AGREEMENT BY CONSTRUCTION: MULTI-AGENT RL FOR DATACENTER NETWORK POWER MANAGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern datacenters (DC) face escalating energy demands, with power consumption becoming a critical bottleneck for computational scaling. To mitigate this, we introduce LinkRL, a decentralized multi-agent reinforcement learning framework for managing power in DC networking. Our agent learns the optimal timing windows to toggle network links between active and low-power states. Specifically, we target the substantial energy these links consume while idle during computation phases. Networking infrastructure accounts for approximately 20% of the total DC power, and the inefficiency of idle links presents a significant optimization opportunity. In LinkRL, we train agents to predict traffic patterns, enabling preemptive sleep and wake-up. We guarantee agreement between agents managing a shared link, by construction and without communication, by restricting each agent's input to features both ends observe identically. We train our framework on real-world LLM training workloads using a high-fidelity networking simulator that serves as NVIDIA's primary platform for design, verification, and benchmarking of its networking products. LinkRL achieves 10.9% datacenter power savings while maintaining latency, outperforming baselines. The same policy transfers to other distributed-compute workloads without retraining. Reallocating these savings to compute yields an 8.9% GPU performance gain. LinkRL is integrated into the design of NVIDIA's next-generation networking hardware.

1 INTRODUCTION

Energy consumption has become a prominent constraint on datacenter scaling, with networking infrastructure contributing roughly 20% (Abts et al., 2010; Barroso et al., 2018). Network links connect nodes like GPUs or switches in the datacenter. These links remain idle 50–80% of the time, especially during GPU computation phases. Despite being inactive, idle links consume nearly as much power as active ones. This inefficiency presents a significant energy-saving opportunity. NVIDIA recently introduced a low-power link state that substantially reduces idle power consumption, leaving open the question of when to enter and exit this state. Answering it requires anticipating traffic: sleeping links during idle gaps and waking them before traffic returns. The bursty, irregular communication of modern ML workloads is too complex for the heuristics in production (IEEE 802.3 Working Group, 2010; Kazimirsky & Sucher, 2024), motivating a learned, predictive policy.

We propose LinkRL: lightweight per-link agents that learn an adaptive policy to toggle their link between active and low-power states based on observed traffic patterns. The key innovation in LinkRL is predictive traffic modeling. Agents learn to anticipate future network activity, allowing them to: (i) preemptively wake links before traffic arrives, minimizing latency penalties, and (ii) identify extended idle periods where links can safely sleep, maximizing energy savings. For training and evaluation we use DNetSim, a high-fidelity datacenter network simulator to model Large Language Model (LLM) training workloads and report both performance impact and power savings. The overall flow of LinkRL is detailed in Figure 1.

Two properties of the problem motivate this design. First, each agent's sleep and wake decisions reshape the traffic neighboring agents observe. Because a sleeping link both delays and diverts traffic, current actions shift the distribution of subsequent network system state, creating a sequential-decision problem under non-stationary dynamics. Second, agents must act from local observations alone as communicating the joint observation is infeasible during sleep. This rules out standard

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

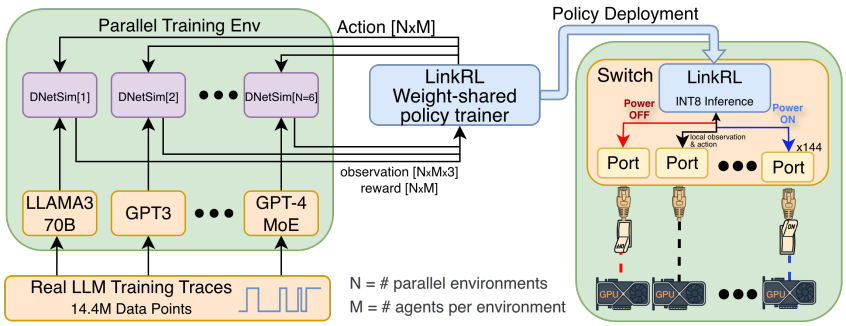


Figure 1: End-to-end LinkRL system. **Left:** parallel DNetSim simulators replay 14.4M data points from real LLM training traces, including Llama3-70B, GPT-3, GPT-4 MoE, and others. **Middle:** a centralized trainer aggregates per-link rollouts and updates a single shared policy via PPO. **Right:** the policy is quantized to INT8 and deployed on dedicated switch hardware.

Multi-Agent RL (MARL) coordination mechanisms such as explicit message passing, and centralized coordinators. This zero-communication constraint also exposes a failure mode of vanilla independent PPO (IPPO) agents (de Witt et al., 2020) that observe only their own port: the two endpoints of a physical link can disagree on whether it should be active or asleep. LinkRL keeps the IPPO learner and adds a symmetric state construction that guarantees endpoint agreement without communication.

Across evaluated scenarios, LinkRL delivers 10.9% total datacenter power savings, with findings supported by hardware correlation studies. An idealized upper bound caps achievable savings at $\approx$ 15% under representative LLM-training conditions, so LinkRL captures most of this headroom; see Appendix C for the calculation. When these power savings are reallocated to compute resources, projections indicate an 8.9% net throughput gain.

Our primary contributions are:

1. **Agreement by construction.** A *symmetric-features* state construction yields consistent multi-agent decisions under a shared policy without runtime inter-agent communication. This addresses a key failure mode of vanilla IPPO agents that observe only their own port: endpoint disagreement on the state of a shared physical link. A centralized critic (MAPPO) does not remove the disagreement either. Removing the symmetric-features construction collapses net throughput gain from 8.9% to 1.6%.
2. **A distributed RL framework** that manages datacenter network links at fine granularity, adaptively balancing energy efficiency against performance under sub-microsecond latency, minimal chip area, and zero inter-agent communication constraints. A compact CNN-MLP policy captures temporal traffic patterns for proactive power state decisions.
3. **Real-World Impact** through empirical evidence: 10.9% datacenter power savings translating to 8.9% GPU performance improvement, with the technology integrated into the design of next-generation networking hardware.
4. **Comprehensive analysis** through ablation studies isolating the contribution of joint sleep/wake control, symmetric features, and multi-link training.

2 Related Work

Machine learning (ML) for Datacenter Network and Resource Optimization. ML has demonstrated success in datacenter energy efficiency, increasingly including network power. GreenTE.ai applies deep RL to adaptively control router/switch power states, using neural-network forward propagation for fast decisions (Pan et al., 2021). In mobile networks, deep RL has been used to tune base-station antenna tilt and transmission power for throughput under quality-of-service constraints (Tran et al., 2024). Our work differs by targeting datacenter link-level granularity, LLM-training traffic patterns, and distributed per-link agents. RL has also been deployed inside network devices for congestion control (Tessler et al., 2022; Fuhrer et al., 2023), with per-flow agents and no shared-resource agreement problem. RL has also been applied to other datacenter resources, from virtual-

machine placement (Wang et al., 2021a) to GPU and CPU power capping (Wang et al., 2024; Azimi et al., 2020); these target individual compute nodes, whereas we target the interconnect.

