ESUN-BENCH: A WORKLOAD-AWARE BENCHMARKING FRAMEWORK FOR ETHERNET-BASED SCALE-UP AI FABRICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Ethernet fabric metrics do not by themselves determine AI execution performance. The same communication delay can be hidden by concurrent computation or exposed at a synchronization boundary. Existing machine learning network co-simulators provide foundations for coupling communication with application execution. However, Ethernet scale-up benchmarking additionally requires mechanism-specific modeling and observability to interpret these interactions. We present ESUN-Bench, a benchmarking framework that connects packet-level simulation with communication operation timing and an iteration directed acyclic graph. In addition, we formalize the interfaces required for this mapping, including packet-to-operation association, communication completion semantics, and dependency-aware timing propagation. Moreover, we define exposed iteration overhead as the modeled iteration-time difference between matched network configurations, retaining workload dependencies that aggregate packet metrics discard. Evaluation across 102 benchmark cases investigates what communication timing reveals about iteration overhead, which workload conditions shape that timing, and how network-mechanism gains translate into application benefits. The results show that preserving communication semantics and execution dependencies improves iteration-overhead estimation over aggregate network metrics. They further identify message granularity, congestion pressure, and spatial contention as workload factors that shape communication service. Mechanism comparisons demonstrate that activation, communication improvement, and iteration benefit are distinct outcomes: the application value of a network optimization depends on the completion times it changes and the execution dependencies those changes affect. As a lightweight simulation framework, ESUN-Bench fills the gap in existing Ethernet-oriented evaluation methodologies and encourages more researchers to engage in ESUN-related studies. Anonymous Project Page: [LINK]

1 INTRODUCTION

The performance objective of an AI interconnect is to enable dependent computation to proceed. Packet latency and goodput characterize network service, but their application-level significance depends on which communication completes, which computation it releases, and whether another dependency already determines execution time. Distributed training makes this distinction consequential: data parallelism overlaps gradient communication with backpropagation, tensor parallelism introduces communication within model execution, and pipeline parallelism schedules dependent work across devices (Li et al., 2020; Narayanan et al., 2021). Mixture-of-experts (MoE) models add data-dependent exchanges whose volume and destination distribution vary with expert routing (Fedus et al., 2022; Rajbhandari et al., 2022). Thus, evaluating an interconnect requires both a description of the traffic and an account of the execution that waits for it.

Ethernet scale-up networks make this evaluation problem particularly relevant. The Open Compute Project Ethernet for Scale Up Networks (OCP ESUN) specifies Ethernet enhancements for tightly coupled accelerator domains, including a compact additional header and mechanisms for differentiated, resilient communication (Open Compute Project (OCP), 2026). Such mechanisms change network service in different ways: reducing header overhead changes serialization cost,

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.

scheduling changes service order, and congestion or flow control changes when packets can advance. ECN-based congestion control (Alizadeh et al., 2010; Zhu et al., 2015), for example, requires a sender response to marked traffic; marking alone is not an end-to-end control loop. Whether these changes improve AI execution depends on their effect on the communication operations that constrain progress.

The missing information is not recoverable from a packet-latency percentile alone. A P99 statistic discards operation identity, ordering, and dependency placement (Beyer et al., 2016). Even a complete distribution of packet latencies does not specify when all transfers required by a collective have completed (Warraich et al., 2025). Likewise, a communication operation can become slower without delaying an iteration if independent computation still finishes later. Conversely, a delayed operation with little available slack can postpone a dependent compute phase and change the critical path. Dependency-based performance analysis and execution-trace frameworks already recognize the importance of these relationships (Zhu et al., 2020; Sridharan et al., 2026). The problem addressed here is to make these relationships explicit when evaluating Ethernet mechanisms under controlled AI communication workloads.

