-seed WAN-corridors behavior-adjustment comparison. The Wilcoxon signed-rank tests are two-sided, with zero paired differences omitted. Paired two-sided t-tests alone are used for the three-seed fat-tree and mesh-grid behavior-adjustment comparisons, the three-seed learning-rate, discount-factor, and hidden-width sensitivity tests, and the three-seed ECMP comparison. The bursty-traffic, mixed-traffic, link-down, and packet-loss experiments are summarized descriptively without inferential testing. All reported p-values are nominal and unadjusted; no multiplicity correction is applied, and results based on $n = 3$ are treated as exploratory.

In addition to reward and paired statistics, the evaluation includes policy-behavior metrics that describe how each policy routes traffic. The average hop count measures the mean selected path length. The longer-than-minimum-hop selection fraction measures how often the selected candidate has a larger hop count than the retained minimum-hop candidate for the same traffic pair. The path-switch frequency measures the fraction of control steps in which a selected path changes relative to the previous step. The normalized

path-selection entropy measures how widely a policy distributes selections across the $K = 3$ candidate paths:

$$H_i = -\frac{1}{\log K} \sum_{k=1}^{K} p_{i,k} \log p_{i,k},$$

where $p_{i,k}$ is the empirical probability that traffic-pair agent i selects candidate path k. The reported entropy is averaged across traffic pairs. Values near zero indicate nearly deterministic path selection, while values near one indicate broad use of the candidate-path set.

Controller deployability is evaluated at three levels. First, the isolated NumPy-policy benchmark measures the time required to select paths jointly for all controlled traffic pairs, reporting the mean, median, and 95th-percentile decision time. Second, the instrumented full control loop reports the total elapsed interval and its principal stages: network-statistics polling, state construction, policy decision, forwarding-rule installation, and OpenFlow barrier synchronization. This distinction prevents isolated neural-network inference time from being interpreted as complete controller latency. Third, a scalability microbenchmark varies the number of traffic-pair agents up to $N = 64$ and the number of candidate paths up to $K = 10$, recording the joint policy-decision time for each configuration. Model parameter count and serialized model size are reported alongside these timing measurements.

Table 6 summarizes the metrics and their role in the evaluation.

Table 6. Evaluation metrics and their roles in the study.

Metric Group	Role in Evaluation
Mean episode reward	Primary scalar objective; higher values indicate lower evaluated routing cost
Rank and average rank	Descriptive cross-topology rank profile across the three evaluated fabrics
RTT and bottleneck utilization	Reward-independent latency and congestion measurements
Flow completion	Fraction of dispatched flows terminating with exit code 0; qualifies comparisons at unequal delivered traffic
Hop count and longer-than-minimum-hop fraction	Selected-path length and use of candidates longer than the retained minimum-hop candidate
Path-switch frequency and entropy	Temporal stability and distribution of selections across the candidate catalogue
Paired reward gap	Seed-matched MADDPG–comparison-policy reward comparison
Bootstrap confidence interval	Uncertainty of the mean paired reward gap
Permutation and sign tests	Paired significance and seed-level win consistency
Paired t- and Wilcoxon tests	Selected physical-metric and behavior-adjustment comparisons; three-seed analyses use the paired t-test only
Cliff's δ	Non-parametric effect-size estimate
Isolated policy timing	Mean, median, and 95th-percentile joint path-selection latency
Full-loop stage timing	Polling, state construction, policy decision, rule installation, barrier synchronization, and total elapsed interval
Scalability timing	Joint decision latency as the numbers of agents N and candidate paths K increase
Model size	Parameter count and serialized policy footprint

For the cross-topology rank analysis, each policy receives one rank per topology under the deployed validated-profile reward configuration. The average rank of policy p is computed as

$$\mathrm{AvgRank}(p) = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \mathrm{Rank}_{\tau}(p),$$

where

$$\mathcal{T} = \{\text{fat-tree}, \text{mesh-grid}, \text{WAN-corridors}\}$$

and $|\mathcal{T}| = 3$. This rank-based summary is useful because the three topologies have different structures and absolute reward scales. Average rank is interpreted descriptively as a compact cross-topology profile rather than as evidence of statistical dominance.

2.9. Auxiliary Evaluation Protocols

The auxiliary evaluations examine traffic-pattern sensitivity, link failure, packet loss, and an ECMP forwarding reference on the fat-tree topology. Each condition uses three matched seeds and the same candidate-generation procedure and $K = 3$ action cardinality as the corresponding live evaluation. Table 7 reports the protocol for each condition. The traffic, link-down, and packet-loss results are interpreted descriptively because their three-seed sample sizes are not intended to support the ten-seed inferential claims of the main comparison.

Table 7. Protocols used in the auxiliary fat-tree evaluations. Each condition uses three matched seeds.

Condition	Evaluation Protocol
Bursty traffic	User Datagram Protocol (UDP) traffic with elephant rates of 6–9 Mbit/s, durations of 4–6 s, and inter-arrival gaps of 10–14 s. The traffic-shock probability is 0.15, with a rate multiplier sampled from 2.0–3.0. This condition changes the stochastic flow-generation distributions rather than imposing a fixed burst start time, duration, or period.
Mixed traffic	Two UDP flow components are used. Mice flows have rates of 0.5–2 Mbit/s, durations of 2–4 s, and inter-arrival gaps of 1–3 s. Elephant flows have rates of 3–4 Mbit/s, durations of 30–60 s, and inter-arrival gaps of 10–20 s. The elephant probability is 0.35, and the maximum number of concurrent flows is 10.
Link down	One bidirectional fat-tree fabric link is disabled at $t = 200$ s and remains unavailable until the end of the episode. The switches remain active, and the resulting topology-discovery event causes the controller to update the topology and recompute candidate paths.
Packet loss	A 1% packet-loss rate is applied bidirectionally to every switch-to-switch fabric link from $t = 60$ s until the end of the episode.
ECMP	Static per-flow ECMP assigns each flow to one of the feasible retained candidates using a destination-port-based hash. The assignment remains fixed for the duration of the flow.

These tests do not implement explicit application service classes, switch failure, link jitter, packet-level multipath, or learned cooperative multipath control.

Complete controller latency is measured separately from isolated policy inference. One instrumented run is collected for each topology, recording network-statistics polling, state construction, policy decision, forwarding-rule installation, OpenFlow barrier synchronization, and total control-loop elapsed time. The measurement is used to identify whether policy computation or the surrounding SDN workflow dominates the practical control interval.

Finally, an offline scalability microbenchmark varies the number of traffic-pair agents and candidate paths according to

$$N \in \{2, 4, 8, 16, 32, 64\}, \qquad K \in \{3, 5, 10\}.$$

For each configuration, the benchmark measures the latency of one joint policy decision across all agents. It excludes topology discovery, network monitoring, candidate-path generation, and forwarding-rule installation, and therefore isolates the computational scaling of the deployed policy-decision stage.

3. Results

This section evaluates the proposed offline MADDPG controller with behavior-adjusted training rewards in terms of comparative routing performance, sensitivity across operating conditions, and practical deployability. Unless stated otherwise, the main validated-profile comparison uses ten paired seeds per topology and the reward configuration $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$. The analysis compares MADDPG with eight learning-based and heuristic comparison policies on the fat-tree, mesh-grid, and WAN-corridors topologies, and complements scalar reward with flow-completion, round-trip-time, policy-behavior, architectural-variant, sensitivity, and paired statistical analyses. Additional experiments examine traffic-pattern variation, link-down and packet-loss conditions, single-path and static per-flow ECMP forwarding, controller-loop overhead, scalability, and the operating envelope across offered-traffic profiles. The results do not support universal superiority of MADDPG: it is the best-performing learned policy on the fat-tree topology, no evaluated policy significantly outperforms it on the mesh-grid topology, and it remains statistically tied with the completion-matched policy group on WAN-corridors, whereas the utilization-aware path heuristic provides the strongest overall reward ranking.

3.1. Main Validated-Profile Performance

Figure 3 summarizes the validated-profile reward results for the nine evaluated policies on the fat-tree, mesh-grid, and WAN-corridors topologies. Reward is reported on its original scale, where higher values are better and zero is the best attainable value. The results are topology-dependent. Util-aware SP obtains the highest mean reward on the fat-tree and WAN-corridors topologies, while Shortest-hops obtains the highest mean reward on the mesh-grid topology. MADDPG ranks second on fat-tree and is the best-performing learned policy on that topology, but it does not obtain the highest point-estimate reward on mesh-grid or WAN-corridors.

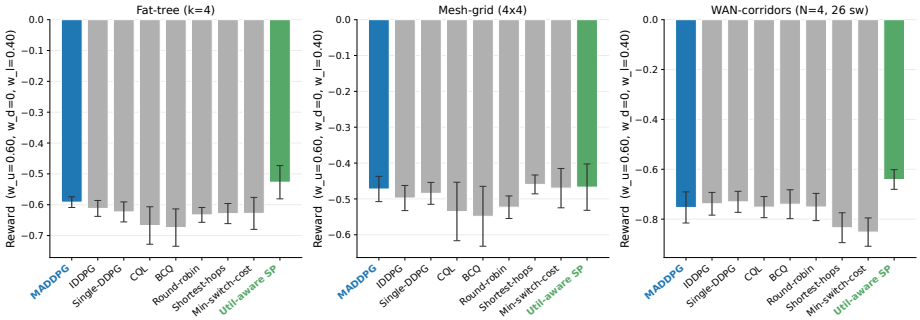


Figure 3. Validated-profile reward by policy on the fat-tree, mesh-grid, and WAN-corridors topologies. Bars show the mean episode reward and error bars show one standard deviation over ten matched seeds. Rewards were computed with $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$; higher reward is better and zero is the best attainable value.

The corresponding mean rewards and point-estimate ranks are reported in Table 8. On the fat-tree topology, Util-aware SP obtains the highest mean reward, -0.527, followed by MADDPG at -0.592. MADDPG nevertheless outperforms every other learning-based policy in this topology: IDDPG, Single-DDPG, CQL, and BCQ obtain mean rewards of -0.612, -0.623, -0.667, and -0.674, respectively. Thus, the extended comparison supports the narrower conclusion that MADDPG is the strongest learned controller on the path-diverse fat-tree, rather than the strongest policy overall.

Table 8. Validated-profile mean reward and point-estimate rank for the nine evaluated policies. The rank within each topology is shown in parentheses; rank 1 is best. Average rank is calculated across the three topologies.

Policy	Fat-Tree	Mesh-Grid	WAN-Corridors	Average Rank
MADDPG	−0.592 (2)	−0.472 (4)	−0.753 (7)	4.33
IDDPG	−0.612 (3)	−0.498 (6)	−0.738 (3)	4.00
Single-DDPG	−0.623 (4)	−0.484 (5)	−0.731 (2)	3.67
CQL	−0.667 (8)	−0.535 (8)	−0.752 (6)	7.33
BCQ	−0.674 (9)	−0.548 (9)	−0.740 (4)	7.33
Round-robin	−0.633 (7)	−0.523 (7)	−0.751 (5)	6.33
Shortest-hops	−0.629 (6)	−0.460 (1)	−0.834 (8)	5.00
Min-switch-cost	−0.628 (5)	−0.470 (3)	−0.852 (9)	5.67
Util-aware SP	−0.527 (1)	−0.467 (2)	−0.641 (1)	1.33

On the mesh-grid topology, Shortest-hops has the highest point-estimate reward, −0.460, followed by Util-aware SP at −0.467, Min-switch-cost at −0.470, and MADDPG at −0.472. These four policies form a closely spaced group, and the subsequent paired analysis shows that none significantly outperforms MADDPG on this topology. On WAN-corridors, Util-aware SP obtains the highest mean reward, −0.641. MADDPG has a point-estimate rank of seven with a mean reward of −0.753, but the policies occupying ranks two through seven span only 0.022 reward units and are interpreted jointly with paired significance and flow-completion results in the subsequent analyses. Across the three topologies, Util-aware SP has the best average rank, 1.33. The main validated-profile comparison therefore does not support describing MADDPG as a universally superior or robust-generalist policy; it instead shows that its relative performance depends on topology and on the comparison with topology-aware heuristics.

3.2. Cross-Topology Rank Profile

Figure 4 presents the validated-profile reward-rank profiles of the nine evaluated policies across the fat-tree, mesh-grid, and WAN-corridors topologies. Lower ranks indicate higher mean reward within a topology. The profiles provide a descriptive summary of how policy ordering changes across network structures; they do not establish statistical separation between adjacent ranks, which is assessed using the paired seed-level analysis reported later in this section.

Across the three topologies, Util-aware SP has the strongest descriptive average rank (1.33), followed by Single-DDPG (3.67), IDDPG (4.00), and MADDPG (4.33). These averages summarize point-estimate ordering only and do not account for paired uncertainty or differences in flow completion. MADDPG is the strongest learned policy on fat-tree, no evaluated policy significantly outperforms it on mesh-grid, and the WAN-corridors ordering must be interpreted jointly with flow completion. The cross-topology rank profile therefore indicates topology-dependent policy effectiveness rather than uniform dominance.

3.3. Completion and Utilization Qualification

The scalar reward does not directly account for the fraction of dispatched flows that complete successfully. In particular, the validated configuration uses $w_d = 0$, and the drop-related reward ingredient is distinct from end-to-end flow completion. A policy may therefore obtain a higher reward by operating under lower realized traffic, even when the reduction results from fewer completed flows. For this reason, reward differences are interpreted jointly with flow completion and bottleneck utilization, especially on the WAN-corridors topology. Flow completion is treated as an observed outcome of the closed-loop traffic process rather than as a matched input condition.

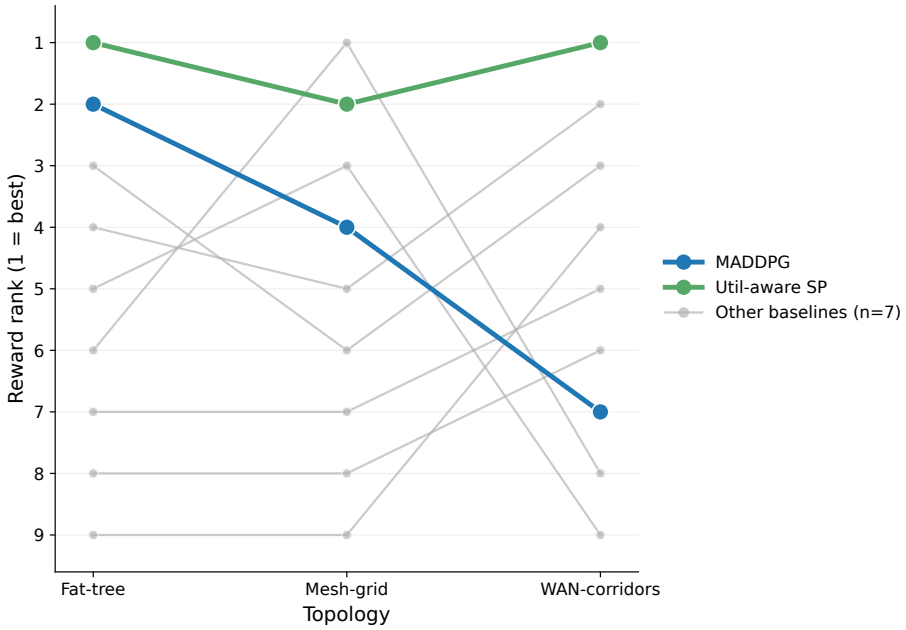


Figure 4. Validated-profile reward-rank profiles across the fat-tree, mesh-grid, and WAN-corridors topologies. Rank 1 indicates the highest mean reward within a topology. The ranks are descriptive point estimates; statistical separation is evaluated using paired seed-level comparisons.