Heuristic and Legacy Link Power Management. Our work competes against two widely deployed non-learning methods: *threshold-based* policies that sleep links after a fixed idle period (IEEE 802.3 Working Group, 2010) and *pattern-predictor* methods that track recurring idle cycles to preempt wake-up (Kazimirsky & Sucher, 2024). Both are deployed in silicon. To strengthen the baseline set, we additionally construct a *supervised + Bayesian* regressor that predicts idle durations from traffic history using Bayesian structural time series (Scott & Varian, 2014). All three struggle with the dynamic traffic patterns of modern ML workloads. We benchmark against representatives of all three in §5.3 with full descriptions in §5.

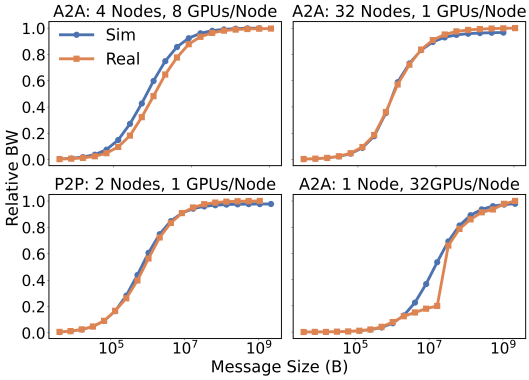


Figure 2: Bandwidth (y-axis) versus message size (x-axis) for measured silicon (**Real**) and simulated (**Sim**) runs across different communication patterns, topologies, and network types.

Multi-Agent Systems under Hard Constraints. Multi-agent reinforcement learning (MARL) has been deployed across diverse real-world domains. Intelligent transportation systems use MARL for traffic signal control, with agents coordinating via shared critics or inter-agent messaging (Chu et al., 2019). Autonomous vehicle fleets apply cooperative MARL for safe navigation (Kiran et al., 2021), and smart grids use distributed agents to balance demand and renewable generation (Wang et al., 2021b). These settings run on second- or millisecond-scale timesteps, leaving room for runtime communication or centralized coordination. In contrast, our datacenter link control operates at sub-microsecond cycles. These latency constraints preclude inter-agent communication entirely, forcing agents to act fully independently based on shared training (parameter sharing across homogeneous agents follows Gupta et al., 2017) and decentralized inference. In the common-information approach (Nayyar et al., 2013) and MACKRL (Schroeder de Witt et al., 2019), common knowledge selects a joint policy that agents then execute with private observations, so two agents can still take different actions. LinkRL restricts the two agents on a shared link to observations of the link itself, which removes that possibility.

3 Problem Formulation

We formulate dynamic link power management as an RL problem: learn a policy π that toggles links between active and low-power states, minimizing energy consumption subject to application performance.

Three properties make this setting non-standard for MARL. First, sub-microsecond deadlines preclude centralized control and inter-agent communication. Second, endpoint agents on a shared link must agree on its power state without messaging; a vanilla IPPO policy with per-port observations has no mechanism to enforce this agreement. Third, actions reshape the traffic neighbors observe, creating non-stationary local dynamics under partial observability.

3.1 Environment: DNetSim

We train and evaluate LinkRL in DNetSim, NVIDIA's internal packet-level network simulator and the main gateway for all design, validation, and benchmarking of NVIDIA's hundreds of networking features and products. Built on the OMNeT++ framework (Varga, 2001), DNetSim models switches at the microarchitecture level and captures per-link buffer and packet timing dynamics within a shared environment, so one agent's actions affect the traffic seen by other agents. We parallelize training by running multiple instances on different LLM workloads, each reporting per-link telemetry and application-level metrics.

DNetSim has been validated against NVIDIA's deployed networking hardware with less than 5% error in network-performance predictions; see Figure 2 and the per-panel breakdown in Appendix E.

It accurately models the $> 100\mu s$ entry and wake-up latencies of low-power transitions, capturing both the packet-level effect—delays from packets arriving at sleeping links—and the downstream impact on job completion time. DNetSim is exposed to the agents as a Gym-compatible environment (Brockman et al., 2016). Per-link observations and the discretization applied to them are detailed in §3.2, and the per-state power model the simulator emits for each link is specified in Appendix C.

3.2 MDP FORMULATION

Given this simulated environment, we formulate datacenter link power management as a decentralized, partially observable Markov decision process (Oliehoek & Amato, 2016). The system state s_t summarizes traffic, queue occupancies, and link power states across the network. Each agent receives only a local partial observation o_t of its associated link and learns a policy $\pi(a_t \mid o_t)$ over its link's power state.

Observation Representation. At each timestep t, every agent receives a partial local observation o_t of the system state s_t. Due to hardware observability constraints, features such as link buffer occupancies are part of s_t but cannot be exposed to the agent. The observation is a compact feature vector containing:

- **Consecutive Idle Time:** The time elapsed since the last data packet was transmitted or received on the link.
- **Time in Current Power State:** Duration the link has spent in its current power state.
- **Recent Traffic-Triggered Wake-Up:** A binary flag indicating if the link was recently forced to transition from a low-power to an active state due to incoming traffic.
- **Traffic History Features:** A summarized representation of recent traffic patterns, such as a short history of idle period durations, to help predict future activity.

Since the inference hardware resides on the switch ASIC, our system must handle two types of network links: switch-to-switch links with agents at both endpoints, and GPU-to-switch links with an agent only at the switch port. For switch-to-switch links, both agents must agree before transitioning to or maintaining a low-power state. For GPU-to-switch links, we employ a primary-secondary approach where the switch initiates low-power transitions and the GPU complies when not actively transmitting data.

Symmetric Features. We restrict $\tilde{o}_t$ to features that are properties of the link itself, rather than of an endpoint, so each endpoint computes the same $\tilde{o}_t$ from its own local observation alone. Formally, if $g : o_t^A \mapsto \tilde{o}_t$ and $h : o_t^B \mapsto \tilde{o}_t$ project onto link-level quantities, the construction guarantees $g(o_t^A) = h(o_t^B) = \tilde{o}_t$ at every cycle in which both endpoints share a clock-aligned view of the link. Under a deterministic shared policy π_θ, both endpoints then emit identical actions without runtime messaging. This is a structural rather than statistical guarantee: with per-port observations, vanilla IPPO (de Witt et al., 2020) has no mechanism for endpoint agreement, and centralized-critic methods such as MAPPO (Yu et al., 2022) and COMA (Foerster et al., 2018) encourage it in training but cannot enforce it at execution. Symmetric features make disagreement impossible by construction. The cost is a restricted observation space, as we must drop port-asymmetric features. Section 5.5 ablates this design against vanilla IPPO with per-port observations.