Modern Machine Learning Network (ML-Network) co-simulation frameworks already provide important foundations for connecting AI execution with network behavior. For instance, the ASTRA-sim series supports distributed AI system exploration and configurable network backends, with later versions introducing graph-based training-loop modeling and more detailed network and device representations (Rashidi et al., 2020; Won et al., 2023; 2026). SimAI integrates training-framework, kernel-computation, and collective-communication models for large-scale LLM performance prediction (Wang et al., 2025). Unlike these frameworks, this paper addresses the evaluation requirements, which concern mechanism-specific models and observables required to interpret Ethernet scaling behavior within such execution contexts.

For Ethernet scale-up evaluation, this coupling must be accompanied by explicit models of the mechanisms under investigation (Ultra Ethernet Consortium, 2025; UALink Consortium, 2025). Compact headers change serialization overhead; hop-by-hop flow control changes transmission eligibility, and link-level recovery changes when the payload required by a communication operation becomes available. Their application-level effects depend on whether the affected transfers determine operation completion and whether that completion constrains subsequent computation. Supporting an AI execution graph and a network backend does not, by itself, establish that these mechanism-specific interactions are represented and exposed for attribution (Wang et al., 2025). An interpretable benchmark must retain the associations needed to follow network events through transfer and communication completion to execution dependencies.

Therefore, we propose ESUN-Bench around a single evaluation question: **when, and through which execution dependencies, does a change in Ethernet fabric behavior affect AI iteration time?** The framework combines controlled workload scenarios, deterministic packet-level simulation, and iteration-DAG analysis to examine which network effects are absorbed by computation-communication overlap and which extend iteration time. It separates three kinds of evidence: first, network measurements describe service; then, mechanism records explain how that service arises; and finally, execution dependencies determine whether the resulting delay is exposed to the application. This Ethernet-oriented evaluation methodology complements the broader system-design capabilities of existing ML-Network co-simulation frameworks. Controlled comparisons are required to establish whether a mechanism provides an application-level benefit.

The main contributions of ESUN-Bench are summarized as follows:

- We develop a workload-aware framework ESUN-Bench that integrates deterministic packet-level simulation, communication-completion reconstruction, and iteration-DAG analysis.
- We formalize exposed iteration overhead (EIO) through paired network configurations that share the workload, compute durations, and execution scheduling policy.
- We construct a 102-case suite spanning controlled network workloads, collective communication, and AI execution templates.
- We examine how workload conditions shape communication timing and how mechanism-induced changes propagate to AI execution.

2 Related Work

2.1 Network Simulation

General-purpose network tools provide complementary foundations for interconnect evaluation. NS-3 (Riley & Henderson, 2010) supports discrete-event packet-level simulation, whereas Mininet (Kaur et al., 2014) enables network emulation with real protocol stacks through operating-system virtualization. Although neither defines an AI execution model as its primary abstraction, their capabilities can be integrated into workload-aware evaluation systems. ML-Network co-simulators explicitly address this integration. ASTRA-sim (Rashidi et al., 2020) supports distributed deep-learning system exploration, and ASTRA-sim 2.0 (Won et al., 2023) introduces graph-based training-loop modeling together with hierarchical network and memory-system extensions. ASTRA-sim 3.0 Won et al. (2026) further develops detailed GPU execution and infrastructure modeling. SimAI integrates training frameworks, kernel computation, and collective communication to predict large-scale LLM training performance (Wang et al., 2025). Daydream (Zhu et al., 2020) uses dependency graphs and execution traces to estimate the effect of training optimizations.

2.2 AI Scale-Up Fabric

AI scale-up fabrics support tightly coupled accelerator communication whose performance depends on both network service and execution dependencies (Saber & Jiang, 2025). The Ultra Ethernet Consortium (UEC) officially released its full Version 1.0 specification in 2025 (Ultra Ethernet Consortium, 2025), its core goal is to address the deficiencies of conventional RoCEv2 (Yan et al., 2023) deployed in large-scale AI clusters and improve transmission efficiency for artificial intelligence and high-performance computing workloads. In parallel, OCP ESUN defines Ethernet scale-up requirements in 2026 that motivate the mechanism scope of this work (Open Compute Project (OCP), 2026). These specifications are tailored exclusively to the scaled expansion scenarios of inter-rack networks within a single cluster, yet they pay no particular attention to network runtime behavior and the execution logic of artificial intelligence training iterations.