Figure 5 shows the relationship among mean flow completion, bottleneck utilization, and reward on WAN-corridors. The six policies comprising MADDPG, IDDPG, Single-DDPG, CQL, BCQ, and Round-robin form a completion-matched group with mean completion between 89.5% and 91.9%. Within this group, none of the five comparison policies has a statistically significant reward advantage over MADDPG. In contrast, Util-aware SP, Shortest-hops, and Min-switch-cost complete substantially less traffic than MADDPG.

Table 9 reports the corresponding WAN-corridors measurements. MADDPG completes 90.2% of dispatched flows with a mean bottleneck utilization of 0.759. Util-aware SP obtains the highest reward and the lowest utilization, but its completion falls to 76.3%, a paired deficit of 13.9 percentage points relative to MADDPG. Shortest-hops and Min-switch-cost complete 72.5% and 73.3%, respectively, while also obtaining lower rewards than MADDPG. The completion deficits of these three policies relative to MADDPG are significant under both paired tests. Thus, the WAN reward differences involving these policies do not represent comparisons at matched delivered load.

The mesh-grid results in Table 10 show a different pattern. Shortest-hops, Util-aware SP, and Min-switch-cost have higher point-estimate rewards than MADDPG, but none significantly outperforms MADDPG in the paired reward analysis. No evaluated policy also differs significantly from MADDPG in mean completion or bottleneck utilization. The comparatively low mean completion of Shortest-hops, 79.8%, is driven by high seed-level variability, with a standard deviation of 25.7 percentage points and a minimum episode-level completion of 33.8%, whereas MADDPG has a minimum of 93.3%. The mesh-grid ordering should therefore be interpreted as a statistically unresolved near-top group rather than as evidence that one of these policies consistently dominates MADDPG.

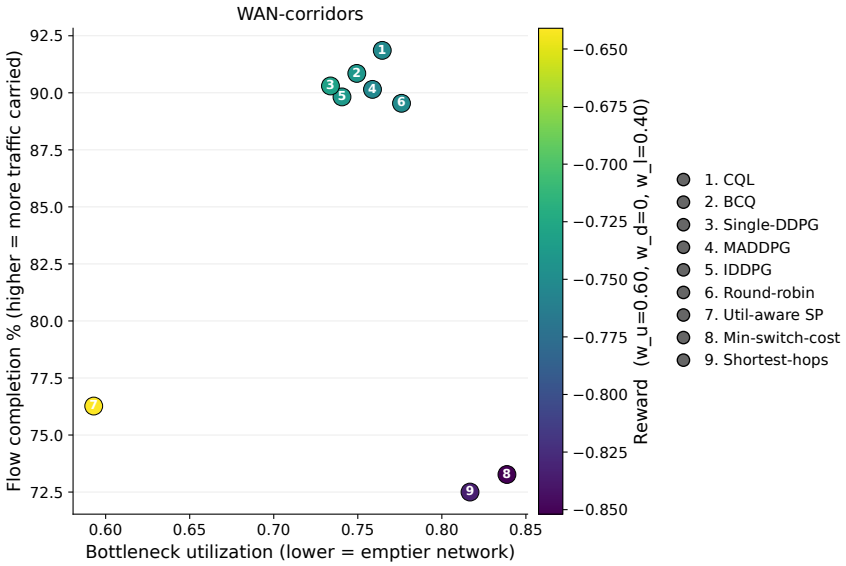


Figure 5. Flow completion and bottleneck utilization on WAN-corridors under the validated-profile setting. Each point represents the mean over ten matched seeds, and point color indicates the mean reward computed with $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$. Higher completion and lower utilization are preferable. Numbered markers are identified in the accompanying key.

Table 9. Flow completion, bottleneck utilization, and reward on WAN-corridors under the validated-profile setting. Values are mean ± standard deviation over ten matched seeds.

Policy	Completion (%)	Bottleneck Utilization	Mean Reward
MADDPG	90.2 ± 4.2	0.759 ± 0.084	-0.753
IDDPG	89.8 ± 5.2	0.741 ± 0.070	-0.738
Single-DDPG	90.3 ± 3.8	0.734 ± 0.060	-0.731
CQL	91.9 ± 4.7	0.765 ± 0.059	-0.752
BCQ	90.9 ± 4.3	0.749 ± 0.077	-0.740
Round-robin	89.5 ± 4.5	0.776 ± 0.079	-0.751
Shortest-hops	72.5 ± 11.5	0.817 ± 0.082	-0.834
Min-switch-cost	73.3 ± 9.4	0.839 ± 0.062	-0.852
Util-aware SP	76.3 ± 9.3	0.593 ± 0.054	-0.641

Table 10. Flow completion, bottleneck utilization, and reward on the mesh-grid topology under the validated-profile setting. Values are mean ± standard deviation over ten matched seeds.

Policy	Completion (%)	Bottleneck Utilization	Mean Reward
MADDPG	95.5 ± 1.6	0.735 ± 0.054	-0.472
IDDPG	95.9 ± 1.2	0.719 ± 0.050	-0.498
Single-DDPG	96.3 ± 0.9	0.722 ± 0.040	-0.484
CQL	92.7 ± 5.1	0.744 ± 0.095	-0.535
BCQ	92.5 ± 4.7	0.755 ± 0.095	-0.548
Round-robin	95.5 ± 0.7	0.750 ± 0.041	-0.523
Shortest-hops	79.8 ± 25.7	0.722 ± 0.037	-0.460
Min-switch-cost	90.6 ± 14.4	0.720 ± 0.068	-0.470
Util-aware SP	92.5 ± 5.2	0.677 ± 0.068	-0.467

The completion qualification does not alter the fat-tree comparison. MADDPG and Util-aware SP have statistically indistinguishable completion on fat-tree (90.2% and 89.0%, respectively) under both paired tests. Their reward difference is therefore evaluated at comparable delivered load and is examined through the policy-behavior and reward-component analyses reported below.

3.4. Reward-Independent RTT Validation

The reward comparisons in Sections 3.1–3.3 are complemented by a reward-independent physical metric: measured end-to-end round-trip time (RTT). Unlike the scalar reward, RTT does not depend on the selected reward weights or on the hop/switch-penalty calibration. Figure 6 reports the mean RTT of all nine policies on the fat-tree, mesh-grid, and WAN-corridors topologies.

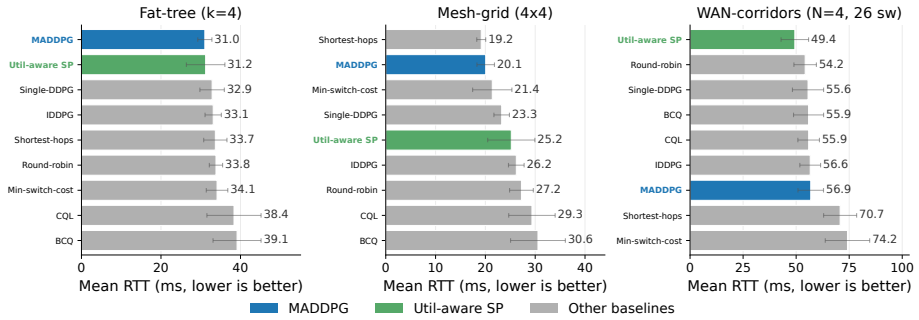


Figure 6. Mean end-to-end RTT under the validated-profile setting on the fat-tree, mesh-grid, and WAN-corridors topologies. Values are averaged over ten matched seeds; lower RTT is better. MADDPG and Util-aware SP are highlighted, while the remaining policies are shown as comparison policies.

The corresponding values and point-estimate RTT ranks are reported in Table 11. On the fat-tree topology, MADDPG obtains the lowest mean RTT, 31.0 ms, followed closely by Util-aware SP at 31.2 ms and Single-DDPG at 32.9 ms. The paired analysis shows that the RTT difference between MADDPG and Util-aware SP is not statistically significant. Consequently, MADDPG provides the lowest measured latency on fat-tree, but its lower RTT does not explain the reward advantage of Util-aware SP, which is examined through the utilization and policy-behavior analyses below.

Table 11. Mean end-to-end RTT and point-estimate RTT rank under the validated-profile setting. Values are averaged over ten matched seeds; lower RTT and rank are better. The number in parentheses preceded by # is the point-estimate RTT rank within that topology.

Policy	Fat-Tree	Mesh-Grid	WAN-Corridors
MADDPG	31.0 (#1)	20.1 (#2)	56.9 (#7)
IDDPG	33.1 (#4)	26.2 (#6)	56.6 (#6)
Single-DDPG	32.9 (#3)	23.3 (#4)	55.6 (#3)
CQL	38.4 (#8)	29.3 (#8)	55.9 (#5)
BCQ	39.1 (#9)	30.6 (#9)	55.9 (#4)
Round-robin	33.8 (#6)	27.2 (#7)	54.2 (#2)
Shortest-hops	33.7 (#5)	19.2 (#1)	70.7 (#8)
Min-switch-cost	34.1 (#7)	21.4 (#3)	74.2 (#9)
Util-aware SP	31.2 (#2)	25.2 (#5)	49.4 (#1)

On mesh-grid, Shortest-hops obtains the lowest mean RTT, 19.2 ms, followed by MADDPG at 20.1 ms and Min-switch-cost at 21.4 ms. This ordering is consistent with the limited path diversity of the topology, where direct routes provide a strong latency reference and additional detours offer little measured benefit. On WAN-corridors, Util-aware SP obtains the lowest mean RTT, 49.4 ms, while MADDPG records 56.9 ms. This WAN latency advantage must, however, be interpreted together with the completion analysis: Util-aware SP completes 13.9 percentage points less traffic than MADDPG. Shortest-hops and Min-switch-cost also complete substantially less traffic, but nevertheless produce the two highest WAN RTT values, 70.7 ms and 74.2 ms, respectively. The RTT results therefore confirm that

the latency outcome is topology- and policy-dependent, rather than providing uniform physical validation of the scalar reward ranking.

3.5. Architectural Variant Comparison

The full MADDPG controller is compared with the implemented IDDPG and Single-DDPG variants under the validated-profile setting. IDDPG trains traffic-pair agents independently without centralized joint-action evaluation, whereas Single-DDPG uses one learned controller for the joint path-control decision. The target-projection configuration is not identical across all variants: IDDPG and Single-DDPG use empirical target projection on all three topologies, whereas the primary MADDPG model uses it only on WAN-corridors. These comparisons are therefore treated as implementation-level architectural variants rather than as perfectly controlled one-factor ablations. Figure 7 reports the paired reward difference between each variant and MADDPG. Negative values indicate that the variant obtains lower reward, whereas positive values indicate a higher point-estimate reward. Behavior adjustment is examined separately in Section 3.6.

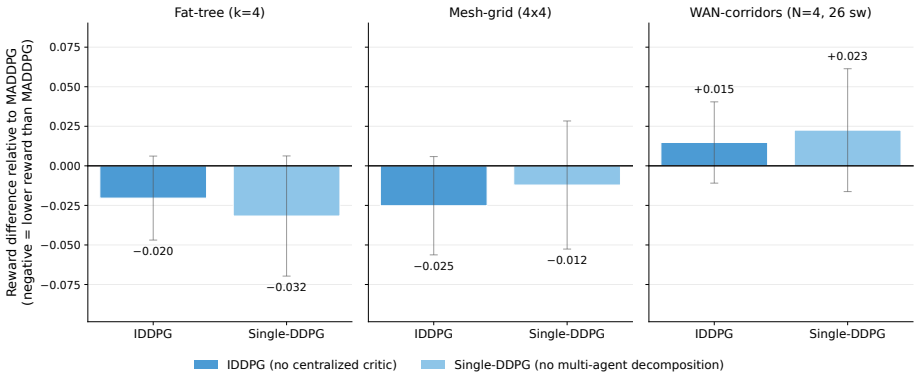


Figure 7. Implementation-level architectural comparisons under the validated-profile setting. Bars show the mean paired reward difference relative to MADDPG over ten matched seeds, and error bars show one standard deviation of the paired differences. Negative values indicate lower reward than MADDPG. IDDPG uses independent traffic-pair critics, whereas Single-DDPG uses one joint controller; target-projection settings are not matched on fat-tree and mesh-grid.

The numerical paired differences between the implemented variants and MADDPG are reported in Table 12. On fat-tree, IDDPG and Single-DDPG obtain point-estimate differences of -0.0204 and -0.0317, respectively, relative to MADDPG. On mesh-grid, the corresponding differences are smaller: -0.0252 for IDDPG and -0.0121 for Single-DDPG. These values describe the implemented configurations and should not be interpreted as isolated causal effects because target projection is unmatched on these two topologies.

The WAN-corridors topology does not show the same direction. IDDPG and Single-DDPG obtain point-estimate reward differences of $+0.0148$ and $+0.0226$, respectively, relative to MADDPG. The subsequent paired statistical analysis does not separate either variant significantly from MADDPG on this topology. This comparison is less affected by the target-projection qualification because projection is enabled for MADDPG, IDDPG, and Single-DDPG on WAN-corridors. The results therefore do not support a topology-independent claim that the implemented IDDPG and Single-DDPG variants improve or degrade reward consistently. They show lower point-estimate rewards than MADDPG on fat-tree, smaller and less uniform differences on mesh-grid, and no established separation on WAN-corridors. Because target-projection settings are unmatched on fat-tree and mesh-grid, these differences cannot be attributed exclusively to centralized training or multi-agent decomposition.

Table 12. Implementation-level architectural comparisons under the validated-profile setting. The reward difference is computed within each matched seed as the variant reward minus the MADDPG reward. Negative values indicate lower reward than MADDPG.

Topology	Implemented Variant	Mean Difference	Standard Deviation	Seeds
Fat-tree	IDDPG (independent traffic-pair critics)	−0.0204	0.0265	10
Fat-tree	Single-DDPG (one joint controller)	−0.0317	0.0380	10
Mesh-grid	IDDPG (independent traffic-pair critics)	−0.0252	0.0311	10
Mesh-grid	Single-DDPG (one joint controller)	−0.0121	0.0405	10
WAN-corridors	IDDPG (independent traffic-pair critics)	+0.0148	0.0257	10
WAN-corridors	Single-DDPG (one joint controller)	+0.0226	0.0388	10

3.6. Behavior-Adjustment and Hyperparameter Sensitivity

The architectural comparisons in Section 3.5 are complemented by an analysis of behavior adjustment and the main training hyperparameters. The selected configuration uses a learning rate of 10^{-3}, a discount factor of $\gamma = 0.99$, 128 hidden units per layer, a behavior-adjustment weight of $\beta = 0.1$, and 30,000 training steps. Each sensitivity variant changes one parameter while retaining the same offline dataset, training budget, and reward configuration $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$. The architecture sensitivity changes only the hidden-layer width; the number of hidden layers is not varied. Target projection is held fixed within each β comparison: it is disabled for both variants on fat-tree and mesh-grid and enabled for both variants on WAN-corridors.

The clearest behavior-adjustment effect is observed on WAN-corridors. Figure 8 compares MADDPG with the selected $\beta = 0.1$ configuration and with the behavior-deviation reward adjustment removed, $\beta = 0$, using the same ten matched seeds. Reward, flow completion, bottleneck utilization, and RTT are reported jointly because a WAN reward improvement would be ambiguous if it resulted from carrying less traffic.

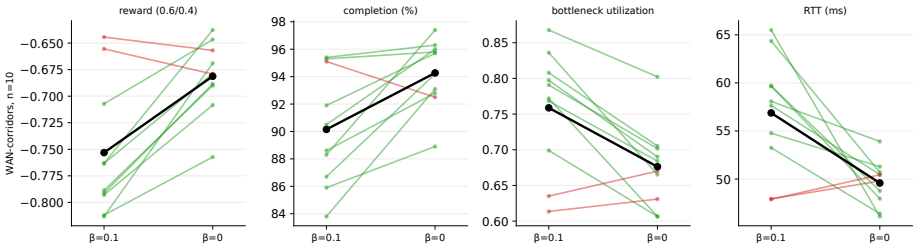


Figure 8. Paired WAN-corridors comparison of MADDPG with the selected behavior-adjustment weight $\beta = 0.1$ and with the behavior-deviation reward adjustment removed, $\beta = 0$. Thin lines connect matched seeds, and the black line shows the mean. Green lines denote seeds that moved in the preferred direction of the plotted metric, and red lines denote seeds that did not. Higher reward and completion are better, whereas lower bottleneck utilization and RTT are better. Target projection is enabled for both variants.