Action Space. At each timestep t, the agent selects an action a_t from a discrete binary action space: (i) **enter/remain in the low-power state**, or (ii) **enter/remain in the active state**.

Reward Function. The reward r_t combines an energy bonus with a performance impact penalty:

- **Energy Savings Bonus:** A positive reward proportional to the duration the link successfully spends in a low-power state, directly reflecting conserved energy.
- **Forced Wake-Up Penalty:** A significant negative reward issued if a data packet arrives when the link is in a low-power state, thereby forcing an unplanned, high latency transition to the active mode.

The overall per-step reward is a weighted combination of these factors:

$$r_t = \alpha \cdot \text{TimeInLowPowerState}_t - \beta \cdot \text{ForcedWakeUpEvent}_t \tag{1}$$

where $\alpha > 0$ and $\beta > 0$ are hyperparameters that tune the trade-off. We use $\alpha = 1.0, \beta = 40.0$, selected by sweeping over a held-out validation set (Appendix D). The large penalty ratio reflects

our low tolerance for performance disruption: a single forced wake-up is weighted as heavily as 40 control cycles spent in low-power state.

Optimization Objective. Each agent aims to learn an optimal policy that maximizes the expected sum of discounted future rewards: $\pi^* = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where $\gamma \in [0, 1)$ is the discount factor. We use $\gamma = 0.8$, sized to the relatively short-term impact of each action. The shaped per-step rewards provide denser feedback than the inherently sparse end-of-episode signal.

Evaluation metrics. Throughout the paper we report three metrics: *Power Reduction*, *Job-Time Overhead (JTO)*, and *Net Throughput Gain (NTG)*; we formally define them in §5.1.

4 METHOD

In this section, we describe our implementation of LinkRL. We cover our RL agent architecture, deployment optimizations for switch hardware, and our multi-agent training pipeline.

4.1 RL AGENT ARCHITECTURE

We employ a distributed multi-agent approach where each network link is managed by a dedicated, lightweight agent. These agents are trained with IPPO (Schulman et al., 2017; de Witt et al., 2020), with weights shared across all agents. Endpoint agreement on a shared link is guaranteed by the symmetric state construction of Section 3.2. Coordination across links is not enforced but emerges from training all agents concurrently in the shared environment (Section 4.3). This distributed setup with independent agents enables scalable deployment across potentially thousands of links in a datacenter while maintaining low computational overhead per link.

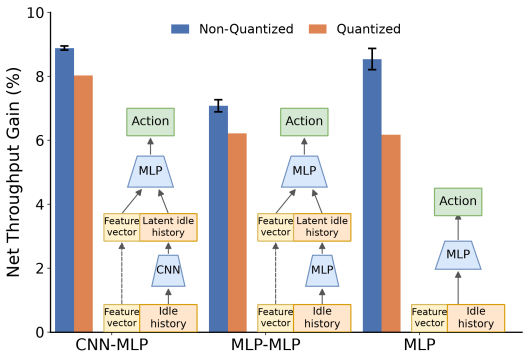


Figure 3: Net Throughput Gain (%) for the three architectures before and after INT8 quantization. INT8 reduces NTG from 8.9% to 8.0% for CNN-MLP and from 8.5% to 6.2% for the MLP.

We compare three architectures (Figure 3): a baseline **MLP** that flattens the full observation; an **MLP-MLP** that preprocesses traffic history with an initial MLP encoder; and a **CNN-MLP** that uses a 1D CNN as a temporal encoder over the idle-time history, exploiting the periodic structure of LLM-training traffic. Full architecture details appear in Appendix A.

Agents share parameters during training but act independently at runtime based on local observations, with consistent decisions on switch-to-switch links guaranteed by symmetric features (§3.2).

4.2 DEPLOYMENT OPTIMIZATIONS

The target switch's 25 μs control cycle is shared across 144 ports, giving each per-port inference $\approx$ 175 ns to complete. This nanosecond budget rules out inter-agent communication at runtime and forces a compact on-switch neural network. We quantize the CNN-MLP to INT8, compressing it by a factor of 4 with about 10% relative NTG degradation, against 27% for the MLP (Figure 3). We further discretize inputs at the 25 μs control-cycle granularity to lower on-chip compute while preserving accuracy. We detail the on-chip footprint—area, power, and cell count in Appendix I.

4.3 MULTI-AGENT TRAINING PIPELINE

Multiple parallel DNetSim instances drive different LLM training traces, each providing observations to its link agents and returning rewards and next states (Figure 1). Agent decisions are coupled, e.g., putting a link to sleep delays the packets queued for it, which through backpressure alters the traffic patterns observed on neighboring links. We therefore train a single shared policy on the aggregated experience of all agents, allowing it to implicitly learn coordinated behaviors that account for network-wide effects. Section 5.5 ablates this design against single-link training.

Per iteration, all agents collect trajectory rollouts in parallel, advantage estimates are computed via Generalized Advantage Estimation (GAE) (Schulman et al., 2015), and a single shared-parameter PPO update is taken on a uniformly-sampled minibatch from the aggregated buffer (Appendix G). The CNN-MLP converges in ≈ 4 days on AMD EPYC Genoa CPU cores, using 6 parallel DNetSim environments plus 1 aggregation core (Appendix A).

5 Experiments

In this section we show that LinkRL outperforms baselines and generalizes stably across domains, with ablations isolating the design choices behind these gains.

5.1 Evaluation Metrics

Power Reduction: The percentage decrease in total compute-node power, obtained by projecting per-link simulation output onto a compute-node power envelope:

$$\text{Power Reduction } (\%) = \left(\frac{P_{\text{base}} - P_{\text{LinkRL}}}{P_{\text{base}}} \right) \times 100,$$

where P_{base} is measured from real hardware and P_{LinkRL} is the node power under LinkRL, derived from the DNetSim output. The deployment fabric is network-dense, with 144-port switches serving 72 GPUs per rack and inter-rack links on the remaining ports, so networking is a large share of node power: placing all links in the low-power state reduces node power by 34% relative to full load (Appendix C).

Job-Time Overhead (JTO): the percentage increase in job completion time caused by link-state transition latencies, measured directly in simulation before any power reallocation:

$$\text{JTO } (\%) = \left(\frac{T_{\text{LinkRL}} - T_{\text{base}}}{T_{\text{base}}} \right) \times 100.$$

Net Throughput Gain (NTG): the projected improvement in application performance after the saved power is reallocated to compute, NTG $= 0.9 \cdot$ PowerReduction $-$ JTO, where the 0.9 factor maps a 1% increase in compute power to $\approx 0.9\%$ throughput gain, derived from silicon measurements (Appendix C).