Congestion-control mechanisms such as DCTCP (Alizadeh et al., 2010) and DCQCN (Zhu et al., 2015) demonstrate how network feedback and endpoint response jointly determine communication service. While, distributed AI systems motivate the communication structures used in such evaluations. For example, Horovod (Sergeev & Del Balso, 2018) and PyTorch Distributed (Li et al., 2020) emphasize collective communication in data-parallel training. Furthermore, Megatron-LM (Shoeybi et al., 2019) and GPipe (Huang et al., 2019) expose dependencies introduced by model and pipeline parallelism. Also, ZeRO (Rajbhandari et al., 2020) changes communication through model-state partitioning. Then, GShard (Lepikhin et al., 2020), Switch Transformers (Fedus et al., 2022), and DeepSpeed-MoE (Rajbhandari et al., 2022) further motivate expert-routing workloads. These systems provide workload semantics relevant to interconnect evaluation, without implying that a synthetic traffic template reproduces their complete implementations. Accordingly, several existing studies have investigated how to correlate these workloads with the operational characteristics of switching architectures. TACCL (Shah et al., 2023) synthesizes topology-specific collective algorithms, while standardized collective representations expose transfer dependencies to downstream tools (Yoo et al., 2025). Chakra (Sridharan et al., 2023) represents operations and execution dependencies in graph-based traces, and CCL-Bench (Ding et al., 2026) packages traces and workload metadata to support explanations beyond aggregate infrastructure scores.

Collectively, existing works merely advocate preserving intermediate semantics spanning packet transmission and application execution, yet lack a dedicated benchmark to map low-level network events to communication completion and execution dependencies for Ethernet scale-up assessment. Different from previous frameworks, ESUN-Bench is a lightweight complementary benchmark for controlled mechanism evaluation and cross-layer interpretation. It focuses on a more specific evaluation objective: explaining how Ethernet scaling mechanisms affect communication completion latency and whether those effects propagate to AI iteration time. This approach distinguishes mechanism activation, communication-level improvement, and application-level benefit, providing a basis for interpreting when and why an Ethernet mechanism improves AI execution.