Table 13 reports the corresponding paired values. Removing the behavior-deviation reward adjustment improves all four WAN measurements in the same direction. Mean reward increases from −0.753 to −0.681, while flow completion increases from 90.2% to 94.3%. At the same time, mean bottleneck utilization decreases from 0.759 to 0.676, and mean RTT decreases from 56.9 ms to 49.6 ms. The reward improvement is therefore not produced by traffic shedding: the $\beta = 0$ controller completes more traffic while operating at lower utilization and latency.

This result is interpreted as a within-method sensitivity result rather than as an additional policy ranking. The $\beta = 0$ MADDPG variant is not compared against policies that retained their selected $\beta = 0.1$ configurations. Within each topology, the two β variants

use matched target-projection settings: projection is disabled for both fat-tree and mesh-grid variants and enabled for both WAN-corridors variants. Across topologies, the paired reward difference between $\beta = 0$ and $\beta = 0.1$ is -0.043 on fat-tree ($p_t = 0.133$, $n = 3$), $+0.000$ on mesh-grid ($p_t = 0.996$, $n = 3$), and $+0.072$ on WAN-corridors ($p_t = 0.002$, $p_W = 0.010$, $n = 10$). Here, p_t denotes the two-sided paired t-test and p_W denotes the two-sided Wilcoxon signed-rank test. Only the WAN-corridors comparison establishes a significant behavior-adjustment effect; the fat-tree and mesh-grid results are exploratory three-seed comparisons without significant separation.

Table 13. Paired WAN-corridors comparison of MADDPG with $\beta = 0.1$ and $\beta = 0$. The difference is computed as the $\beta = 0$ value minus the $\beta = 0.1$ value. "Improved seeds" accounts for the preferred direction of each metric. The displayed p_t values are from two-sided paired t-tests.

Metric	$\beta = 0.1$	$\beta = 0$	Difference	Paired t-Test p_t	Improved Seeds
Reward	-0.753	-0.681	$+0.072$	0.0024	8/10
Flow completion (%)	90.2	94.3	$+4.1$	0.0083	9/10
Bottleneck utilization	0.759	0.676	-0.082	0.0035	8/10
RTT (ms)	56.9	49.6	-7.3	0.0084	8/10

Figure 9 extends the analysis to learning rate, discount factor, and hidden-layer width. These parameters are evaluated on fat-tree with three matched seeds, while the behavior-adjustment comparison is shown on all three topologies. Each plotted value is the paired reward difference relative to the selected configuration. Table 14 reports the corresponding numerical values.

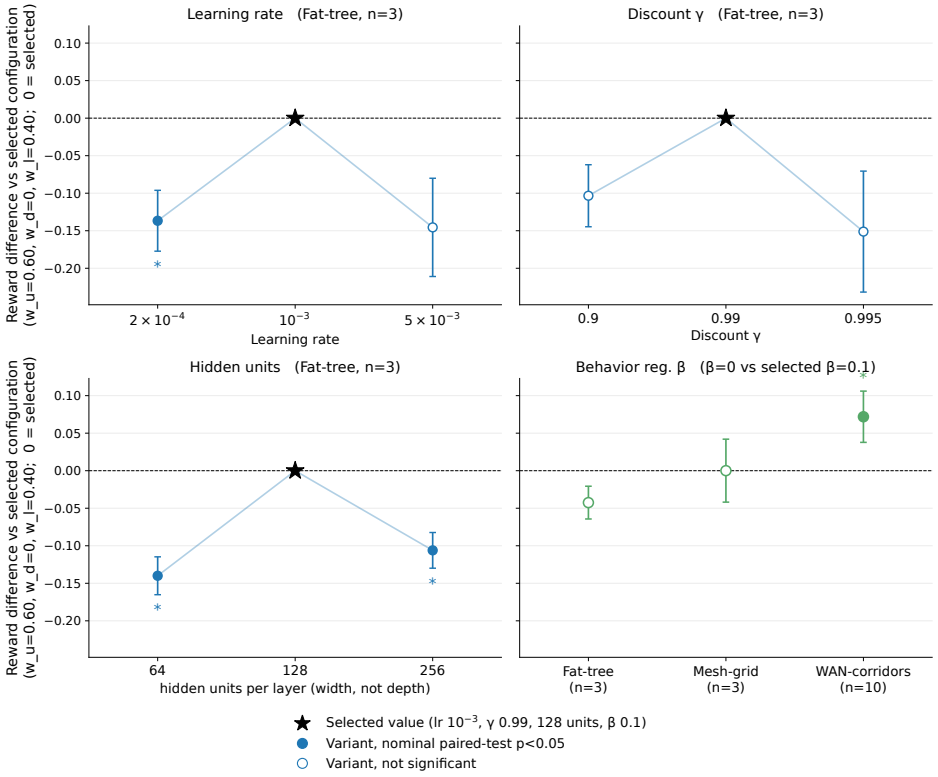


Figure 9. Sensitivity of MADDPG to learning rate, discount factor, hidden-layer width, and behavior-adjustment weight. Values show the paired reward difference relative to the selected configuration under $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$. A value of zero denotes the selected configuration. The black star marks the selected configuration in each panel. Filled markers with an asterisk indicate a nominal two-sided paired t-test result of $p_t < 0.05$.

Table 14. MADDPG hyperparameter sensitivity. Each variant changes one parameter relative to the selected configuration. The difference is the variant reward minus the selected-configuration reward on the same matched seeds. For the three-seed comparisons, p_t denotes the two-sided paired t-test. For the ten-seed WAN-corridors behavior-adjustment comparison, both the paired t-test p_t and Wilcoxon signed-rank test p_W are reported.

Parameter	Value	Topology	Mean Reward	Standard Deviation	Seeds	Difference and Paired Test
Learning rate	10^{-3}	Fat-tree	−0.605	0.013	3	Reference
Learning rate	2×10^{-4}	Fat-tree	−0.742	0.041	3	-0.137, $p_t = 0.030$
Learning rate	5×10^{-3}	Fat-tree	−0.751	0.065	3	-0.146, $p_t = 0.083$
Discount factor	0.99	Fat-tree	−0.605	0.013	3	Reference
Discount factor	0.9	Fat-tree	−0.709	0.041	3	-0.103, $p_t = 0.055$
Discount factor	0.995	Fat-tree	−0.756	0.081	3	-0.151, $p_t = 0.073$
Hidden units	128	Fat-tree	−0.605	0.013	3	Reference
Hidden units	64	Fat-tree	−0.745	0.025	3	-0.140, $p_t = 0.022$
Hidden units	256	Fat-tree	−0.711	0.024	3	-0.106, $p_t = 0.022$
Behavior adjustment	$\beta = 0.1$	Fat-tree	−0.605	0.013	3	Reference
Behavior adjustment	$\beta = 0$	Fat-tree	−0.648	0.022	3	-0.043, $p_t = 0.133$
Behavior adjustment	$\beta = 0.1$	Mesh-grid	−0.497	0.061	3	Reference
Behavior adjustment	$\beta = 0$	Mesh-grid	−0.497	0.042	3	$+0.000$, $p_t = 0.996$
Behavior adjustment	$\beta = 0.1$	WAN-corridors	−0.753	0.062	10	Reference
Behavior adjustment	$\beta = 0$	WAN-corridors	−0.681	0.034	10	$+0.072$, $p_t = 0.002$, $p_W = 0.010$

On fat-tree, none of the tested learning-rate, discount-factor, or hidden-width perturbations improves the point-estimate reward relative to the selected configuration. The alternative values reduce mean reward by approximately 0.10–0.15. The paired differences are nominally significant for the learning rate 2×10^{-4} and for both alternative hidden widths, although these comparisons contain only three matched seeds and are therefore interpreted cautiously. The discount-factor variants and the learning rate 5×10^{-3} show the same negative direction without significant separation.

The results indicate that each tested learning-rate, discount-factor, and hidden-width perturbation produces a lower fat-tree mean reward under the same data and training budget. However, these comparisons use only three matched seeds and are interpreted as exploratory. The architecture analysis varies hidden-layer width only and does not evaluate network depth. The behavior-adjustment effect is less uniform: removing the adjustment has no measurable effect on mesh-grid, produces a non-significant reward reduction on fat-tree, and significantly improves reward, completion, utilization, and RTT on WAN-corridors. The influence of β is therefore topology-dependent rather than uniformly beneficial across the evaluated environments.

3.7. Reward-Weight Sensitivity

The validated reward configuration uses $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$, giving greater weight to bottleneck utilization than to latency. To test whether the policy ordering depends on this specific scalarization, the same validated-profile trajectories were re-scored ex post under $w_u \in \{0.2, 0.4, 0.5, 0.6, 0.8\}$, with $w_d = 0$ and $w_l = 1 - w_u$. No policy was retrained: the measured utilization, latency, and realized hop/switch-penalty terms remained fixed, and only their relative weighting changed. The analysis covers all nine policies and all three topologies. Figure 10 shows the resulting mean rewards for every tested weighting.

The corresponding point-estimate ranks are reported in Table 15. On fat-tree, the leading order is unchanged across all five weight settings: Util-aware SP remains rank 1 and MADDPG remains rank 2. The difference between these two policies is not statistically significant at the latency-dominant settings $w_u = 0.2$ and $w_u = 0.4$, but becomes significant at the deployed setting $w_u = 0.6$ and at $w_u = 0.8$. Thus, the fat-tree ranking is stable with respect to the tested utilization–latency trade-off, but the analysis does not support the previous claim that MADDPG is the highest-ranked policy.

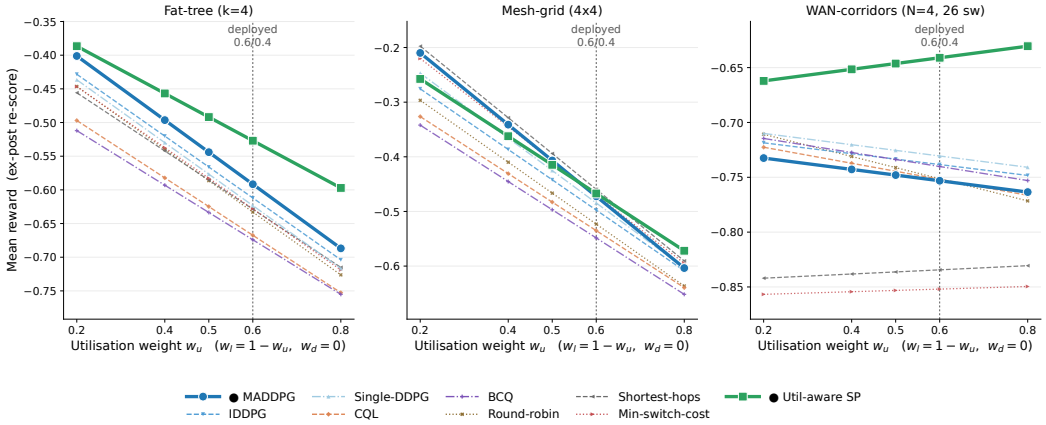


Figure 10. Reward-weight sensitivity under ex post re-scoring of the validated-profile trajectories. The utilization weight w_u is varied while $w_l = 1 - w_u$ and $w_d = 0$. The deployed configuration is $w_u/w_l = 0.6/0.4$. All nine policies are evaluated on the fat-tree, mesh-grid, and WAN-corridors topologies; higher reward is better.

Table 15. Reward-weight sensitivity ranks under ex post re-scoring. Rank 1 indicates the highest mean reward within a topology and weight setting. The deployed configuration is shown in bold.

Topology	w_u/w_l	MADDPG	IDDPG	Single	CQL	BCQ	RR	SH	MSC	UASP	Best policy
Fat-tree	0.2/0.8	2	3	4	8	9	5	7	6	1	Util-aware SP
Fat-tree	0.4/0.6	2	3	4	8	9	6	7	5	1	Util-aware SP
Fat-tree	0.5/0.5	2	3	4	8	9	7	6	5	1	Util-aware SP
Fat-tree	**0.6/0.4**	**2**	**3**	**4**	**8**	**9**	**7**	**6**	**5**	**1**	**Util-aware SP**
Fat-tree	0.8/0.2	2	3	5	8	9	7	4	6	1	Util-aware SP
Mesh-grid	0.2/0.8	2	6	4	8	9	7	1	3	5	Shortest-hops
Mesh-grid	0.4/0.6	2	6	5	8	9	7	1	3	4	Shortest-hops
Mesh-grid	0.5/0.5	2	6	5	8	9	7	1	3	4	Shortest-hops
Mesh-grid	**0.6/0.4**	**4**	**6**	**5**	**8**	**9**	**7**	**1**	**3**	**2**	**Shortest-hops**
Mesh-grid	0.8/0.2	5	6	4	8	9	7	2	3	1	Util-aware SP
WAN-corridors	0.2/0.8	7	5	2	6	4	3	8	9	1	Util-aware SP
WAN-corridors	0.4/0.6	7	4	2	6	3	5	8	9	1	Util-aware SP
WAN-corridors	0.5/0.5	7	3	2	6	4	5	8	9	1	Util-aware SP
WAN-corridors	**0.6/0.4**	**7**	**3**	**2**	**6**	**4**	**5**	**8**	**9**	**1**	**Util-aware SP**
WAN-corridors	0.8/0.2	5	3	2	6	4	7	8	9	1	Util-aware SP

RR: Round-robin; SH: Shortest-hops; MSC: Min-switch-cost; UASP: Util-aware SP.

The mesh-grid point-estimate ordering changes as utilization receives greater weight. Shortest-hops is rank 1 for $w_u \leq 0.6$, while Util-aware SP becomes rank 1 at $w_u = 0.8$. MADDPG ranks second at $w_u = 0.2$, 0.4, and 0.5, fourth at the deployed $w_u = 0.6$, and fifth at $w_u = 0.8$. These rank changes do not represent statistically established separations: at every tested weight, the paired comparison between MADDPG and the point-estimate winner is non-significant. The mesh-grid result should therefore be interpreted as movement within a statistically unresolved leading group rather than as evidence of a significant reward-weight-induced reordering.

On WAN-corridors, Util-aware SP remains rank 1 and Single-DDPG remains rank 2 under every tested weighting. MADDPG remains rank 7 for $w_u \leq 0.6$ and moves to rank 5 at $w_u = 0.8$, but it is never statistically tied with Util-aware SP. This WAN reward separation must still be interpreted together with the flow-completion qualification in Section 3.3, because changing the utilization–latency weights does not correct differences in delivered traffic.

Overall, MADDPG does not become the highest-ranked policy on any topology under any tested reward weighting. Util-aware SP remains rank 1 on fat-tree and WAN-corridors across the complete sweep and also becomes rank 1 on mesh-grid when utilization is weighted most heavily. The sensitivity analysis therefore shows that the revised main conclusions are not artifacts of the deployed 0.6/0.4 weighting: the fat-tree and WAN leaders

remain stable, while the mesh-grid point-estimate ordering changes without statistically significant separation from MADDPG.