5.2 Experimental Setup

Our primary training and evaluation domain is real-world traffic from large-scale distributed LLM training. We focus on this domain because LLM training represents a dominant fraction of modern datacenter compute and exhibits the long, irregular idle gaps that admit dynamic link-power management. The full scope justification appears in §5.4. Our training dataset includes six distinct traces from models such as Llama3 70B, GPT-3 175B MoE, GPT-4 MoE, GPT-3 40B, and GPT-3 175B Dense, comprising 14.4 million data points. Each trace records the communication operations executed during a workload's runtime, which are then simulated in DNetSim to replicate the network traffic present during the original workload's execution. All training and evaluation run on a simulated two-level fat-tree with switch-to-switch links, so every such link is controlled by an agent at each end; node-level power is projected onto the deployment fabric of Appendix F. For generalization testing, we use six additional traces from different LLM workloads, including Llama3 8B, Llama2 70B, and GPT-3 43B, executed on separate datacenter hardware at varying scales. For cross-domain transfer, we further evaluate on real HPC traces and generic ML training traces without retraining.

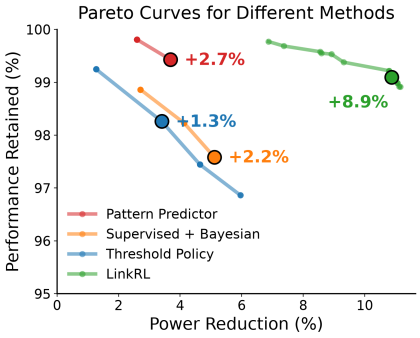


Figure 4: Pareto curves showing power reduction vs. performance retained % i.e., $1-$JTO. Annotations indicate net throughput gain from power reallocation.

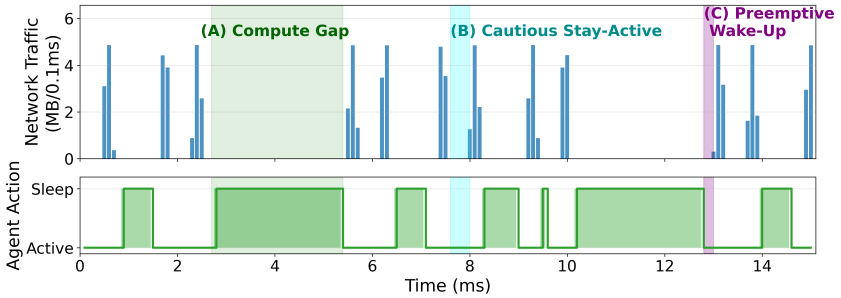


Figure 5: Micro-timeline of LinkRL controlling a single link over a 15 ms window from a held-out trace. Top: network traffic intensity. Bottom: agent action (active/sleep). (A) The agent identifies the onset of a 2.7 ms compute phase and transitions to low-power state. (B) The agent remains active through a 0.4 ms transient idle gap where the wake-up penalty would exceed the energy savings. (C) The agent initiates wake-up 200 μs *before* the next communication burst, masking the physical transition latency. A reactive threshold policy would commonly miss the sleep opportunity in (A) or incur a forced wake-up penalty at (C).

Table 1: Architectures and baselines on held-out training-trace segments. Net Throughput Gain reported with std across three random seeds.

Architecture	Power Reduction	Job-Time Overhead	Net Throughput Gain	Parameter Count
MLP	10.27%	0.73%	8.51% ±0.25%	3,924
MLP-MLP	8.87%	0.92%	7.06% ±0.14%	24,388
CNN-MLP	**10.87%**	**0.90%**	**8.88%** ±0.05%	9,324
Threshold Policy	3.39%	1.74%	1.31%	—
Pattern Predictor	3.67%	0.57%	2.73%	—
Supervised + Bayesian	5.12%	2.41%	2.20%	4,737

Our primary results use the CNN-MLP architecture, which captures temporal traffic patterns and, unlike the smaller MLP, retains about 90% of its NTG under INT8 quantization (Figure 3). Full training hyperparameters are in Appendix A.

We benchmark against three baselines. The first two are deployed in silicon today, while the third is a baseline we construct to provide an ML-based comparison. We select all hyperparameters, LinkRL and baselines, using a held-out validation split of the training traces. We report numbers on test traces never seen during training or tuning.

Threshold Policy: a simple heuristic that transitions links to low-power state after a fixed inactivity threshold.

Pattern Predictor: a scoreboard heuristic that tracks recurring idle cycles to anticipate the next idle window.

Supervised + Bayesian: a baseline we construct using Bayesian structural time series (Scott & Varian, 2014). The offline regressor predicts idle durations from traffic history, with a Bayesian adjustment that conservatively underestimates idle length to avoid wake-up penalties.

A hyperparameter sweep over the validation split selected each baseline's operating point: the Threshold Policy's inactivity threshold at 400 μs, the Pattern Predictor's idle-time discretization with 16 buckets, and the Supervised + Bayesian over-estimation penalty at $\lambda = 2$. Full sweeps appear in Appendices D and B.

5.3 Results

Table 1 compares the three architectures of §4.1 on held-out segments of the LLM training traces, where mean and std are measured across three random seeds. The best CNN-MLP reaches 10.9% power reduction at 0.9% JTO, for 8.9% NTG. The 3,924-parameter MLP also reaches 8.5% NTG, showing that even very small models capture the key patterns. All three learned architectures more

Table 2: Generalization across workloads.

	LLM-training (In-domain Held Out)			HPC	Generic ML
Method	Power Red.	JTO	NTG	NTG	NTG
Threshold Policy	3.23%	1.80%	1.11%	**12.96%**	3.26%
Pattern Predictor	6.26%	0.57%	5.06%	10.33%	2.68%
Supervised + Bayesian	3.83%	1.48%	1.97%	10.81%	4.07%
LinkRL (Ours)	**9.03%**	**1.01%**	**7.12%** $\pm 0.043\%$	12.14% $\pm 1.13\%$	**5.27%** $\pm 1.13\%$

Table 3: Ablation study results

Method	Power Reduction	Job-Time Overhead	Net Throughput Gain
LinkRL (full)	**10.87%**	**0.90%**	**8.88%**
Wake-up only	7.41%	1.97%	4.70%
Vanilla IPPO (per-port observations)	4.26%	2.27%	1.56%
MAPPO (per-port observations)	8.07%	1.95%	5.31%
Single-link training	8.02%	1.45%	5.77%

than triple the NTG of the best non-learned baseline and significantly outperform the Supervised + Bayesian baseline. The learned policy anticipates traffic and adapts to inter-agent dynamics through training, neither of which a classic predictor can do. Figure 4 shows the resulting Pareto frontier, and Figure 5 shows a recorded single-link policy exhibiting the predictive sleep entry, transient-gap rejection, and preemptive wake-up behaviors.