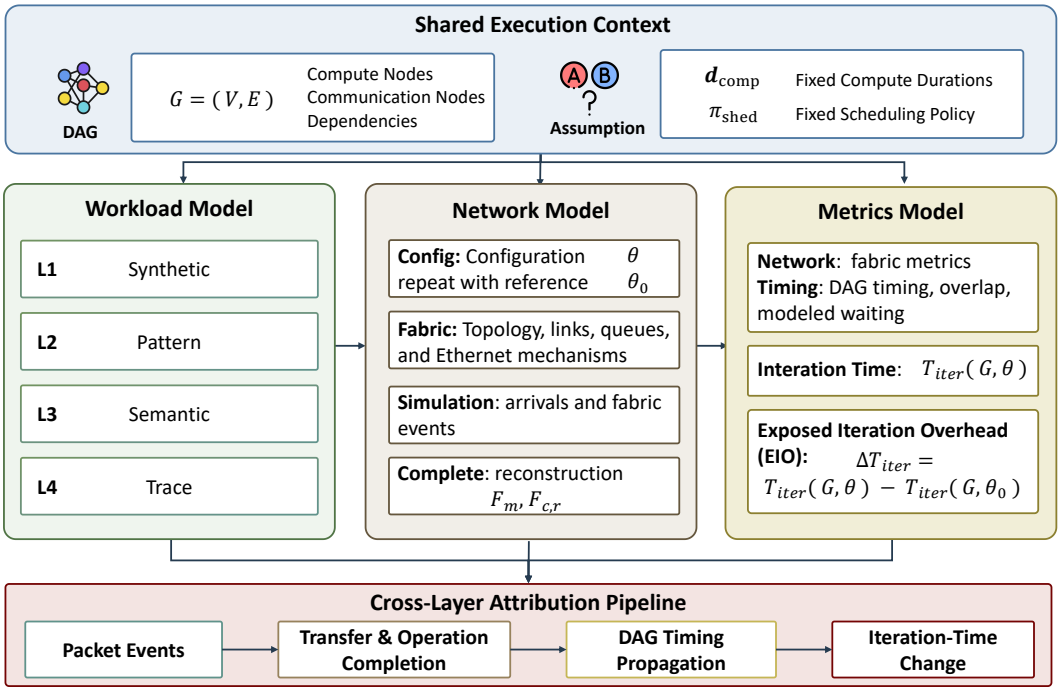

Figure 1: **ESUN-Bench architecture.** ESUN-Bench integrates a hierarchical workload model, an explicit Ethernet network model, and a cross-layer metrics model. The Workload Model generates traffic and retains communication semantics and operation–transfer–packet associations. The Network Model records packet delivery and mechanism behavior under configuration θ, then reconstructs transfer and communication completion. The Metrics Model combines communication timing with $(G, \mathbf{d}_{\text{comp}}, \pi_{\text{sched}})$ to determine iteration performance and interpret exposed delay. In addition, we abbreviate the iteration-time expression $T_{\text{iter}}(G, \theta; \mathbf{d}_{\text{comp}}, \pi_{\text{sched}})$ as $T_{\text{iter}}(G, \theta)$ when the shared execution context is implicit.

3 Methodology

3.1 Problem Formulation

Let an AI iteration be represented by a directed acyclic graph $G = (V, E)$, with computation nodes V_{comp} and communication nodes V_{comm}. Each computation node has a specified duration and a device assignment. Each communication node has participants, a primitive, a payload description, and a transfer schedule. Edges encode data dependencies and synchronization; resource-order constraints are additionally required to obtain an executable schedule. A network configuration θ specifies the topology, links, buffering, scheduling, routing, and enabled control mechanisms.

We denote the indexed compute-duration vector and the AI execution scheduling policy by $\mathbf{d}_{\text{comp}} = (d_v)_{v \in V_{\text{comp}}}$ and π_{sched}, respectively. The latter governs the selection and resource ordering of ready AI operations; it is distinct from network queue-scheduling policies, such as FIFO, SP, or WRR, which belong to θ. Across a paired comparison, the shared execution context satisfies $\mathbf{d}_{\text{comp}}^{(0)} = \mathbf{d}_{\text{comp}}$ and $\pi_{\text{sched}}^{(0)} = \pi_{\text{sched}}$. Given this context, the evaluation objective is to determine how replacing a reference network configuration θ_0 with θ changes iteration time, and to explain that change through the communication operations affected by network service. We write the modeled iteration time as $T_{\text{iter}}(G, \theta; \mathbf{d}_{\text{comp}}, \pi_{\text{sched}})$ and define the exposed iteration overhead (EIO) metric as shown in Eq. 1.

$$\Delta T_{\text{iter}} = T_{\text{iter}}(G, \theta; \mathbf{d}_{\text{comp}}, \pi_{\text{sched}}) - T_{\text{iter}}(G, \theta_0; \mathbf{d}_{\text{comp}}, \pi_{\text{sched}}). \tag{1}$$

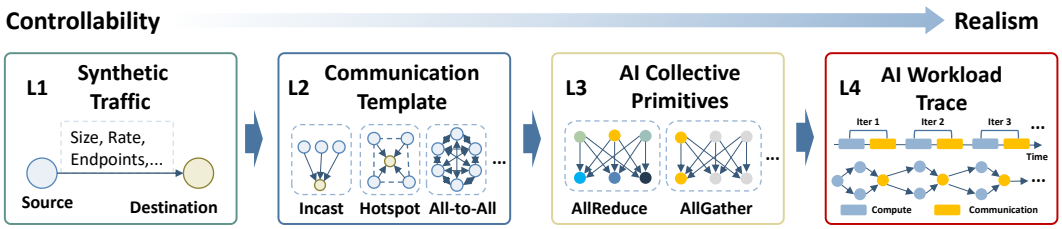

Figure 2: **Hierarchical workload abstraction.** ESUN-Bench progressively trades controllability for realism and semantic richness across four levels, from synthetic traffic to trace-based AI workloads.