3.8. Hop/Switch Penalty Sensitivity

The reward includes a realized gated hop/switch penalty term that discourages unnecessary detours and path changes. To test whether the results depend on its calibration, the same validated-profile trajectories were re-scored ex post while keeping $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$ fixed. The realized penalty was multiplied by $m \in \{0, 0.5, 1, 2\}$, where $m = 1$ is the deployed configuration. No policy was retrained, and the utilization and latency measurements remained unchanged. The $m = 1$ re-score reproduces the main validated-profile results exactly, so only the penalty magnitude changes in this analysis. Figure 11 shows the re-scored mean rewards for every tested multiplier.

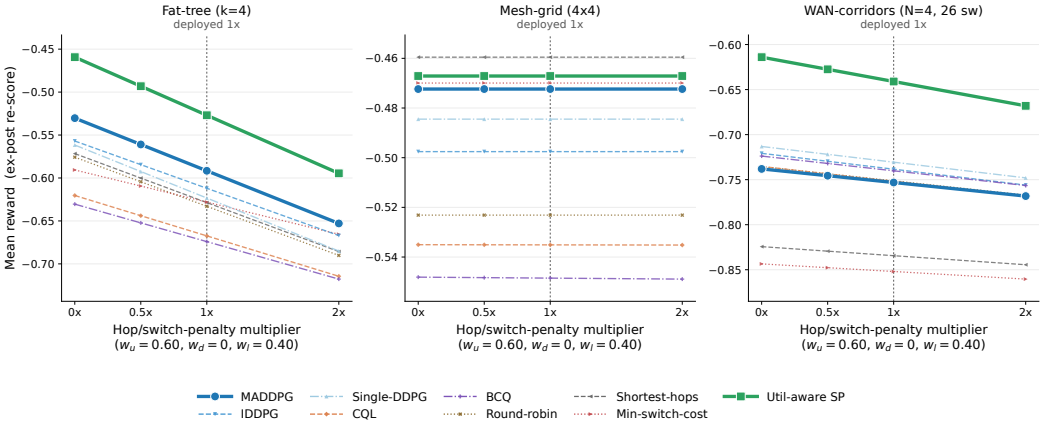


Figure 11. Hop/switch-penalty sensitivity under ex post re-scoring of the validated-profile trajectories. The reward weights remain fixed at $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$, while the realized penalty is scaled from $0\times$ to $2\times$. The deployed configuration is $1\times$. All nine policies are evaluated on the fat-tree, mesh-grid, and WAN-corridors topologies; higher reward is better.

Table 16 reports the corresponding point-estimate ranks. On fat-tree, Util-aware SP remains rank 1 and MADDPG remains rank 2 at every tested multiplier. Scaling the penalty changes the reward level and some middle-ranked positions but does not alter the leading order. In particular, Min-switch-cost moves from rank 7 at $0\times$ and $0.5\times$ to rank 5 at $1\times$ and rank 3 at $2\times$, consistent with its direct preference for low switch cost. IDDPG and Single-DDPG each fall by one rank at $2\times$. The significant fat-tree separation between Util-aware SP and MADDPG therefore persists across the tested penalty range.

Table 16. Hop/switch-penalty sensitivity ranks under ex post re-scoring. Rank 1 indicates the highest mean reward within a topology and penalty setting. The deployed $1\times$ configuration is shown in bold.

Topology	Penalty	MADDPG	IDDPG	Single	CQL	BCQ	RR	SH	MSC	UASP	Best Policy
Fat-tree	0×	2	3	4	8	9	6	5	7	1	Util-aware SP
Fat-tree	0.5×	2	3	4	8	9	6	5	7	1	Util-aware SP
Fat-tree	**1×**	**2**	**3**	**4**	**8**	**9**	**7**	**6**	**5**	**1**	**Util-aware SP**
Fat-tree	2×	2	4	5	8	9	7	6	3	1	Util-aware SP
Mesh-grid	0×	4	6	5	8	9	7	1	3	2	Shortest-hops
Mesh-grid	0.5×	4	6	5	8	9	7	1	3	2	Shortest-hops
Mesh-grid	**1×**	**4**	**6**	**5**	**8**	**9**	**7**	**1**	**3**	**2**	**Shortest-hops**
Mesh-grid	2×	4	6	5	8	9	7	1	3	2	Shortest-hops
WAN-corridors	0×	7	3	2	5	4	6	8	9	1	Util-aware SP
WAN-corridors	0.5×	7	3	2	6	4	5	8	9	1	Util-aware SP
WAN-corridors	**1×**	**7**	**3**	**2**	**6**	**4**	**5**	**8**	**9**	**1**	**Util-aware SP**
WAN-corridors	2×	6	3	2	7	4	5	8	9	1	Util-aware SP

RR: Round-robin; SH: Shortest-hops; MSC: Min-switch-cost; UASP: Util-aware SP.

The mesh-grid ordering is unchanged across the complete multiplier range. Shortest-hops remains rank 1, Util-aware SP rank 2, Min-switch-cost rank 3, and MADDPG rank 4. The mean rewards vary by at most approximately 0.001 between 0× and 2×, indicating that the realized gated penalty is close to zero for all mesh-grid policies. Moreover, the leading Shortest-hops, Util-aware SP, Min-switch-cost, and MADDPG group remains statistically unresolved. The stable point-estimate order therefore reflects the small realized penalty rather than sensitivity to its calibration.

On WAN-corridors, Util-aware SP remains rank 1, Single-DDPG rank 2, and IDDPG rank 3 at every multiplier. MADDPG remains rank 7 through the deployed 1× setting and moves to rank 6 at 2×. This change is only a swap within the closely spaced Round-robin, CQL, and MADDPG group and does not represent a statistically established improvement. The WAN reward ordering also remains subject to the flow-completion qualification in Section 3.3.

Overall, the leading conclusions are invariant over the tested 0×–2× range. Util-aware SP remains first on fat-tree and WAN-corridors, Shortest-hops remains first on mesh-grid, and MADDPG does not become rank 1 under any multiplier. The hop/switch penalty affects reward levels and selected middle-ranked positions, particularly on fat-tree, but it does not determine the leading policy ordering.

3.9. Policy-Behavior Mechanism Analysis

The preceding comparisons establish the relative performance of the evaluated policies but do not explain the routing behavior that produces these outcomes. We therefore analyze mean hop count, the longer-than-minimum-hop selection fraction, bottleneck utilization, RTT, path-switch frequency, normalized path-selection entropy, and flow completion. Figure 12 summarizes the first four metrics, while Table 17 reports the complete behavioral profile. The results reveal three distinct mechanisms: utilization-driven path switching on fat-tree, convergence toward near-shortest-path behavior on mesh-grid, and a completion–utilization trade-off on WAN-corridors.

Table 17. Policy-behavior metrics under the validated-profile setting. Values are means over ten seeds. "Longer-path fraction" denotes the fraction of selections longer than the retained minimum-hop candidate, and completion is the percentage of dispatched flows that terminate successfully.

Topology	Policy	Hops	Longer-Path Fraction	Util.	RTT (ms)	Switch	Entropy	Completion (%)
Fat-tree	MADDPG	4.137	0.161	0.721	31.031	0.360	0.814	90.208
Fat-tree	IDDPG	4.301	0.252	0.741	33.134	0.464	0.918	90.031
Fat-tree	Single-DDPG	4.392	0.290	0.748	32.862	0.329	0.775	88.515
Fat-tree	CQL	4.402	0.304	0.791	38.365	0.504	0.962	90.879
Fat-tree	BCQ	4.361	0.288	0.793	39.134	0.521	0.967	92.391
Fat-tree	Round-robin	4.506	0.332	0.762	33.830	0.370	0.999	88.263
Fat-tree	Shortest-hops	4.000	0.000	0.744	33.670	0.000	0.000	78.000
Fat-tree	Min-switch-cost	4.000	0.000	0.772	34.101	0.000	0.000	91.832
Fat-tree	Util-aware SP	4.021	0.041	0.600	31.232	0.783	0.734	88.954
Mesh-grid	MADDPG	2.174	0.048	0.735	20.063	0.067	0.180	95.510
Mesh-grid	IDDPG	3.383	0.570	0.719	26.207	0.476	0.940	95.896
Mesh-grid	Single-DDPG	3.016	0.368	0.722	23.270	0.292	0.829	96.272
Mesh-grid	CQL	3.406	0.587	0.744	29.345	0.599	0.990	92.710
Mesh-grid	BCQ	3.370	0.567	0.755	30.555	0.503	0.984	92.467
Mesh-grid	Round-robin	3.442	0.593	0.750	27.239	0.370	0.999	95.463
Mesh-grid	Shortest-hops	2.000	0.000	0.722	19.166	0.000	0.000	79.848
Mesh-grid	Min-switch-cost	2.300	0.075	0.720	21.381	0.000	0.000	90.619
Mesh-grid	Util-aware SP	2.898	0.331	0.677	25.202	0.602	0.764	92.526
WAN-corridors	MADDPG	8.018	0.320	0.759	56.877	0.524	0.951	90.150
WAN-corridors	IDDPG	8.081	0.357	0.741	56.626	0.575	0.981	89.820
WAN-corridors	Single-DDPG	8.050	0.316	0.734	55.598	0.592	0.978	90.300
WAN-corridors	CQL	8.042	0.339	0.765	55.896	0.612	0.993	91.860
WAN-corridors	BCQ	8.022	0.325	0.749	55.852	0.596	0.992	90.850
WAN-corridors	Round-robin	8.044	0.336	0.776	54.246	0.370	0.999	89.540
WAN-corridors	Shortest-hops	7.934	0.000	0.817	70.727	0.000	0.000	72.500
WAN-corridors	Min-switch-cost	7.933	0.000	0.839	74.191	0.000	0.000	73.270
WAN-corridors	Util-aware SP	8.254	0.320	0.593	49.379	0.990	0.999	76.270

On fat-tree, MADDPG and Util-aware SP select paths of comparable mean length, 4.137 and 4.021 hops, respectively, and obtain statistically indistinguishable RTT values of 31.031 ms and 31.232 ms. Their flow-completion rates are also statistically indistinguishable, at 90.2% and 89.0%, under both paired tests. The main difference is bottleneck utilization: Util-aware SP maintains a mean utilization of 0.600, compared with 0.721 for MADDPG. It also changes paths more frequently, with a switch frequency of 0.783 compared with 0.360. Because Util-aware SP selects the candidate path with the lowest observed bottleneck utilization at each decision step, this behavior directly targets the reward component receiving the largest weight.

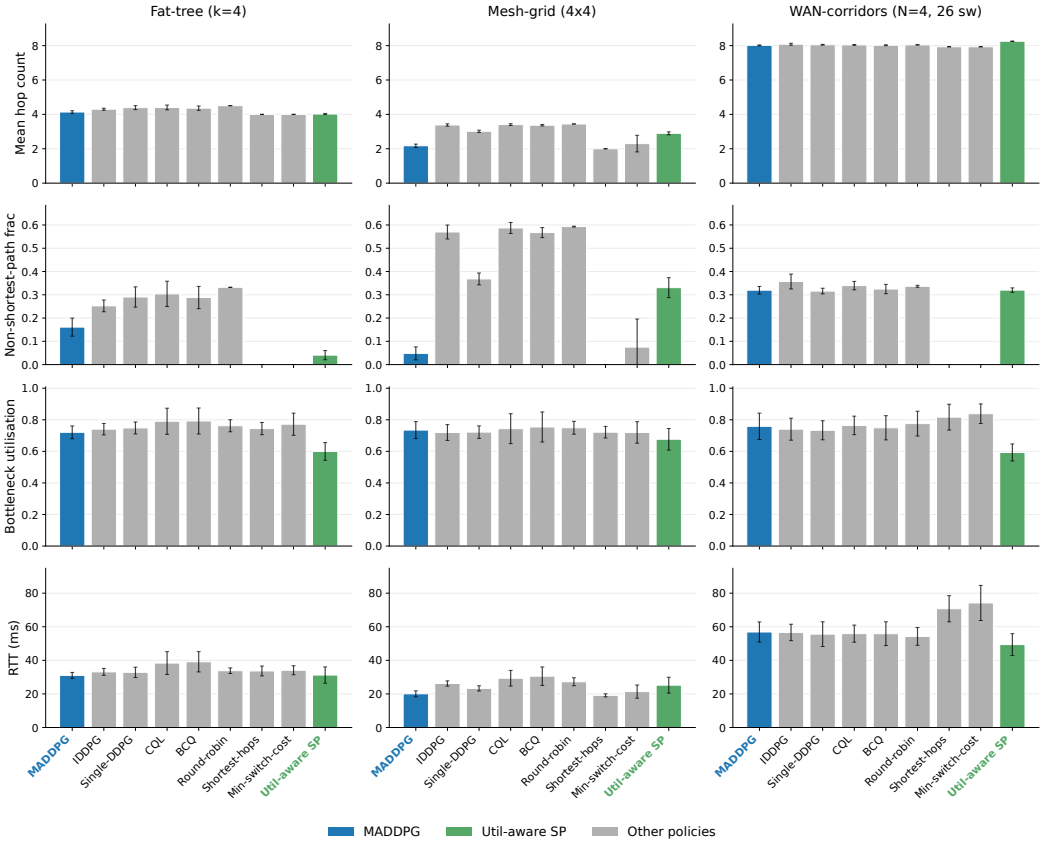


Figure 12. Policy-behavior metrics under the validated-profile setting on the fat-tree, mesh-grid, and WAN-corridors topologies. Bars show means over ten matched seeds and error bars show one standard deviation. The rows report mean hop count, longer-than-minimum-hop selection fraction, bottleneck utilization, and RTT. Lower values are preferable except where additional path diversity intentionally supports congestion avoidance.

The fat-tree reward difference between MADDPG and Util-aware SP is decomposed in Table 18. The total paired reward gap is

$$R_{\mathrm{MADDPG}} - R_{\mathrm{UASP}} = -0.0647.$$

The utilization term contributes -0.0726, corresponding to 112.4% of the net gap because the latency and penalty terms offset a small part of the utilization disadvantage. The utilization difference is significant ($p_t < 10^{-4}$), whereas the normalized latency-loss and hop/switch-penalty differences are not significant under the paired t-tests. The fat-tree result is therefore explained by lower instantaneous bottleneck utilization, not by lower latency, shorter paths, lower penalty, or reduced flow completion.

Table 18. Decomposition of the fat-tree reward gap between MADDPG and Util-aware SP. Component differences are computed as MADDPG minus Util-aware SP, and reward contributions follow $R = -(0.60U + 0.40L + P)$.

Component	MADDPG	Util-Aware SP	Difference	Reward Contribution	Paired t-Test p_t
Bottleneck utilization U	0.721	0.600	+0.121	−0.0726	0.00008
Latency loss L	0.245	0.249	−0.004	+0.0017	0.88638
Hop/switch penalty P	0.061	0.068	−0.006	+0.0063	0.11136
Total reward gap		$R_{\text{MADDPG}} - R_{\text{UASP}}$		−0.0647	—

The mesh-grid topology exhibits a different mechanism. MADDPG converges toward near-shortest-path behavior, selecting paths longer than the retained minimum-hop candidate in only 4.8% of decisions with a mean length of 2.174 hops. In contrast, IDDPG, Single-DDPG, CQL, and BCQ use paths longer than the retained minimum-hop candidate in 36.8–58.7% of decisions and incur higher RTT. Despite these substantially different routing choices, no evaluated policy differs significantly from MADDPG in bottleneck utilization or mean flow completion on mesh-grid. The additional detours therefore change path length and latency without establishing a measurable congestion advantage. This explains why the leading reward differences on mesh-grid remain statistically unresolved: the topology provides alternative paths, but those alternatives do not materially change the shared bottleneck conditions.

WAN-corridors must be interpreted jointly with flow completion. Util-aware SP reduces mean bottleneck utilization from 0.759 to 0.593 and RTT from 56.9 ms to 49.4 ms, but its completion rate falls from 90.2% to 76.3%. Shortest-hops and Min-switch-cost also complete substantially less traffic, at 72.5% and 73.3%, yet produce higher utilization and RTT than MADDPG. The three policies with statistically significant reward differences relative to MADDPG are therefore also the three policies with significantly lower completion. Among the completion-matched policies—IDDPG, Single-DDPG, CQL, BCQ, and Round-robin—none has a significant reward, utilization, or RTT advantage over MADDPG.