5.4 Scope and Cross-Domain Generalization

Why LLM training. We focus mainly on LLM training workloads. Three observations motivate this. (1) LLM training is one of the dominant fractions of datacenter compute and energy spend (Patterson et al., 2021), making it the highest-impact target for link-level power management. (2) Dynamic link-power management is viable only when idle gaps are long enough to amortize the $> 100\,\mu s$ latency of each sleep and wake transition. Inference workloads, with predominantly sub-$100\,\mu s$ idle gaps (Pope et al., 2023; Verma & Vaidya, 2023), are therefore out-of-regime by construction. (3) Most other HPC workloads occupy the opposite extreme: long idle gaps where simple threshold heuristics already capture most savings. LinkRL targets the intermediate regime where idle gaps are long enough to admit sleep but irregular enough to defeat heuristics.

Out-of-regime evaluation. To validate observation (3) above and test cross-domain transfer, we evaluate LinkRL without retraining on real HPC traces and on generic ML training traces. The HPC evaluation traces are captured from seven different sources, totaling 19.6M data points, and detailed in full in Appendix K. These traces exhibit substantially longer and more regular idle gaps than LLM-training traces, with average idle gap of 18.6 ± 3.0 ms across the HPC traces vs. 1.7 ± 1.1 ms across the twelve LLM workloads. The generic ML training traces capture the alternating compute and communication phases common to distributed ML training, with cycle length and idle/active durations varying per cycle. They represent the broader class of non-autoregressive distributed-ML workloads (e.g., vision, classification, recommendation), structurally similar to LLM training but distinct in detail. Table 2 reports NTG across all three settings.

LinkRL generalizes stably across all three domains in Table 2. The held-out LLM set spans new model sizes from 8B to 175B, both Dense and MoE architectures, and multiple parallelism strategies. LinkRL achieves the highest NTG on this set. On HPC, it trails the threshold heuristic by less than 1 NTG point, the empirical signature of "regime where heuristics already work." On generic ML traces, it outperforms all baselines. This cross-domain stability shows LinkRL learns transferable traffic patterns that apply beyond the training distribution.

5.5 Ablations and Sensitivity

To understand the impact of key design decisions in our approach, we perform several ablation studies, summarized in Table 3.

Wake-up only. We remove the agent's control over sleep-entry decisions, leaving only wake-up control: low-power entry uses a fixed idle threshold, as in the Threshold Policy, while the agent

decides only when to wake. Full control outperforms wake-up-only on both axes, indicating that controlling sleep entry teaches the agent system dynamics that wake-up-only cannot capture.

Vanilla IPPO with per-port observations. We keep the IPPO learner unchanged and give each port agent its own asymmetric observation, without the symmetric per-link construction of §3.2. The link-level approach with symmetric features improves NTG from 1.6% to 8.9%. Without an explicit agreement mechanism, the two endpoint agents disagree on link state often enough to dominate the reward.

MAPPO with per-port observations. We replace the local critic with a centralized critic that sees both endpoints' observations during training, keeping per-port observations at execution (Yu et al., 2022). Compared with vanilla IPPO on the same per-port observations, the centralized critic raises NTG from 1.6% to 5.3%, but the two endpoints still disagree on 4.4% of control cycles, whereas LinkRL's disagreement is zero by construction, and NTG remains 3.6 points below LinkRL's 8.9%. Training-time coordination does not substitute for agreement by construction.

Single-link training. We remove multi-link training by collecting rollouts from only one designated agent during PPO updates, discarding experiences from all other agents while keeping the simulation environment unchanged. Multi-link training reaches 8.9% NTG vs. 5.8% for single-link training, indicating that exposure to diverse traffic patterns and to the coupled dynamics between links is itself a source of generalization.

6 Discussion and Conclusion

LinkRL shows that decentralized RL agents can manage datacenter network power even when hardware timing leaves no room for them to communicate. Two findings carry the result. First, the symmetric-features construction makes endpoint agents agree on a shared link by construction rather than by communication or by training-time coordination; removing it, leaving vanilla IPPO with per-port observations, collapses NTG from 8.9% to 1.6%. Second, training every agent concurrently in the shared simulation environment lets each agent implicitly learn how its own actions reshape the traffic seen by its neighbors, raising NTG from 5.8% under single-link training to 8.9%. Together, these findings suggest that for MARL settings where deadlines preclude communication, coordination can emerge from state symmetry and shared-environment training rather than from explicit messaging or centralized critics. Cross-domain evaluation further shows the resulting policy is stable: without retraining, LinkRL leads on generic ML traces and stays within 1 NTG point of a heuristic engineered for HPC's regime. By enabling more computation within existing power envelopes, LinkRL also offers a path toward sustainable scaling of modern AI workloads without proportional infrastructure expansion.

When does the symmetric-features construction generalize? Whenever (i) two or more agents share control of a single resource, (ii) runtime communication is constrained, and (iii) the action-relevant information is captured by features each agent can compute locally and that are invariant to agent identity. We expect these conditions to hold in many hardware-adjacent control problems where deadlines structurally rule out coordination protocols.

Limitations. Our evaluation simulates a two-level fat-tree running a single LLM-training job; rationale in §5.4. Because the targeted multi-tier low-power link states are only available in next-generation switch hardware, evaluation relies on simulation validated against deployed hardware (§3.1, $< 5\%$ error). Because LinkRL operates only at the physical layer, the network's routing and flow-control mechanisms are unaffected. Wake-up delays are absorbed as ordinary queueing latency (Appendix H). For robustness against distribution shift, we designed a hybrid safety guardrail that falls back to a heuristic policy when the forced-wake rate exceeds a configurable threshold (Appendix J). The guardrail was never triggered in our experiments.

Future work. On-switch PPO updates are currently precluded by hardware constraints, so agents are trained offline and deployed with frozen weights. Enabling on-switch fine-tuning would let policies continuously adapt to live traffic. The same shared-policy approach also extends naturally to similar power-management features in other link technologies and to inter-node fabrics carrying multiple concurrent jobs.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, to draft sections of the paper from author-provided content, and for literature retrieval and discovery. We have not used generative AI tools for synthetic data generation, hypothesis refinement, research methodology design, mathematical proofs, or result interpretation. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3.1 describes the simulation environment and its validation against deployed hardware, with the per-panel error breakdown in Appendix E. Appendix A lists all training hyperparameters, network sizes, and training compute. Appendix B specifies the learned baseline, and Appendix D reports the hyperparameter sweeps that selected every operating point. Appendix C gives the full power-projection methodology, and Appendix K characterizes every training and evaluation trace. Results in Table 1 report mean and standard deviation across three random seeds.

REFERENCES

Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*, pp. 338–347, 2010. doi: 10.1145/1815961.1816004.