Holding the execution policy fixed does not require identical operation start times or realized resource order across configurations. These follow the modeled dependencies, communication durations, and scheduling rules. If projection instead preserves a trace-defined resource order, that order is an additional declared constraint.

3.2 Framework

ESUN-Bench therefore adopts a cross-layer, workload-aware methodology that connects AI communication semantics to packet-level network behavior and ultimately to AI execution performance. As shown in Figure 1, the workload graph G, compute durations, and scheduling policy are evaluated under a reference network configuration θ_0 and an evaluated configuration θ. The Workload Model represents $L1 - L4$ inputs as communication operations and transfer dependencies, then performs packetization with identity mapping. The Network Model simulates the resulting traffic under configuration θ, records packet arrivals and fabric events, and reconstructs communication completion. The Metrics Model combines these records with the shared execution context to report network diagnostics and application-visible timing effects.

3.3 Workload Model

As illustrated in Figure 2, ESUN-Bench provides four abstraction levels. **L1** generates configurable synthetic point-to-point traffic for isolating basic network behavior. **L2** introduces controllable communication templates, including incast, fanout, hotspot, all-to-all, and mixed traffic, to reproduce spatial contention and temporal burstiness. **L3** captures AI collective semantics through AllReduce, AllGather, and ReduceScatter. **L4** reconstructs workloads from execution traces as sequences or DAGs of computation and communication phases, preserving synchronization and execution dependencies. Graph-based execution representations and collective schedule representations provide relevant precedents for these interfaces (Sridharan et al., 2023; Yoo et al., 2025).

We describe a communication operation $c = (\mathrm{id}_c, \mathcal{R}_c, \mathrm{op}_c, \mathbf{B}_c, \mathcal{H}_c, \kappa_c)$, where $\mathcal{R}_c$ is the participant set, op_c is the communication primitive, $\mathbf{B}_c$ records transfer sizes, $\mathcal{H}_c$ is the transfer dependency graph, and κ_c is the traffic class. A matrix of transfer sizes is useful for all-to-all and MoE traffic because equal total volume does not imply equal load at every destination. The release conditions of operation c come from its predecessors in G.

For a transfer m of payload size B_m and packet payload capacity M, the packetizer generates $n_m = \lceil B_m/M \rceil$ and $\sum_{p\in\mathcal{P}_m} b_p = B_m$. Here, $\mathcal{P}_m$ contains logical payload packets, so retransmissions do not increase the original payload count. The minimum association record is $(\mathrm{id}_c, \mathrm{id}_m, \mathrm{id}_p)$, together with source, destination, traffic class, and timestamps. When chunks or collective steps are modeled, their identifiers and dependencies are also retained. These associations allow the analyst to determine which operation owns a delayed transfer and which downstream computation consumes its output.

3.4 Network Model

For a packet p, let s_p denote the timestamp from which latency is measured and a_p its successful delivery time, giving $L_p = a_p - s_p$. A transfer m completes when all required payload packets in $\mathcal{P}_m$

have arrived. For a collective operation c, the transfer graph $\mathcal{H}_c$ specifies step dependencies, and $\mathcal{T}_{c,r}$ contains the terminal transfers required to produce participant r's local output. Its communication completion time is detailed in Eq. 2.

$$F_{c,r} = \max_{m \in \mathcal{T}_{c,r}} F_m = \max_{m \in \mathcal{T}_{c,r}} \max_{p \in \mathcal{P}_m} a_p, \tag{2}$$

where F_m denotes transfer completion and a_p refers to successful delivery after any required retransmissions. Subsequent local computation can proceed once this communication dependency and its other prerequisites are satisfied. When the workload requires a global barrier, completion is determined by $\max_{r \in \mathcal{R}_c} F_{c,r}$. Otherwise, participants retain their local completion times. Topology-specific collective algorithms determine the transfer dependencies and therefore affect completion timing (Shah et al., 2023). The Network Model represents Ethernet mechanisms through their effects on packet service and delivery. Header accounting determines serialization cost, while queue scheduling determines service order. PFC and credit-based flow control govern transmission eligibility through pause states and credit availability, respectively. ECN marking records congestion feedback; its effect on subsequent traffic depends on the configured sender response. Link-level recovery accounts for retransmission following modeled transmission errors. The simulator records these mechanism events alongside packet timing and operation identifiers, allowing changes in communication completion to be associated with the underlying network behavior. Further details on mechanism implementations, parameter settings, and modeling assumptions are provided in Appendix A.2.