The behavioral evidence consequently does not support a single mechanism that applies across all topologies. On fat-tree, the utilization-aware heuristic gains an advantage by switching frequently and directly minimizing bottleneck utilization at comparable latency and completion. On mesh-grid, MADDPG approaches shortest-path behavior because additional detours do not provide measurable congestion relief. On WAN-corridors, scalar reward must be qualified by delivered traffic. These mechanisms explain the topology-dependent performance observed in the main comparison without attributing the results solely to the scalar reward ranking.

3.10. Paired Statistical Analysis

The preceding subsections compare mean rewards, ranks, and routing behavior. To test whether these differences are consistent across seeds, we performed paired seed-level comparisons between MADDPG and each comparison policy. For each topology and comparison policy, the paired reward gap is defined as

$$\Delta_i = \bar{R}_i^{\text{MADDPG}} - \bar{R}_i^{\text{comparison}},$$

where i indexes the matched evaluation seed. Positive values therefore indicate that MADDPG achieved a higher reward on the same seed. The expanded analysis covers eight comparison policies on the fat-tree, mesh-grid, and WAN-corridors topologies, using ten matched seeds and rewards computed with $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$. We report the mean paired gap, a bootstrap 95% confidence interval for the mean gap [40],

an exact paired permutation p-value based on all 2^{10} sign flips [41], an exact two-sided sign-test p-value, the number of seed-level wins, Cliff's dominance statistic [42], and the median paired gain. Figure 13 visualizes the paired mean gaps and bootstrap confidence intervals. The ten individual seed-level gaps for each comparison are available from the corresponding author with the supporting analysis outputs.

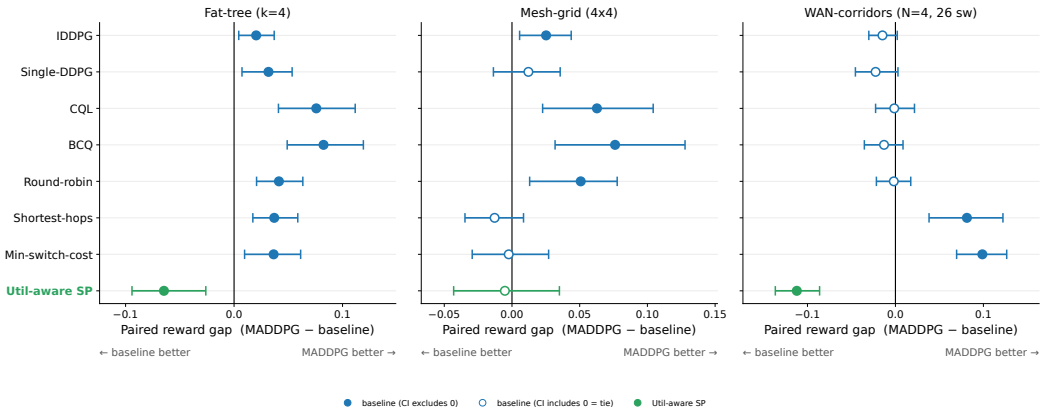


Figure 13. Paired reward gaps between MADDPG and each comparison policy under the validated-profile setting on the fat-tree, mesh-grid, and WAN-corridors topologies. Positive values indicate higher reward for MADDPG, while negative values indicate higher reward for the comparison policy. Error bars show bootstrap 95% confidence intervals for the mean paired gap across ten matched seeds. Filled markers indicate intervals that exclude zero; open markers indicate intervals that include zero.

Table 19 reports the numerical comparisons. On fat-tree, the bootstrap confidence interval is positive and excludes zero for IDDPG, Single-DDPG, CQL, BCQ, Round-robin, Shortest-hops, and Min-switch-cost. The exact permutation test is significant for six of these seven comparisons; the IDDPG comparison has a positive bootstrap interval but a permutation p-value of 0.0586. The largest mean gains are obtained against BCQ ($+0.0825$) and CQL ($+0.0758$). Util-aware SP is the only fat-tree comparison policy with a negative interval excluding zero: the mean gap is -0.0647, the bootstrap interval is $[-0.0941, -0.0261]$, and MADDPG wins on $2/10$ seeds. Thus, MADDPG outperforms the other learned policies on fat-tree but is significantly outperformed by the utilization-aware heuristic.

On mesh-grid, MADDPG has positive bootstrap intervals excluding zero against IDDPG, CQL, BCQ, and Round-robin. The intervals include zero against Single-DDPG, Shortest-hops, Min-switch-cost, and Util-aware SP. In particular, the three policies with higher point-estimate mean reward than MADDPG—Shortest-hops, Min-switch-cost, and Util-aware SP—have mean paired gaps of -0.0128, -0.0024, and -0.0052, respectively, with all confidence intervals crossing zero. No evaluated policy therefore significantly outperforms MADDPG on mesh-grid, despite differences in descriptive rank.

On WAN-corridors, MADDPG is statistically tied with IDDPG, Single-DDPG, CQL, BCQ, and Round-robin because all five bootstrap intervals include zero. It has positive intervals excluding zero against Shortest-hops and Min-switch-cost, with mean gaps of $+0.0813$ and $+0.0988$, respectively. Util-aware SP significantly outperforms MADDPG, with a mean gap of -0.1121, a bootstrap interval of $[-0.1365, -0.0862]$, and MADDPG wins on $0/10$ seeds.

The WAN separations require the completion qualification established in Section 3.3. Shortest-hops, Min-switch-cost, and Util-aware SP complete 17.6, 16.9, and 13.9 percentage points less traffic than MADDPG, respectively. Consequently, the two positive MADDPG gaps and the negative Util-aware SP gap are not comparisons at matched delivered load. The five comparison policies statistically tied with MADDPG form the completion-matched comparison group.

Table 19. Paired seed-level statistical comparisons between MADDPG and the eight comparison policies under the validated-profile setting. Positive mean gaps, median gains, and Cliff's δ values favor MADDPG.

Topology	Comparison Policy	Mean Gap	Bootstrap 95% Interval	Permutation p	Sign-Test p	Wins	Cliff's δ	Median Gain
Fat-tree	IDDPG	0.0204	[+0.0043, +0.0370]	0.0586	0.3438	7/10	0.52	0.0118
Fat-tree	Single-DDPG	0.0317	[+0.0073, +0.0535]	0.0410	0.3438	7/10	0.62	0.0461
Fat-tree	CQL	0.0758	[+0.0409, +0.1117]	0.0039	0.0215	9/10	0.78	0.0639
Fat-tree	BCQ	0.0825	[+0.0490, +0.1192]	0.0020	0.0020	10/10	0.94	0.0568
Fat-tree	Round-robin	0.0413	[+0.0207, +0.0634]	0.0059	0.0215	9/10	0.86	0.0396
Fat-tree	Shortest-hops	0.0370	[+0.0173, +0.0587]	0.0078	0.1094	8/10	0.74	0.0328
Fat-tree	Min-switch-cost	0.0365	[+0.0097, +0.0614]	0.0352	0.1094	8/10	0.52	0.0403
Fat-tree	Util-aware SP	−0.0647	[−0.0941, −0.0261]	0.0156	0.1094	2/10	−0.62	−0.0821
Mesh-grid	IDDPG	0.0252	[+0.0056, +0.0437]	0.0449	0.3438	7/10	0.46	0.0346
Mesh-grid	Single-DDPG	0.0121	[−0.0137, +0.0356]	0.3906	0.7539	6/10	0.44	0.0214
Mesh-grid	CQL	0.0627	[+0.0226, +0.1043]	0.0195	0.3438	7/10	0.42	0.0587
Mesh-grid	BCQ	0.0761	[+0.0318, +0.1278]	0.0098	0.0215	9/10	0.66	0.0563
Mesh-grid	Round-robin	0.0508	[+0.0130, +0.0777]	0.0234	0.0215	9/10	0.72	0.0585
Mesh-grid	Shortest-hops	−0.0128	[−0.0347, +0.0085]	0.3047	0.3438	3/10	−0.08	−0.0132
Mesh-grid	Min-switch-cost	−0.0024	[−0.0294, +0.0271]	0.8613	1.0000	5/10	0.02	−0.0026
Mesh-grid	Util-aware SP	−0.0052	[−0.0431, +0.0350]	0.7910	0.7539	4/10	−0.08	−0.0025
WAN-corridors	IDDPG	−0.0148	[−0.0303, +0.0021]	0.1250	0.3438	3/10	−0.24	−0.0194
WAN-corridors	Single-DDPG	−0.0226	[−0.0455, +0.0029]	0.1133	0.3438	3/10	−0.40	−0.0369
WAN-corridors	CQL	−0.0013	[−0.0226, +0.0217]	0.9121	1.0000	5/10	−0.04	0.0036
WAN-corridors	BCQ	−0.0130	[−0.0353, +0.0086]	0.3066	0.7539	4/10	−0.12	−0.0105
WAN-corridors	Round-robin	−0.0019	[−0.0216, +0.0174]	0.8652	0.7539	4/10	0.00	−0.0032
WAN-corridors	Shortest-hops	0.0813	[+0.0382, +0.1222]	0.0137	0.1094	8/10	0.62	0.0954
WAN-corridors	Min-switch-cost	0.0988	[+0.0696, +0.1265]	0.0020	0.0020	10/10	0.78	0.1043
WAN-corridors	Util-aware SP	−0.1121	[−0.1365, −0.0862]	0.0020	0.0020	0/10	−0.84	−0.1196

The expanded paired analysis does not support describing MADDPG as the most robust cross-topology controller. It significantly outperforms most comparison policies on fat-tree but loses to Util-aware SP, is significantly outperformed by no policy on mesh-grid, and remains tied with all completion-matched comparison policies on WAN-corridors. The supported conclusion is that MADDPG remains competitive across the evaluated topologies without being uniformly superior.

3.11. Auxiliary Traffic, Fault, and ECMP Tests

The main evaluation is complemented by three groups of auxiliary fat-tree tests addressing traffic-pattern variation, link-down and packet-loss conditions, and an ECMP forwarding reference. These experiments use three seeds and the reward configuration $w_u = 0.60$, $w_d = 0$, and $w_l = 0.40$. The traffic-pattern and fault-condition results are reported descriptively, whereas the ECMP comparison uses an exploratory paired test. These limited-sample results do not replace the ten-seed validated-profile comparison.

3.11.1. Traffic-Pattern Variation

Table 20 reports the three-seed results for the bursty and mixed traffic profiles defined in Section 2.9. MADDPG obtains mean rewards of −0.595 under bursty traffic and −0.602 under mixed traffic, both close to its three-seed validated-profile reference of −0.605. Util-aware SP obtains the highest point-estimate reward in both conditions. Under bursty traffic, Shortest-hops also exceeds MADDPG by point estimate, whereas under mixed traffic MADDPG ranks second among the five evaluated policies. These values are descriptive means over three seeds, and no inferential tests are assigned to these comparisons.

The proximity of the MADDPG results across the validated, bursty, and mixed profiles indicates no marked point-estimate degradation under the two tested changes to UDP mice and elephant traffic generation. This descriptive result does not establish superiority under altered traffic mixtures, because Util-aware SP remains stronger by point estimate in both conditions.

Table 20. Fat-tree reward under bursty and mixed traffic patterns. Values are mean ± standard deviation over three seeds; higher reward is better.

Traffic Pattern	Policy	Reward	Seeds
Bursty	MADDPG	-0.595 ± 0.019	3
Bursty	IDDPG	-0.618 ± 0.026	3
Bursty	Round-robin	-0.638 ± 0.028	3
Bursty	Shortest-hops	-0.580 ± 0.024	3
Bursty	Util-aware SP	-0.493 ± 0.025	3
Mixed	MADDPG	-0.602 ± 0.045	3
Mixed	IDDPG	-0.616 ± 0.030	3
Mixed	Round-robin	-0.665 ± 0.029	3
Mixed	Shortest-hops	-0.673 ± 0.023	3
Mixed	Util-aware SP	-0.536 ± 0.049	3

3.11.2. Tested Link-Down and Packet-Loss Conditions

Table 21 reports the three-seed results for the link-down and packet-loss conditions defined in Section 2.9. MADDPG obtains a mean reward of -0.713 under link down and -0.672 under packet loss, compared with the three-seed validated-profile reference of -0.605. The corresponding decreases are 0.108 and 0.067, respectively, with the larger point-estimate degradation occurring under the tested link failure. These comparisons are descriptive and are not assigned inferential p-values.

Table 21. Fat-tree reward under the tested link-down and packet-loss conditions. Values are mean ± standard deviation over three seeds; higher reward is better. The results are descriptive.

Condition	Policy	Reward	Seeds
Link down	MADDPG	-0.713 ± 0.060	3
Link down	IDDPG	-0.708 ± 0.042	3
Link down	Round-robin	-0.734 ± 0.032	3
Link down	Shortest-hops	-0.733 ± 0.058	3
Link down	Util-aware SP	-0.560 ± 0.056	3
Packet loss	MADDPG	-0.672 ± 0.012	3
Packet loss	IDDPG	-0.700 ± 0.040	3
Packet loss	Round-robin	-0.669 ± 0.012	3
Packet loss	Shortest-hops	-0.758 ± 0.035	3
Packet loss	Util-aware SP	-0.577 ± 0.058	3

Util-aware SP has the highest point-estimate reward under both tested conditions. MADDPG remains above Round-robin and Shortest-hops in the link-down experiment and above IDDPG and Shortest-hops in the packet-loss experiment. These three-seed descriptive results do not establish general fault tolerance or superiority. They cover one persistent link failure and one fabric-wide packet-loss condition; switch failure and link jitter are not evaluated.

3.11.3. Single-Path and ECMP Forwarding

The static per-flow ECMP reference defined in Section 2.9 is compared with the deployed single-path MADDPG controller on fat-tree. Table 22 shows that ECMP obtains a mean reward of -0.688, compared with -0.605 for single-path MADDPG. The paired difference is -0.083, with a nominal two-sided paired t-test value of $p_t = 0.0116$.

Table 22. Single-path MADDPG and static per-flow ECMP forwarding on fat-tree. The difference is computed as ECMP minus single-path MADDPG; higher reward is better. The three-seed p_t value is from a two-sided paired t-test and is interpreted as exploratory.

Forwarding Mode	Mean Reward	Difference	Paired t-Test p_t	Seeds
MADDPG single-path	−0.605	Reference	—	3
ECMP per-flow	−0.688	−0.083	0.0116	3

The ECMP comparison therefore does not show an advantage for the tested static per-flow allocation under the evaluated fat-tree condition and scalar objective. This exploratory three-seed result does not establish that single-path forwarding is generally superior to multipath scheduling. It shows only that this destination-port-based ECMP reference does not improve upon the deployed MADDPG policy in the reported experiment. The comparison does not evaluate learned inter-agent coordination, traffic-class-aware splitting, or flowlet scheduling.

3.12. Controller Overhead and Scalability

3.12.1. Isolated Policy Inference

The computational cost of the learned decision module was first measured independently of the surrounding controller operations. Each exported, torch-free NumPy policy was evaluated over 1500 logged routing decisions after a 50-call warm-up. One joint routing decision selects one candidate path for every controlled traffic pair. This benchmark isolates policy evaluation and therefore excludes state polling, Ryu event processing, forwarding-rule installation, and barrier synchronization. Figure 14 reports the median and 95th-percentile inference times for MADDPG and its two architectural variants on all three topologies.