Reza Azimi, Chao Jing, and Sherief Reda. Powercoord: Power capping coordination for multi-cpu/gpu servers using reinforcement learning. *Sustainable Computing: Informatics and Systems*, 28:100412, 2020.

Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. *The Datacenter as a Computer: Designing Warehouse-Scale Machines*. Morgan and Claypool, San Rafael, CA, 3rd edition, 2018. doi: 10.2200/S00874ED3V01Y201809CAC046.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3): 1086–1095, 2019.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, 2018.

Benjamin Fuhrer, Yuval Shpigelman, Chen Tessler, Shie Mannor, Gal Chechik, Eitan Zahavi, and Gal Dalal. Implementing reinforcement learning datacenter congestion control in NVIDIA NICs. In *IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 331–343, 2023.

Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems (AAMAS) Workshops*, 2017.

IEEE 802.3 Working Group. Ieee standard for energy-efficient ethernet. Technical Report 802.3az-2010, IEEE, 2010.

Amit Kazimirsky and Nir Sucher. Predicting inactivity patterns for a signal conductor, 2024. U.S. Patent No. 12,081,361 B2, granted September 3, 2024 (pre-grant publication US 2024/0073058 A1). Assignee: Mellanox Technologies Ltd.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control*, 58(7):1644–1658, 2013.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, 2016.

Tian Pan, Xiaoyu Peng, Qianqian Shi, Zizheng Bian, Xingchen Lin, Enge Song, Fuliang Li, Yang Xu, and Tao Huang. GreenTE.ai: Power-aware traffic engineering via deep reinforcement learning. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pp. 1–6. IEEE, 2021.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2023.

Christian Schroeder de Witt, Jakob Foerster, Gregory Farquhar, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. Multi-agent common knowledge reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 9927–9939, 2019.

John Schulman, Philipp Moritz, Sergey Levine, Michael I Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Steven L Scott and Hal R Varian. Predicting the present with bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation*, 5(1-2):4–23, 2014.

Chen Tessler, Yuval Shpigelman, Gal Dalal, Amit Mandelbaum, Doron Haritan Kazakov, Benjamin Fuhrer, Gal Chechik, and Shie Mannor. Reinforcement learning for datacenter congestion control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 12615–12621, 2022.

Dinh-Hieu Tran, Nguyen Van Huynh, Soumeya Kaada, Van Nhan Vo, Eva Lagunas, and Symeon Chatzinotas. Deep reinforcement learning for network energy saving in 6g and beyond networks. *arXiv preprint arXiv:2408.10974*, 2024.

András Varga. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, pp. 65. sn, 2001.

Shashank Verma and Neal Vaidya. Mastering LLM techniques: Inference optimization. NVIDIA Technical Blog, November 2023. URL `https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization/`. Accessed: 2026-02-06.

B. Wang, F. Liu, and W. Lin. Energy-efficient vm scheduling based on deep reinforcement learning. *Future Generation Computer Systems*, 125:616–628, 2021a. doi: 10.1016/j.future.2021.07.023.

Jianhong Wang, Wangkun Xu, Yunjie Gu, Wenbin Song, and Tim C. Green. Multi-agent reinforcement learning for active voltage control on power distribution networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 3271–3284, 2021b.

Yiming Wang, Meng Hao, Hui He, Weizhe Zhang, Qiuyuan Tang, Xiaoyang Sun, and Zheng Wang. Drlcap: Runtime gpu frequency capping with deep reinforcement learning. *IEEE Transactions on Sustainable Computing*, 9(5):712–726, 2024.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

A HYPERPARAMETERS

Table 4 lists the full set of hyperparameters used to train the CNN-MLP policy reported in the main experiments. Values were selected by sweeping over a held-out validation set drawn from the training traces.

Table 4: LinkRL hyperparameters

Reward weights	
α (energy savings bonus)	1.0
β (forced wake-up penalty)	40.0
PPO configuration	
Discount factor γ	0.8
GAE λ	0.95
Clipping range ϵ	0.2
Learning rate	1×10^{-4}
Entropy coefficient	0.01
Value-function coefficient	0.5
Gradient clip	0.5
Rollout length per environment	8 steps
Rollouts per batch	128
PPO minibatch size	32
PPO epochs per update	4
Maximum policy updates	10,000
Network architecture (CNN-MLP)	
CNN layers	3
CNN channels per layer	16
CNN kernel size	9
MLP units (post-encoder)	32-32-32-16
Total parameter count	9,324
Quantization	INT8
Input processing	
Step time (simulation control cycle)	$25\,\mu s$
Input discretization granularity	$25\,\mu s$
Traffic-history window length	32 steps
Symmetric observations	enabled
Wake-up / sleep delays (simulator)	
Low-power state wake/sleep delay	$100\,\mu s$
Training compute	
Parallel simulation environments	6
Aggregation/update cores	1
Convergence wall-clock time	≈ 4 days
Hardware	CPU only (no GPU)

B SUPERVISED + BAYESIAN BASELINE

Regressor. 3-layer 1D-CNN (kernels $\{9, 3, 8\}$, channels $\{4, 8, 16\}$, dropout 0.2, linear head; $4{,}737$ parameters), fed the same 32-sample ($800\,\mu s$) window as LinkRL and trained to predict the next traffic value on the same six LLM-training traces. Trained with MSE, Adam (lr=5×10^{-4}), batch $2{,}048$, gradient clip 1.0 for up to 200 epochs.

Bayesian decision rule. The link sleeps when $\tilde{t} = \max(0, \hat{t}/\lambda)$ exceeds the $100\,\mu s$ wake/sleep break-even, where $\lambda \geq 1$ is the over- vs. under-estimation penalty ratio. Sweeping $\lambda \in \{1, 1.5, 2, 3, 4\}$ on the validation split, validation NTG peaks at $\lambda=2$ (used in Tables 1, 2).

C POWER PROJECTION METHODOLOGY

We project Power Reduction and Net Throughput Gain (NTG) from per-link simulator outputs using the target deployment fabric of 8 racks with 72 GPUs each (Appendix F).

Per-state power model. DNetSim assigns each link an instantaneous power that takes one of three values depending on state: active and transmitting, active but idle, and low-power. Measured system-wide on the deployment fabric and relative to full load, node power is 0.88 with all links idle and 0.66 with all links in the low-power state. The simulator further models a single distinct low-power state with an activation/deactivation delay of $100\,\mu s$. Per-link energy is the time-weighted sum across these states, computed from per-cycle telemetry. Aggregate network power is the sum across all links.

Baseline power decomposition. The compute-node baseline is decomposed as

$$P_{\text{base}} = P_{\text{compute}} + P_{\text{network}}, \tag{2}$$

where P_{compute} is the compute portion (measured from real hardware) and P_{network} is the aggregate network power above.

Projecting node power under LinkRL. DNetSim reports the fraction of time $\rho \in [0, 1]$ that links spend in the low-power state (distinct from the RL discount factor γ of §3.2). Writing P_{idle} and P_{lp} for node power with all links idle and with all links in the low-power state, the projected node power is

$$P_{\text{LinkRL}} = P_{\text{base}} - \rho\,(P_{\text{idle}} - P_{\text{lp}}), \tag{3}$$

and Power Reduction is the percentage decrease, $(P_{\text{base}} - P_{\text{LinkRL}})/P_{\text{base}}$. For the headline run, $\rho = 44\%$.

Reallocation factor. The $0.9\times$ multiplier maps a percent increase in compute power to the resulting percent throughput gain: a 1% increase yields $\approx 0.9\%$ throughput gain in the target operating regime, derived from silicon measurements. The product is applied to Power Reduction before subtracting JTO. Sensitivity is reported in §D.

Upper-bound estimate. An idealized upper bound on achievable savings follows from the per-state power model and the network's idle fraction. Consider a hypothetical controller that drives every link into the low-power state for every idle period long enough to amortize the $100\,\mu s$ entry and exit latency. Under representative LLM-training conditions, links are idle approximately 80% of the time, of which about 60% of total time is reachable in the low-power state, with the remaining idle time spent in transitions at idle power. Relative to full load, node power of the deployment fabric is 0.88 with all links idle and 0.66 with all links in the low-power state. Average node power before and after this controller is

$$\begin{aligned} P_{\text{baseline}} &= 0.20 + 0.80 \cdot 0.88 \approx 0.905, \\ P_{\text{perfect}} &= 0.20 + 0.20 \cdot 0.88 + 0.60 \cdot 0.66 \approx 0.772, \end{aligned}$$

yielding a maximum node-level Power Reduction of $(0.905 - 0.772)/0.905 \approx 14.7\%$. LinkRL's measured 10.9% (Table 1) captures roughly 74% of this reachable headroom, while incurring a 0.9% job-time overhead that the perfect controller avoids by definition.

D Sensitivity Analysis

We sweep three quantities to verify that headline results are not dependent on a single configured value.

Threshold-baseline idle parameter. The Threshold Policy baseline transitions links to low-power state after a configurable idle period. We evaluated three operating points along the energy–latency frontier. The middle setting was selected as the empirical optimum and is the value reported in Table 1.

Threshold setting	Power Reduction (%)	JTO (%)	NTG (%)
$3000\,\mu s$ (least sleep)	1.27	0.75	0.39
$400\,\mu s$ (reported in Table 1)	3.39	1.74	1.31
$100\,\mu s$ (aggressive sleep)	4.64	2.55	1.63

Power reduction is monotonic: aggressive sleep saves more network power (4.64% vs. 3.39%) but the additional forced wake-ups push JTO up by 0.81pp, leaving net throughput only slightly higher.

We report the middle setting, which gives up a small amount of NTG for a lower and more stable JTO.

Reward weight β. The reward weight β controls the energy-vs-latency trade-off at training time. Larger β values penalize forced wake-ups more heavily, producing more conservative policies. We trained CNN-MLP policies at three values of β and report the best-NTG model at each. The first row matches the headline configuration in Table 1.

β	Power Reduction (%)	JTO (%)	NTG (%)
40 (headline)	10.87	0.90	8.88
80	8.09	0.42	6.86
120	1.98	0.33	1.45

Both Power Reduction and JTO decrease monotonically with β: the policy sleeps less aggressively, saving less power but incurring fewer forced wake-ups. NTG decreases as well, with the headline $\beta = 40$ achieving the highest NTG. NTG stays above the best non-RL baseline (Pattern Predictor, 2.73% in Table 1) at $\beta = 80$ but falls below it at $\beta = 120$, indicating that the trade-off cliff sits between $\beta = 80$ and $\beta = 120$.

Power reallocation factor. While we report NTG using the $0.9\times$ factor derived in §C, we evaluate sensitivity by recomputing NTG for the headline CNN-MLP run (Power Reduction $= 10.87\%$, JTO $= 0.90\%$) under several factor values.

Reallocation factor	NTG (%)
0.85	8.34
0.90	8.88 (reported)
0.95	9.43
1.00	9.97

Since NTG is linear in f, the headline 8.88% varies by ± 0.54pp across $f \in [0.85, 0.95]$, and the qualitative comparison with the baselines in Table 1 is unchanged.

E Sim-to-Real Correlation Supplementary Data

Figure 2 in the main body shows simulator versus deployed-hardware bandwidth across four collective configurations (inter-node, intra-node, and point-to-point patterns), totaling 75 aligned measurement points. The median absolute percentage error across these points is 3.7% overall and 2.7% in the large-message regime (≥ 1 MB) that dominates LLM-training collectives, supporting the $< 5\%$ figure quoted in the main body. Mean error is more sensitive to small-message outliers and to differing bandwidth units across panels (Alg BW vs Bus BW).

The mid-range variance visible in some panels of Figure 2 reflects real-cluster noise (multi-tenant interference, library and OS variance) rather than simulator drift.

F Compute Node Topology

The projection methodology of §5.1 uses the deployment fabric. Each rack comprises 72 GPUs interconnected by 144-port switches, of which 72 ports connect to GPUs and the remainder to the other racks. The full fabric comprises 8 such racks (576 GPUs), and all node-level power figures are measured system-wide over this fabric.

G PPO Update with Shared Parameters Across Agents

LinkRL trains a single set of policy parameters θ shared across all link agents (and a single set of value-network parameters ϕ, also shared). This appendix details how each agent's experience contributes to the shared update.

Per-agent rollout. At each training iteration, every agent $i \in \{1, \ldots, N\}$ in every parallel environment $j \in \{1, \ldots, M\}$ collects a trajectory $\tau^{(i,j)} = \{(s_t^{(i,j)}, a_t^{(i,j)}, r_t^{(i,j)})\}_{t=0}^{T}$ under the current

policy $\pi_{\theta_{\text{old}}}$. Generalized Advantage Estimates $\hat{A}_t^{(i,j)}$ are computed per trajectory (Schulman et al., 2015).

Aggregated buffer. All experience tuples are then aggregated into a single buffer

$$\mathcal{D} \;=\; \bigcup_{i=1}^{N} \bigcup_{j=1}^{M} \bigcup_{t=0}^{T-1} \left\{ \left(s_t^{(i,j)},\, a_t^{(i,j)},\, \hat{A}_t^{(i,j)} \right) \right\}. \tag{4}$$

Minibatches are sampled uniformly from $\mathcal{D}$ regardless of which agent or environment generated the tuple. Each transition therefore contributes equally to the gradient.