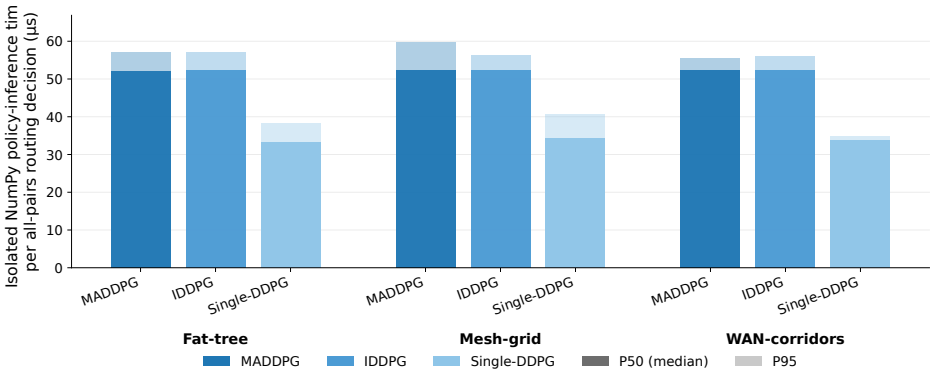


Figure 14. Isolated inference time of the exported NumPy policies on the fat-tree, mesh-grid, and WAN-corridors topologies. One measurement covers path selection for all controlled traffic pairs. Bars show the median and 95th-percentile times over 1500 logged decisions.

Table 23 reports the corresponding model footprints and timing statistics. MADDPG and IDDPG each contain 79,372 parameters and occupy 318.8 KB, while Single-DDPG contains 91,660 parameters and occupies 367.3 KB. MADDPG has median inference times of 52.0 µs, 52.4 µs, and 52.3 µs on fat-tree, mesh-grid, and WAN-corridors, respectively. Its 95th-percentile times remain between 55.4 µs and 59.7 µs.

The isolated timing is consistent across the three topologies because the deployed actor layout and the number of controlled traffic pairs are unchanged. Single-DDPG is faster in this benchmark because it performs one joint forward pass rather than one actor evaluation per traffic pair. All three learned controllers nevertheless complete an all-pairs decision in

less than 60 µs at the 95th percentile. These values characterize only the exported decision module and must not be interpreted as full SDN control-loop latency.

Table 23. Isolated policy-inference overhead and exported model footprint. Timing was measured over 1500 logged all-pairs routing decisions.

Topology	Policy	Parameters	Size (KB)	Mean (µs)	P50 (µs)	P95 (µs)
Fat-tree	MADDPG	79,372	318.8	52.6	52.0	57.0
Fat-tree	IDDPG	79,372	318.8	52.9	52.2	57.1
Fat-tree	Single-DDPG	91,660	367.3	34.1	33.2	38.3
Mesh-grid	MADDPG	79,372	318.8	53.6	52.4	59.7
Mesh-grid	IDDPG	79,372	318.8	52.8	52.2	56.4
Mesh-grid	Single-DDPG	91,660	367.3	35.3	34.2	40.6
WAN-corridors	MADDPG	79,372	318.8	52.8	52.3	55.4
WAN-corridors	IDDPG	79,372	318.8	53.0	52.4	55.9
WAN-corridors	Single-DDPG	91,660	367.3	33.8	33.7	34.8

3.12.2. Full Controller-Loop Timing

To address the distinction between isolated inference and deployed controller operation, the live Ryu–Mininet loop was instrumented for one MADDPG run on each topology. The recorded stages include network-statistics polling, state construction, policy decision, forwarding-rule installation, barrier synchronization, and the complete elapsed control interval. Table 24 reports the mean stage times from the instrumented runs.

Table 24. Mean full-loop timing breakdown for the instrumented MADDPG run on each topology. All values are in milliseconds. The complete elapsed interval includes operations outside the individually instrumented stage timers.

Topology	Polling	State	Decision	Installation	Barrier	Loop Elapsed
Fat-tree	486.784	0.146	2.078	3.886	7.165	575.791
Mesh-grid	512.570	0.166	0.386	3.825	7.185	617.239
WAN-corridors	509.489	0.172	0.809	7.467	8.613	562.030

The complete measured control interval ranges from 562.030 ms on WAN-corridors to 617.239 ms on mesh-grid. Network-statistics polling is the dominant explicitly timed stage, requiring approximately 487–513 ms. State construction remains below 0.2 ms, the controller-side decision stage remains below 2.1 ms, forwarding-rule installation requires 3.8–7.5 ms, and barrier synchronization requires 7.2–8.6 ms. The full-loop measurements therefore confirm that the approximately 52 µs exported-policy inference cost is not the dominant contributor to deployed controller latency. The limiting cost in the present implementation is monitoring and the surrounding controller workflow.

3.12.3. Decision Scalability

A separate offline microbenchmark evaluates how joint policy-decision time scales with the number of controlled traffic pairs N and the number of candidate paths K. The state and local-observation dimensions grow as

$$d_s = 7 + 5KN, \qquad d_o = 7 + 5K.$$

The benchmark evaluates $N \in \{2, 4, 8, 16, 32, 64\}$ and $K \in \{3, 5, 10\}$. Figure 15 shows the resulting joint decision latency.

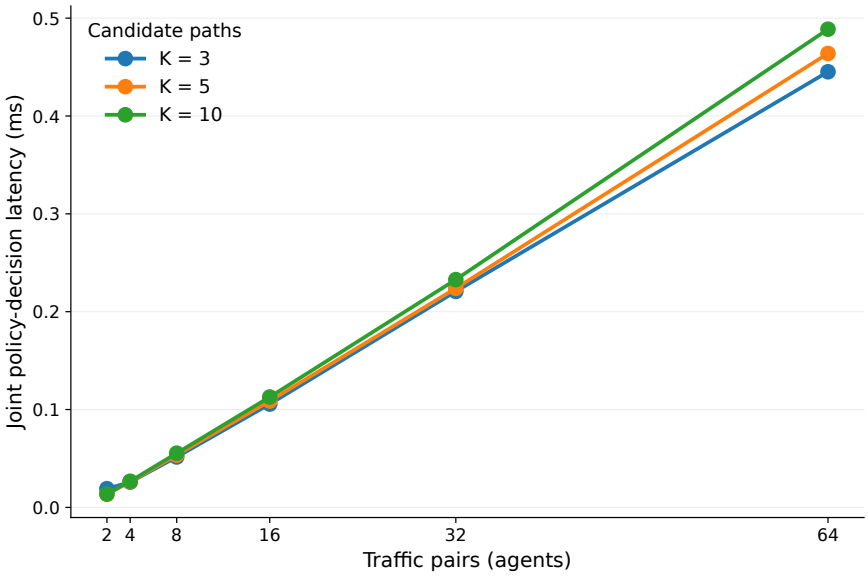


Figure 15. Offline joint policy-decision latency as the number of traffic-pair agents N and candidate paths K increases. Each measurement includes action selection for all N agents.

Table 25. Joint policy-decision latency in the scalability microbenchmark. Values are in milliseconds and include action selection for all traffic-pair agents.

Traffic Pairs N	$K = 3$	$K = 5$	$K = 10$
2	0.0191	0.0134	0.0137
4	0.0258	0.0262	0.0267
8	0.0515	0.0535	0.0554
16	0.1056	0.1087	0.1128
32	0.2207	0.2239	0.2328
64	0.4452	0.4639	0.4887

Table 25 reports the corresponding measured latencies. Joint decision latency grows approximately linearly with the number of traffic pairs. At $N = 64$, the measured latency is 0.4452 ms for $K = 3$, 0.4639 ms for $K = 5$, and 0.4887 ms for $K = 10$. Increasing the candidate-path count has a smaller effect than increasing the number of agents: the per-agent observation dimension rises from 22 at $K = 3$ to 57 at $K = 10$, while the corresponding actor parameter count rises from 19,843 to 25,226. The complete joint decision remains below 0.5 ms throughout the tested range. This synthetic scaling result concerns policy computation only and does not include topology discovery, candidate-path generation, monitoring, or forwarding-rule installation.

3.13. Operating Envelope

The preceding analyses use the validated offered-traffic profile for the main policy comparison. To examine the operating envelope of the proposed controller without mixing policy sets or unequal seed counts, this analysis reports MADDPG alone across three topologies and three nominal offered-traffic profiles. The moderate-low, validated, and heavy labels describe the configured traffic profiles rather than topology-normalized realized load levels. All nine topology–profile combinations use the same three seeds, allowing the measurements to be compared on a uniform 3 topologies $\times$ 3 profiles $\times$ 3 seeds $\times$ 1 policy design.

Figure 16 reports reward, realized bottleneck utilization, and end-to-end RTT. The figure does not rank MADDPG against other policies; the validated-profile policy comparison is reported separately in Section 3.1.

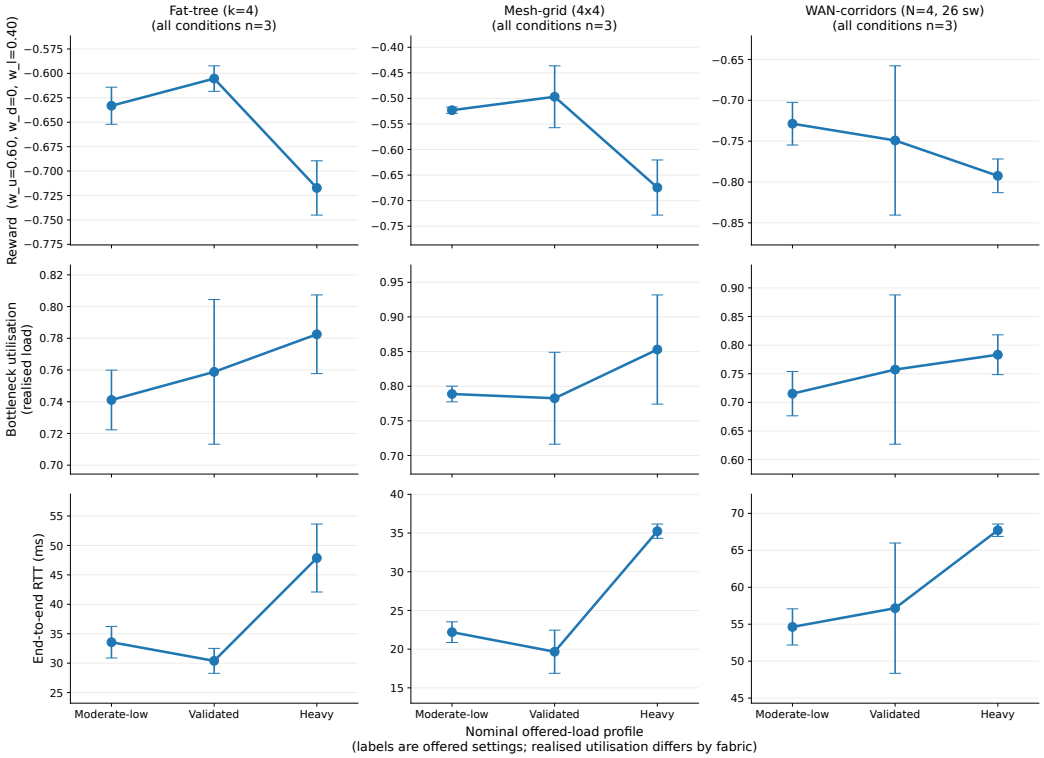


Figure 16. Operating envelope of MADDPG across three nominal offered-traffic profiles on the fat-tree, mesh-grid, and WAN-corridors topologies. The rows report reward, realized bottleneck utilization, and end-to-end RTT. Values are means over the same three seeds in every condition, and error bars show one standard deviation. The profile labels denote configured offered-traffic settings; realized utilization differs by topology.

Table 26 reports the corresponding measurements. Under the heavy offered-traffic profile, MADDPG obtains its lowest mean reward on each topology: -0.717 on fat-tree, -0.674 on mesh-grid, and -0.792 on WAN-corridors. The heavy profile also produces the highest mean realized utilization and RTT within each topology. Fat-tree RTT rises to 47.9 ms, mesh-grid RTT to 35.2 ms, and WAN-corridors RTT to 67.7 ms.

Table 26. Operating envelope of MADDPG across three topologies and three nominal offered-traffic profiles. Values are mean ± standard deviation over the same three seeds. The drop column reports the mean logged per-step drop fraction.

Topology	Offered-Traffic Profile	Reward	Bottleneck Utilization	RTT (ms)	Drop	Seeds
Fat-tree	Moderate-low	-0.633 ± 0.019	0.741 ± 0.019	33.6 ± 2.7	0.014	3
Fat-tree	Validated	-0.605 ± 0.013	0.759 ± 0.046	30.4 ± 2.1	0.009	3
Fat-tree	Heavy	-0.717 ± 0.028	0.783 ± 0.025	47.9 ± 5.8	0.023	3
Mesh-grid	Moderate-low	-0.523 ± 0.006	0.789 ± 0.011	22.2 ± 1.3	0.005	3
Mesh-grid	Validated	-0.497 ± 0.061	0.783 ± 0.066	19.7 ± 2.8	0.008	3
Mesh-grid	Heavy	-0.674 ± 0.054	0.853 ± 0.079	35.2 ± 0.9	0.019	3
WAN-corridors	Moderate-low	-0.729 ± 0.026	0.715 ± 0.039	54.6 ± 2.4	0.008	3
WAN-corridors	Validated	-0.749 ± 0.091	0.757 ± 0.130	57.2 ± 8.8	0.012	3
WAN-corridors	Heavy	-0.792 ± 0.021	0.783 ± 0.035	67.7 ± 0.8	0.014	3

The moderate-low and validated profiles do not produce the same realized-load ordering on every topology. On fat-tree, realized utilization increases from 0.741 under the moderate-low profile to 0.759 under the validated profile. On WAN-corridors, it similarly increases from 0.715 to 0.757. On mesh-grid, however, the moderate-low profile produces a slightly higher mean utilization than the validated profile, 0.789 compared with 0.783.

The profile labels must therefore be interpreted as offered-traffic configurations rather than as directly comparable realized congestion levels across topologies.

The consistent result is the degradation under the heavy profile. Reward decreases, realized utilization reaches its maximum, and RTT increases on all three topologies. Nevertheless, mean bottleneck utilization remains below 0.86 in every condition, and the controller continues to produce executable routing decisions. The operating-envelope analysis therefore identifies increased traffic pressure as a clear performance limitation, while avoiding claims about comparative policy rank outside the validated-profile experiments.

4. Discussion

The results support a bounded interpretation of offline multi-agent reinforcement learning for SDN path control. MADDPG remains competitive across the evaluated settings, but its relative performance depends on topology structure, reward scalarization, coverage of the logged behavior data, and whether policies are compared at matched flow completion. It is the strongest learned policy on fat-tree, no evaluated policy significantly outperforms it on mesh-grid, and it remains tied with the completion-matched policy group on WAN-corridors, while Util-aware SP achieves the strongest overall reward ranking. The following discussion interprets these findings in relation to topology-dependent routing mechanisms, behavior adjustment, objective sensitivity, controller overhead, and the requirements of 6G-oriented network softwarization.

4.1. Interpretation of the Fat-Tree Result

The strongest relative result for MADDPG is observed on the fat-tree topology, where it is the highest-ranked learned policy but is outperformed by Util-aware SP. This is consistent with the working hypothesis that learned path control is most useful when the topology provides sufficient path diversity for congestion-aware routing decisions. Fat-tree data-center designs were introduced to provide scalable bandwidth using commodity switches and multiple alternative paths between communicating hosts [18]. In such multi-stage fabrics, static shortest-path or switch-cost heuristics may fail to use the available path diversity effectively when traffic concentrates on particular links. Prior work on data-center flow scheduling has similarly shown that dynamic path selection can improve utilization of multi-stage switching fabrics when traffic placement matters [20]. The revised results extend this interpretation to offline multi-agent reinforcement learning while also showing that a utilization-aware heuristic can exploit the same path diversity more directly under the deployed objective.

The validated-profile comparison places Util-aware SP first on fat-tree with a mean reward of -0.527, followed by MADDPG at -0.592. MADDPG nevertheless remains the best-performing learned policy and outperforms IDDPG, Single-DDPG, CQL, and BCQ. The difference between MADDPG and Util-aware SP is not explained by flow completion, path length, or latency. Their completion rates are statistically indistinguishable, at 90.2% and 89.0%, respectively, under both paired tests; their mean path lengths are 4.137 and 4.021 hops; and their RTT values, 31.031 ms and 31.232 ms, are also statistically indistinguishable. The principal difference is bottleneck utilization: Util-aware SP maintains a mean utilization of 0.600, compared with 0.721 for MADDPG, while switching paths more frequently (0.783 versus 0.360).

The reward-gap decomposition confirms this mechanism. The utilization term contributes approximately 112% of the observed reward gap of -0.0647, because the latency and hop/switch-penalty terms offset a small part of the utilization disadvantage. The utilization contribution is statistically significant, whereas the latency and penalty contributions are not. Since the deployed reward assigns its largest weight to bottleneck utilization,

$w_u = 0.60$, Util-aware SP has a structural per-step advantage: it selects the retained candidate with the lowest currently observed bottleneck utilization. MADDPG, in contrast, must recover this behavior from transitions generated by the logged random behavior policy without online exploration.

The sensitivity analyses show that this ordering is not caused by one narrow reward calibration. Util-aware SP remains rank 1 and MADDPG remains rank 2 on fat-tree across all tested utilization–latency weightings and all tested hop/switch-penalty multipliers. The reward gap between them is not statistically significant in the most latency-dominant settings, but it is significant at the deployed $w_u/w_l = 0.6/0.4$ configuration and when utilization receives greater weight. The paired seed-level analysis gives the same bounded result: MADDPG has bootstrap confidence intervals favoring it against the other seven comparison policies, whereas Util-aware SP is the only policy with an interval that significantly favors the comparison policy. The stable conclusion is therefore that MADDPG is the strongest learned fat-tree controller, while the utilization-aware heuristic is strongest under the evaluated scalar objective.

The implemented architectural variants obtain lower point-estimate rewards than MADDPG on fat-tree: IDDPG differs by -0.0204, and Single-DDPG differs by -0.0317. Because target-projection settings are unmatched on this topology, these differences cannot be attributed exclusively to centralized joint-action evaluation or the per-pair multi-agent decomposition. The behavior-adjustment result is less conclusive on this topology: removing the behavior-deviation reward adjustment changes reward by -0.043, but the three-seed paired comparison is not statistically significant ($p_t = 0.133$). The tested learning-rate, discount-factor, and hidden-width perturbations also produce lower point-estimate rewards than the selected configuration, although these sensitivity experiments contain only three seeds. The fat-tree evidence therefore supports the usefulness of the MADDPG architecture as an offline learned controller, but it does not justify attributing its performance to uniformly beneficial behavior adjustment or claiming superiority over a heuristic that directly minimizes the dominant per-step reward component.

4.2. Mesh-Grid as a Structural Boundary

The mesh-grid result should be interpreted as a structural boundary rather than as a failure of the learned controller. Unlike the fat-tree topology, where multiple alternative paths can be exploited to avoid congested links, the mesh-grid topology evaluated in this study has limited useful path diversity and short feasible paths. In this setting, the shortest-hop heuristic is already a strong topology-specific policy. The validated-profile point estimates reflect this structure: Shortest-hops obtains the highest mean reward, -0.460, followed by Util-aware SP at -0.467, Min-switch-cost at -0.470, and MADDPG at -0.472. However, the paired comparisons do not significantly separate any of these three policies from MADDPG. The RTT measurements show a similar descriptive ordering, with 19.166 ms for Shortest-hops and 20.063 ms for MADDPG. The mesh-grid result should therefore be interpreted as a statistically unresolved leading group rather than as a significant degradation of the learned controller.

The policy-behavior metrics explain this boundary. On mesh-grid, MADDPG selects paths with an average length of 2.174 hops and selects a candidate longer than the retained minimum-hop candidate in only 4.8% of decisions. This behavior is much closer to Shortest-hops than to the other learned controllers. IDDPG and Single-DDPG select a candidate longer than the retained minimum-hop candidate in 57.0% and 36.8% of decisions and obtain mean RTT values of 26.207 ms and 23.270 ms, respectively; CQL and BCQ also select a candidate longer than the retained minimum-hop candidate in more than half of their decisions. Despite these substantial behavioral differences, none of the evaluated policies

differs significantly from MADDPG in bottleneck utilization or mean flow completion on mesh-grid. The additional detours therefore increase path length and latency without establishing measurable congestion relief. MADDPG instead suppresses most unnecessary detours and converges toward near-shortest-path behavior.

The sensitivity analyses support the same bounded interpretation. Across the reward-weight sweep, MADDPG ranks second at $w_u \in \{0.2, 0.4, 0.5\}$, fourth at the deployed $w_u = 0.6$, and fifth at $w_u = 0.8$. Shortest-hops has the highest point-estimate reward for $w_u \leq 0.6$, while Util-aware SP leads at $w_u = 0.8$. Nevertheless, MADDPG is not significantly separated from the point-estimate winner at any tested weight. The hop/switch-penalty sweep is also stable: Shortest-hops, Util-aware SP, Min-switch-cost, and MADDPG retain ranks one through four, respectively, for every multiplier from $0\times$ to $2\times$. The realized penalty is close to zero for all mesh-grid policies, and this leading group remains statistically unresolved. The mesh-grid conclusion is therefore not determined by a particular reward weighting or penalty calibration.

The paired statistical analysis further supports this conclusion. MADDPG significantly outperforms IDDPG, CQL, BCQ, and Round-robin, with mean paired reward gaps of $+0.0252$, $+0.0627$, $+0.0761$, and $+0.0508$, respectively. It is statistically tied with Single-DDPG, Shortest-hops, Min-switch-cost, and Util-aware SP. In particular, the mean gaps against Shortest-hops, Min-switch-cost, and Util-aware SP are -0.0128, -0.0024, and -0.0052, respectively, and all corresponding bootstrap confidence intervals include zero. No evaluated policy therefore significantly outperforms MADDPG on mesh-grid.

This result is useful for deployment because it identifies when learned path control should and should not be expected to provide a gain. In topologies with limited useful path diversity, a simple shortest-hop heuristic may already provide a strong reference policy for latency-sensitive routing. The relevant result is not that MADDPG significantly outperforms Shortest-hops, but that it avoids the extensive detouring observed in several weaker learned policies while remaining statistically tied with the leading topology-specific methods. Combined with the fat-tree result, the mesh-grid analysis identifies a structural limitation of adaptive path selection: offline multi-agent control provides clearer benefits when alternative paths can materially change bottleneck utilization, whereas its advantage is reduced when different routing choices leave congestion conditions largely unchanged.

4.3. WAN-Corridors Interpretation

The WAN-corridors topology produces a different evaluation issue from the fat-tree and mesh-grid cases because reward differences coincide with substantial differences in completed traffic. Util-aware SP obtains the highest mean reward, but its flow-completion rate is 76.3%, compared with 90.2% for MADDPG. Shortest-hops and Min-switch-cost similarly complete only 72.5% and 73.3% of dispatched flows. Because the deployed objective uses $w_d = 0$, these reward comparisons are not comparisons at matched delivered load and must be interpreted jointly with completion.

The paired analysis separates the WAN policies into two groups. Util-aware SP significantly outperforms MADDPG in scalar reward, whereas MADDPG significantly outperforms Shortest-hops and Min-switch-cost. However, all three policies also have significantly lower completion than MADDPG. In contrast, IDDPG, Single-DDPG, CQL, BCQ, and Round-robin have completion rates comparable to MADDPG, and none shows a statistically significant reward advantage over it. The appropriate WAN conclusion is therefore that MADDPG belongs to a completion-matched statistical tie group rather than that its descriptive reward rank alone represents a clear performance ordering.

The behavioral measurements further distinguish traffic shedding from effective routing. Util-aware SP operates at lower bottleneck utilization and lower RTT than MADDPG,

but it also carries 13.9 percentage points less traffic. Shortest-hops and Min-switch-cost shed comparable amounts of traffic yet obtain higher utilization and substantially higher RTT, showing that reduced completion alone does not guarantee a favorable reward. Nevertheless, the WAN results demonstrate that scalar congestion and latency objectives can favor policies operating under different realized traffic volumes unless completion is reported as an accompanying outcome.

WAN-corridors also provides the clearest evidence that behavior adjustment is topology-dependent. Removing the behavior-deviation reward adjustment increases MADDPG reward from -0.753 to -0.681, increases completion from 90.2% to 94.3%, reduces bottleneck utilization from 0.759 to 0.676, and reduces RTT from 56.9 ms to 49.6 ms. Because completion improves rather than decreases, this within-method gain is not explained by traffic shedding. The corresponding behavior-adjustment effects on fat-tree and mesh-grid are not statistically significant, so the evidence does not support treating a fixed positive β as uniformly beneficial across network structures.

The WAN-corridors result therefore narrows the deployment claim in two ways. First, comparisons based on the present scalar reward require a matched-completion qualification. Second, the behavior-adjustment weight must reflect how well the logged behavior distribution supports useful routing decisions in the target topology. Offline MADDPG remains competitive among policies carrying comparable traffic, but neither the selected behavior-adjustment weight nor its descriptive reward position should be generalized across topologies without this qualification.

4.4. Implications for 6G-Oriented Network Softwarization

The results are relevant to 6G-oriented network softwarization because they address three requirements that are repeatedly associated with future programmable networks: adaptive control, low-latency operation, and AI-assisted management. 6G visions emphasize that future networks will need to support heterogeneous latency-sensitive services, integrate communication with computation, and rely increasingly on intelligent control mechanisms [1]. Edge-intelligence studies make a related point from the computing side: artificial-intelligence and machine-learning functions are expected to move closer to the edge, where network management and real-time decision-making must operate under tighter delay and resource constraints [4]. In this context, SDN path control is a useful abstraction because it exposes the forwarding decision to software while preserving a clear deployment path through the controller.

Although the present study focuses on the softwarized network-control layer, the reliability of higher-layer routing decisions ultimately depends on the quality of measurements obtained from the underlying communication system. This coupling is particularly relevant to near-field extremely large-scale multiple-input multiple-output (XL-MIMO) systems, where channel estimation under low signal-to-noise ratio (SNR) conditions remains challenging. Recent enhanced polar-domain channel-estimation methods illustrate ongoing efforts to provide more reliable physical-layer information in such environments [8]. Integrating this information with SDN path control is outside the scope of the present experiments, but it represents a relevant direction for cross-layer AI-assisted 6G control.

The proposed controller contributes to this softwarization setting in a specific way. It does not require online reinforcement-learning exploration in the deployed network; instead, it is trained from logged Ryu–Mininet transitions and exported as a deterministic argmax path-selection policy. This separation avoids exploratory routing actions during deployment, where unsupported decisions could directly affect latency, utilization, or service continuity. However, the behavior-adjustment results show that proximity to the logged behavior is not uniformly beneficial. Removing the behavior-deviation reward adjustment

has no significant effect on mesh-grid, produces a non-significant reduction on fat-tree, and significantly improves reward, completion, utilization, and RTT on WAN-corridors. Behavior adjustment should therefore be treated as a topology- and dataset-dependent design choice rather than as an inherently beneficial property of offline deployment.

The literature on software-defined networking, network function virtualization (NFV), and network slicing has long treated programmability as a key enabler for flexible service-specific networking [33,34]. The present study does not implement full network slicing or NFV orchestration, but it addresses a lower-level control primitive that such systems depend on: selecting latency-aware paths under changing load. The experimental results show that this primitive can support competitive offline multi-agent control, but its benefit depends on topology and on the available heuristic alternatives. MADDPG is the strongest learned policy on fat-tree, no evaluated policy significantly outperforms it on mesh-grid, and it remains tied with the completion-matched policy group on WAN-corridors. Util-aware SP nevertheless achieves the strongest overall reward ranking. The contribution is therefore not a universal replacement for topology-aware routing rules, but an executable offline-learning framework whose relative advantages and limitations can be measured under different network structures.

The controller-overhead measurements distinguish isolated policy inference from complete SDN control-loop operation. The exported MADDPG policy contains 79,372 parameters, occupies 318.8 KB, and has median all-pairs inference times of approximately 52 µs across the three evaluated topologies, with 95th-percentile times below 60 µs. The instrumented live control loop is substantially slower, with complete mean intervals of approximately 562–617 ms. Network-statistics polling alone requires approximately 487–513 ms, whereas state construction remains below 0.2 ms, policy decision below 2.1 ms, forwarding-rule installation between 3.8 and 7.5 ms, and barrier synchronization between 7.2 and 8.6 ms. The learned decision module is therefore not the dominant source of controller latency; monitoring and the surrounding controller workflow determine the practical control interval.

The broader implication is not that one MADDPG controller is sufficient for all 6G network-control problems. The mesh-grid and WAN-corridors results show that policy effectiveness depends on topology structure, logged-data coverage, and whether comparisons are made at matched delivered traffic. The operating-envelope analysis further shows that the heavy offered-traffic profile degrades reward and increases realized utilization and RTT on all three topologies, while the moderate-low and validated profiles do not produce a uniform realized-load ordering across network structures. This bounded behavior is valuable for 6G-oriented systems because future networks are expected to be heterogeneous rather than uniform. A practical AI-assisted controller should therefore expose its operating envelope, report reliability outcomes such as flow completion alongside scalar reward, and retain topology-aware heuristics when they provide a stronger or simpler control rule. The results of this study support that operating-regime-aware view of network softwarization.

4.5. Limitations and Future Work

This study has several limitations that define the scope of the reported conclusions. First, the evaluation uses three emulated topologies—fat-tree, mesh-grid, and WAN-corridors—within a Ryu–Mininet SDN environment. These topologies provide path-diverse, lower-diversity, and corridor-like routing structures, but they do not reproduce the hardware, protocol heterogeneity, traffic scale, or operational constraints of industrial data centers, wide-area networks, or 6G edge-slicing deployments. The results should therefore

be interpreted as evidence from controlled executable emulation rather than as universal performance claims for programmable network infrastructures.

Second, the main validated-profile comparison uses ten matched seeds for each of the three topologies, whereas most auxiliary traffic-pattern, fault, ECMP, and hyperparameter experiments use three seeds. The behavior-adjustment comparison uses ten seeds on WAN-corridors but only three seeds on fat-tree and mesh-grid. The full control-loop timing is based on one instrumented run per topology. These smaller auxiliary samples establish effect direction and implementation behavior, but they provide less statistical resolution than the main paired comparison and are interpreted as exploratory where inferential tests are reported.

Third, the reward-weight and hop/switch-penalty sensitivity analyses are ex post re-scoring experiments. They evaluate identical deployed trajectories under alternative scalarizations and therefore test whether the interpretation of the recorded behavior depends on the objective specification. They do not show how policies would change if retrained directly under each reward configuration. Future work should retrain the controllers under latency-dominant, utilization-dominant, reliability-aware, and service-class-specific objectives and compare the resulting deployed behavior.

Fourth, the offline dataset is generated using a random behavior policy. Although this procedure provides broad candidate-action coverage, the learned controller remains constrained by the states and actions represented in the logged transitions. The topology-dependent behavior-adjustment result illustrates this limitation: the selected $\beta = 0.1$ configuration is not significantly separated from $\beta = 0$ on fat-tree or mesh-grid, while removing the behavior-deviation reward adjustment significantly improves reward, completion, utilization, and RTT on WAN-corridors. Future work should investigate dataset-coverage diagnostics and online-safe fine-tuning that permits bounded adaptation without unrestricted exploration in the operational network.