Shared-parameter PPO objective. The PPO clipped surrogate (Schulman et al., 2017) is then applied with the shared θ:

$$L^{\text{CLIP}}(\theta) \;=\; \mathbb{E}_{(s,a,\hat{A})\sim\mathcal{D}} \left[\min \left(\rho_\theta(s,a)\, \hat{A},\;\; \text{clip}\big(\rho_\theta(s,a),\, 1{-}\epsilon,\, 1{+}\epsilon\big)\, \hat{A} \right) \right], \tag{5}$$

where $\rho_\theta(s,a) = \pi_\theta(a \mid s)/\pi_{\theta_{\text{old}}}(a \mid s)$ is the importance ratio and ϵ is the clipping range. The full optimization target combines L^{CLIP}, a mean-squared-error value-function loss on the shared critic V_ϕ, and an entropy bonus $S[\pi_\theta]$, weighted by hyperparameters c_v and c_e (Appendix A).

Centralized training, decentralized execution. The aggregated update is the only point at which experiences from multiple agents interact. At inference time, each agent acts independently on its local observations under the shared π_θ, with symmetric features (§3.2) guaranteeing identical actions on the two endpoints of a switch-to-switch link.

H Routing and Flow-Control Interaction

A natural concern is that link-level power-state changes might disrupt routing protocols (e.g., trigger ECMP rehashing) or break flow control. They do not.

Physical-layer operation. LinkRL operates strictly at the physical layer, analogous to the IEEE 802.3az Low-Power Idle (LPI) mechanism. Links retain their logical "UP" status with respect to upper layers (Layer 2 and above) at all times. Routing tables are not updated on a power-state change, and ECMP hashing remains stable: the link continues to advertise the same topology to the control plane. The sub-millisecond timescale of sleep cycles would in any case make control-plane updates infeasible.

Flow-control behavior on wake-up. When traffic is routed to a link in the low-power state, the packet is held in the egress queue while the link finalizes its re-activation procedure. In credit-based or PFC-based fabrics, the resulting absence of credit return naturally exerts backpressure on the sender. Because the wake-up latency ($< 100\,\mu s$) is short relative to standard transport-layer timeouts, no packet is lost and no transport-level timeout is triggered. The fabric perceives the wake-up as transient queueing latency, which standard congestion-control algorithms are already designed to handle.

The only application-visible cost of a misprediction is the wake-up latency itself, precisely what the reward function (Eq. 1) minimizes and what JTO (§5.1) measures.

I Hardware Deployment Footprint

The deployed quantized CNN-MLP unit (INT8, $9{,}324$ parameters) fits within the latency, area, and power envelopes of the target switch hardware. Numbers reflect the planned integration into a major hardware provider's next-generation networking hardware.

Latency budget. A target switch with 144 ports and a $25\,\mu s$ control cycle leaves a per-port budget of $25\,\mu s/144 \approx 175$ ns per inference. The quantized CNN-MLP runs within this budget on dedicated inference logic embedded in the switch, with weights stored in on-chip SRAM.

Chip area. The inference logic occupies a maximum area of approximately $10{,}000\,\mu m^2$ per switch, equivalent to approximately $70{,}000$ standard cells.

Power envelope. The deployed unit operates within approximately 100 mW per switch.

J HYBRID SAFETY GUARDRAIL

Distribution shift at deployment is a standing concern for learning-based controllers. Our generalization results (§5.4, Table 2) show robustness across unseen LLM-training workloads. As a runtime fallback for production anomalies, we additionally integrate a hybrid safety guardrail.

Mechanism. The guardrail is a small deterministic supervisor that tracks the rate of forced wake-ups, i.e., packets arriving at links in the low-power state. If this rate exceeds a configurable threshold over a short sliding window, the supervisor disengages the learned policy and falls back to the threshold-based heuristic baseline of §5. The learned policy continues to run in *shadow mode*, generating predictions without actuating the hardware. The supervisor re-enables the learned policy once its shadow-mode forced-wake rate returns below the threshold.

The guardrail was never triggered across any training or generalization run reported in this paper. Headline results (Tables 1, 2, 3) reflect the learned policy in continuous control of the link.

K TRACE CHARACTERIZATION

Table 5 reports run-length statistics for every trace used in training and evaluation. Statistics are computed on the binary busy/idle timeline of the trace's primary channel.

Table 5: Run-length statistics across training, held-out LLM, and HPC traces, computed on the binary busy/idle timeline of each trace's primary channel. The seven HPC captures (three switch port-counter, four GPU NVLink residency) sit structurally apart from every LLM trace: 5–6% busy fraction (vs. 12–86% across LLM traces), 18.6 ± 3.0 ms average idle (vs. 1.7 ± 1.1 ms across LLM traces), and 2–6 ms maximum busy bursts (vs. up to 137 ms for Llama-3 traces).

Role	Trace	Busy (%)	Avg idle (ms)	p90 idle (ms)	Max idle (ms)	Avg busy (ms)	Max busy (ms)
Training (6 LLM-training workloads)							
Train	`gpt4_MOE`	22.4	2.50	6.1	905	0.72	1.7
Train	`gpt3_175_dense`	33.4	0.61	1.1	334	0.30	0.7
Train	`gpt3_175_MOE`	19.9	0.92	2.9	319	0.23	0.7
Train	`gpt3_40B_dense`	16.5	3.42	3.4	269	0.68	11.5
Train	`llama3_no_sharp`	82.2	1.36	4.2	245	6.27	134.5
Train	`llama3_sharp`	85.1	0.95	2.1	307	5.41	136.7
Held-out LLM (6 unseen workloads)							
Held-out	`llama3_8B_test`	84.3	2.02	3.2	269	10.79	41.6
Held-out	`llama2_70B_test`	86.9	0.68	2.0	165	4.50	180.2
Held-out	`llama3_70B_test`	86.0	0.65	1.2	321	3.99	82.0
Held-out	`gpt3_175_MOE_test`	25.3	0.91	2.8	403	0.31	1.0
Held-out	`gpt3_43B_test`	19.2	3.05	2.6	248	0.73	141.1
Held-out	`gpt3_175B_test`	11.9	3.43	11.4	366	0.46	1.4
HPC (out-of-regime)							
HPC	Aggregate	5.74	18.65	94.7	1186	1.14	4.26

The held-out LLM set spans new model sizes (8B–175B), architectures (Dense vs. MoE), and a new model family (`llama2_70B_test`, with the largest sustained busy burst in the catalog). The seven HPC captures come from two source schemas: switch port-counter dumps (30 s, `0xc6`/`0xc7`/`0xc9`) and GPU NVLink residency dumps (100 s, `0x87`/`0x90`/`0xb7`/`0xbd`). All seven exemplify the regime described in §5.4: long predictable idles where threshold heuristics already capture most of the available savings.