Fifth, the deployed reward uses $w_d = 0$, and the logged drop-related term does not measure end-to-end flow completion. Scalar reward comparisons can therefore be misleading when policies carry different amounts of traffic, as observed on WAN-corridors. Flow completion must be reported together with reward, utilization, and RTT whenever delivered load is not matched. In addition, the moderate-low, validated, and heavy operating-envelope labels denote configured offered-traffic profiles rather than topology-normalized realized load levels. The standardized envelope shows that the heavy profile degrades reward and increases utilization and RTT on all three topologies, but the moderate-low and validated profiles do not produce a uniform realized-load ordering across fabrics.

Sixth, several comparisons remain incomplete. CQL, BCQ, and Util-aware SP broaden the baseline set, but the evaluation does not include a mathematical optimization solver or a more advanced SDN-specific multi-agent reinforcement-learning method. The hyperparameter analysis varies hidden-layer width but not network depth. The architectural comparisons are also qualified by a target-stabilization asymmetry: IDDPG and Single-DDPG use empirical target projection on all three topologies, whereas MADDPG uses it only on WAN-corridors. Consequently, these variants should not be interpreted as perfectly isolated one-factor ablations. The fault experiments cover one persistent link failure and one fabric-wide packet-loss condition but do not evaluate link jitter or switch failure. Finally, the ECMP experiment uses static destination-port-based per-flow assignment and does not evaluate learned inter-agent coordination, traffic-class-aware splitting, flowlet routing, or cooperative multipath scheduling.

Finally, the controller selects one path from $K = 3$ retained candidates for each traffic pair during each control interval. In the reported runs, the topology graph was obtained through dynamic link-layer discovery, and the pruned breadth-first generator retained

the first K admitted paths without explicitly sorting adjacency insertion order beforehand. Exact candidate-catalogue membership can therefore depend on topology-discovery order across separate runs. Future implementations should persist authoritative candidate catalogues or impose an explicit graph-ordering rule before path generation. The scalability microbenchmark evaluates larger values of N and K, but it measures policy computation rather than topology discovery, candidate-path construction, or forwarding-rule installation. Future work will focus on four concrete extensions: evaluation on hardware or containerized multi-controller testbeds; online-safe policy fine-tuning under bounded operational constraints; traffic-aware dynamic candidate-path generation with reproducible catalogue construction; and learned service-aware multipath control for heterogeneous 6G network slices.

5. Conclusions

This paper presented an offline multi-agent reinforcement learning approach with behavior-adjusted training rewards for latency-aware SDN path control in 6G-oriented network softwarization. The proposed controller uses a centralized-training/decentralized-execution MADDPG formulation in which each traffic pair acts as an agent and selects one of $K = 3$ candidate paths. The policy is trained from logged Ryu controller transitions and deployed for live evaluation in a Ryu–Mininet SDN environment. This design avoids online exploration during deployment while preserving adaptive path-selection capability inside a software-defined control loop.

The expanded evaluation covers nine policies on the fat-tree, mesh-grid, and WAN-corridors topologies. Util-aware SP provides the strongest overall reward ranking, placing first on fat-tree and WAN-corridors and second on mesh-grid. MADDPG ranks second on fat-tree and is the best-performing learned policy on that topology, while also obtaining the lowest measured fat-tree RTT. On mesh-grid, no evaluated policy significantly outperforms MADDPG despite differences in point-estimate rank. On WAN-corridors, MADDPG is statistically tied with the policies that achieve comparable flow completion, whereas the significant reward differences involving Util-aware SP, Shortest-hops, and Min-switch-cost coincide with substantially lower completed traffic.

The mechanism analysis shows that these outcomes arise for different reasons across the three topologies. On fat-tree, the advantage of Util-aware SP is explained by lower instantaneous bottleneck utilization at comparable flow completion, RTT, hop count, and path cost; its frequent path switching directly targets the reward component with the largest weight. On mesh-grid, MADDPG converges toward near-shortest-path behavior, while the additional detours selected by several other learned policies increase path length and latency without producing measurable congestion relief. On WAN-corridors, scalar reward must be interpreted jointly with flow completion because policies operating under different delivered traffic volumes are not directly comparable.

The architectural-comparison and sensitivity results do not support a topology-independent benefit for every implemented design variant. IDDPG and Single-DDPG obtain lower point-estimate rewards than MADDPG most clearly on fat-tree, while their differences are smaller or reversed in direction on the other topologies. Because target-projection settings are not matched on fat-tree and mesh-grid, these differences cannot be attributed exclusively to the centralized critic or the per-pair multi-agent decomposition. Behavior adjustment is also topology-dependent: removing the behavior-deviation reward adjustment has no significant effect on mesh-grid, produces a non-significant reward reduction on fat-tree, and significantly improves reward, completion, utilization, and RTT on WAN-corridors. The reward-weight and hop/switch-penalty sweeps do not make

MADDPG the highest-ranked policy on any topology, but they show that the revised leading-order conclusions are not artifacts of one scalarization or penalty calibration.

The exported MADDPG decision module remains computationally lightweight, with 79,372 parameters, a 318.8 KB footprint, median all-pairs inference times of approximately 52 µs, and 95th-percentile times below 60 µs across the three topologies. The complete instrumented controller interval is substantially longer, approximately 562–617 ms, and is dominated by network-statistics polling rather than policy evaluation. The scalability microbenchmark further shows that joint policy computation remains below 0.5 ms for up to 64 traffic-pair agents and 10 candidate paths, although this benchmark excludes monitoring, candidate-path construction, and forwarding-rule installation.

The main conclusion is therefore bounded. Offline MADDPG with behavior-adjusted training rewards is not a universal replacement for topology-aware routing heuristics, and the selected behavior-adjustment weight is not uniformly beneficial across network structures. It is nevertheless a competitive and diagnostically characterized learned controller: it is the strongest learned policy on fat-tree, is significantly outperformed by no evaluated policy on mesh-grid, and remains competitive with completion-matched policies on WAN-corridors. Future work should evaluate the controller on hardware or containerized testbeds, investigate online-safe fine-tuning, develop reproducible traffic-aware candidate-path generation, and examine learned service-aware multipath control for heterogeneous 6G network slices.

Author Contributions: Conceptualization, A.E.K. and D.V.L.; methodology, A.E.K.; software, Z.O.; validation, D.V.L.; formal analysis, A.E.K.; investigation, A.E.K.; resources, Y.S.N.; data curation, Z.O.; writing—original draft preparation, A.E.K.; writing—review and editing, A.E.K.; visualization, Z.O.; supervision, D.V.L.; project administration, Y.S.N.; funding acquisition, Y.S.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan, grant number BR24993211, under the project "Development of solutions for 5G networks in ORAN concept and efficient routing and load balancing in SDN environments for Kazakhstan."

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The processed results supporting the findings of this study are reported in the article. Additional logged controller transitions, analysis scripts, and simulation outputs generated from the Ryu–Mininet experiments are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used OpenAI ChatGPT (GPT-5) for language refinement, text editing, and improvement of the manuscript structure and presentation. The authors reviewed and revised all AI-assisted output and take full responsibility for the final content of this publication. All scientific ideas, methodological design, experimental settings, analyses, results, interpretations, and conclusions were developed by the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

6G	Sixth-generation networks
AI	Artificial intelligence
BCQ	Batch-Constrained Q-learning
CQL	Conservative Q-learning
CTDE	Centralized training with decentralized execution

DDPG	Deep Deterministic Policy Gradient
ECMP	Equal-Cost Multi-Path
IDDPG	Independent Deep Deterministic Policy Gradient
MAC	Medium access control
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
NFV	Network function virtualization
RL	Reinforcement learning
RTT	Round-trip time
SDN	Software-defined networking
SNR	Signal-to-noise ratio
UDP	User Datagram Protocol
XL-MIMO	Extremely Large-Scale Multiple-Input Multiple-Output

References

1. Saad, W.; Bennis, M.; Chen, M. A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems. *IEEE Netw.* **2020**, *34*, 134–142. [CrossRef]
2. Letaief, K.B.; Chen, W.; Shi, Y.; Zhang, J.; Zhang, Y.J.A. The Roadmap to 6G: AI Empowered Wireless Networks. *IEEE Commun. Mag.* **2019**, *57*, 84–90. [CrossRef]
3. Letaief, K.B.; Shi, Y.; Lu, J.; Lu, J. Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 5–36. [CrossRef]
4. Peltonen, E.; Bennis, M.; Capobianco, M.; Debbah, M.; Ding, A.; Gil-Castiñeira, F.; Jurmu, M.; Karvonen, T.; Kelanti, M.; Kliks, A.; et al. 6G White Paper on Edge Intelligence. *arXiv* **2020**, arXiv:2004.14850.
5. Fettweis, G.P. The Tactile Internet: Applications and Challenges. *IEEE Veh. Technol. Mag.* **2014**, *9*, 64–70. [CrossRef]
6. Mao, Q.; Hu, F.; Hao, Q. Deep Learning for Intelligent Wireless Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2595–2621. [CrossRef]
7. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3133–3174. [CrossRef]
8. Wang, H.; Yan, T.; Zhou, N.; Li, X.; Wen, F.; Du, W. Enhanced Polar-Domain Channel Estimation for Near-Field XL-MIMO in Low-SNR Scenarios. *IEEE Trans. Veh. Technol.* **2025**, *74*, 19407–19419. [CrossRef]
9. Kreutz, D.; Ramos, F.M.V.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
10. Nunes, B.A.A.; Mendonca, M.; Nguyen, X.N.; Obraczka, K.; Turletti, T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1617–1634. [CrossRef]
11. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
12. Akyildiz, I.F.; Lee, A.; Wang, P.; Luo, M.; Chou, W. A Roadmap for Traffic Engineering in SDN-OpenFlow Networks. *Comput. Netw.* **2014**, *71*, 1–30. [CrossRef]
13. Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a Globally-Deployed Software Defined WAN. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 12–16 August 2013; pp. 3–14. [CrossRef]
14. Hong, C.Y.; Kandula, S.; Mahajan, R.; Zhang, M.; Gill, V.; Nanduri, M.; Wattenhofer, R. Achieving High Utilization with Software-Driven WAN. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, Hong Kong, China, 12–16 August 2013; pp. 15–26. [CrossRef]
15. Ryu Project Team. Ryu SDN Framework. Available online: https://ryu-sdn.org/ (accessed on 23 June 2026).
16. Lantz, B.; Heller, B.; McKeown, N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*; ACM: New York, NY, USA, 2010. [CrossRef]
17. Greenberg, A.; Hamilton, J.R.; Jain, N.; Kandula, S.; Kim, C.; Lahiri, P.; Maltz, D.A.; Patel, P.; Sengupta, S. VL2: A Scalable and Flexible Data Center Network. In Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, Barcelona, Spain, 17–21 August 2009; pp. 51–62. [CrossRef]
18. Al-Fares, M.; Loukissas, A.; Vahdat, A. A Scalable, Commodity Data Center Network Architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*; ACM: New York, NY, USA, 2008; pp. 63–74. [CrossRef]
19. Benson, T.; Anand, A.; Akella, A.; Zhang, M. MicroTE: Fine Grained Traffic Engineering for Data Centers. In Proceedings of the Seventh Conference on Emerging Networking Experiments and Technologies, Tokyo, Japan, 6–9 December 2011; pp. 1–12. [CrossRef]

20. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic Flow Scheduling for Data Center Networks. In Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 28–30 April 2010.
21. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1509.02971.
22. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *arXiv* **2017**, arXiv:1706.02275.
23. Boutaba, R.; Salahuddin, M.A.; Limam, N.; Ayoubi, S.; Shahriar, N.; Estrada-Solano, F.; Caicedo, O.M. A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities. *J. Internet Serv. Appl.* **2018**, *9*, 16. [CrossRef]
24. Xie, J.; Yu, F.R.; Huang, T.; Xie, R.; Liu, J.; Wang, C.; Liu, Y. A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 393–430. [CrossRef]
25. Xu, Z.; Tang, J.; Meng, J.; Zhang, W.; Wang, Y.; Liu, C.H.; Yang, D. Experience-Driven Networking: A Deep Reinforcement Learning Based Approach. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 1871–1879. [CrossRef]
26. Casas-Velasco, D.M.; Rendon, O.M.C.; da Fonseca, N.L.S. DRSIR: A Deep Reinforcement Learning Approach for Routing in Software-Defined Networking. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 4807–4820. [CrossRef]
27. Dulac-Arnold, G.; Levine, N.; Mankowitz, D.J.; Li, J.; Paduraru, C.; Gowal, S.; Hester, T. Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis. *Mach. Learn.* **2021**, *110*, 2419–2468. [CrossRef]
28. Levine, S.; Kumar, A.; Tucker, G.; Fu, J. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv* **2020**, arXiv:2005.01643.
29. Prudencio, R.F.; Maximo, M.R.O.A.; Colombini, E.L. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 10237–10257. [CrossRef] [PubMed]
30. Fujimoto, S.; Meger, D.; Precup, D. Off-Policy Deep Reinforcement Learning without Exploration. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 2052–2062.
31. Kumar, A.; Fu, J.; Tucker, G.; Levine, S. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32. [CrossRef]
32. Kumar, A.; Zhou, A.; Tucker, G.; Levine, S. Conservative Q-Learning for Offline Reinforcement Learning. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1179–1191. [CrossRef]
33. Ordonez-Lucena, J.; Ameigeiras, P.; Lopez, D.; Ramos-Munoz, J.J.; Lorca, J.; Folgueira, J. Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 80–87. [CrossRef]
34. Barakabitze, A.A.; Ahmad, A.; Mijumbi, R.; Hines, A. 5G Network Slicing Using SDN and NFV: A Survey of Taxonomy, Architectures and Future Challenges. *Comput. Netw.* **2020**, *167*, 106984. [CrossRef]
35. Mestres, A.; Rodriguez-Natal, A.; Carner, J.; Barlet-Ros, P.; Alarcón, E.; Solé, M.; Muntés-Mulero, V.; Meyer, D.; Barkai, S.; Hibbett, M.J.; et al. Knowledge-Defined Networking. *ACM SIGCOMM Comput. Commun. Rev.* **2017**, *47*, 2–10. [CrossRef]
36. Miuccio, L.; Riolo, S.; Samarakoon, S.; Bennis, M.; Panno, D. Emerging Generalized Wireless MAC Communication Protocols via Abstraction. *IEEE Open J. Commun. Soc.* **2025**, *6*, 6842–6865. [CrossRef]
37. Jadoon, M.A.; Pastore, A.; Navarro, M.; Valcarce, A. Learning Random Access Schemes for Massive Machine-Type Communication With MARL. *IEEE Trans. Mach. Learn. Commun. Netw.* **2024**, *2*, 95–109. [CrossRef]
38. Nurakhov, Y.; Kyzyrkanov, A.; Otarbay, Z.; Lebedev, D. Adaptive Software-Defined Network Control Using Kernel-Based Reinforcement Learning: An Empirical Study. *Appl. Sci.* **2025**, *15*, 12349. [CrossRef]
39. Kyzyrkanov, A.E.; Nurakhov, Y.S.; Otarbay, Z.; Lebedev, D.V. Log-Driven Proximal Policy Optimization for Adaptive Traffic Control in Software-Defined Networks. *Appl. Sci.* **2026**, *16*, 4424. [CrossRef]
40. Efron, B.; Tibshirani, R.J. *An Introduction to the Bootstrap*; Chapman and Hall/CRC: Boca Raton, FL, USA, 1994. [CrossRef]
41. Hemerik, J.; Goeman, J.J. Exact testing with random permutations. *TEST* **2018**, *27*, 811–825. [CrossRef] [PubMed]
42. Cliff, N. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychol. Bull.* **1993**, *114*, 494–509. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.