3.5 Metrics Model

The Metrics Model maps communication timing to AI iteration performance. Let $G^+(G, \theta)$ be the workload DAG augmented with the resource-order constraints realized under the shared execution policy π_{sched}. Let $\mathcal{P}(G^+)$ denote its source-to-sink paths and $d_v(G, \theta)$ the duration of node v: compute-node durations come from $\mathbf{d}_{\text{comp}}$, while communication-node durations are reconstructed from release and completion events in the Network Model. The iteration time is the longest path duration shown in Eq. 3.

$$T_{\text{iter}}(G, \theta) = \max_{\rho \in \mathcal{P}(G^+(G,\theta))} \sum_{v \in \rho} d_v(G, \theta), \tag{3}$$

where, ρ is a dependency path and v indexes its nodes. Under the modeled schedule, communication delays extend iteration time only when they increase the longest path duration; delays absorbed by concurrent computation do not change T_{iter}. To connect the Network Model's communication timing to AI execution, we hold G, $\mathbf{d}_{\text{comp}}$, and π_{sched} fixed and define the exposed iteration overhead (EIO) metric as ΔT_{iter}. The difference measures application-visible iteration overhead in the same time unit as T_{iter}: positive values indicate slower execution, negative values indicate improvement, and zero indicates no iteration-time change. For comparisons across workload sizes, normalized EIO (nEIO) is defined in Eq. 4 and expressed as a percentage of the positive reference iteration time.

$$\Delta T^{nor}_{\text{iter}} = 100 \times \frac{\Delta T_{\text{iter}}}{T_{\text{iter}}(G, \theta_0)}. \tag{4}$$

4 Experiments

Our evaluation addresses three questions: **(Q1)** what does communication timing expose at the AI iteration level? (Sec. 4.2) **(Q2)** which workload and network conditions produce the relevant timing changes? (Sec. 4.3) **(Q3)** How should network mechanism gains be interpreted at the AI iteration level? (Sec. 4.4)

4.1 Experimental Setup

The primary benchmark suite contains 102 test cases, and Table 1 summarizes the major scenario groups, case counts, coverage ranges, and evaluation targets. All experiments use the deterministic

Table 1: **Coverage of the 102-case ESUN-Bench suite.** Each group specifies the workload or configuration variables examined and the primary measurements used for evaluation.

Group	Cases	Workload or Variable	Primary Observable
Protocol overhead	28	Payload size: $8B - 8MB$	Header efficiency, transfer service
Congestion	23	Incast, hotspot, offered load, fan-in	Queueing, ECN/PFC/credit behavior
Collectives	4	AllReduce, AllGather, ReduceScatter	Transfer schedule, collective completion
AI traffic patterns	15	All-to-all, MoE skew, mixed traffic	Contention, completion imbalance
Topology	10	Ring, mesh, Clos; hop and scale variation	Path, queue behavior
Scheduling	9	Buffer and FIFO/SP/WRR combinations	Class-based waiting time
Reliability	4	Error stress	Delivery, recovery timing
AI trace templates	9	Three execution-template families	Dependency exposure, iteration timing
Total	102		

packet-level simulator described in Section 3, executed with Python 3.13.0, Pytorch 2.12.0 and Nvidia GeForce RTX 3090. More detailed experimental setups are provided in the Appendix A.3.

4.2 What Does Communication Timing Expose at the AI Iteration Level?

The results of Table 2 show that communication timing exposes the application-visible iteration overhead that packet statistics alone do not describe. The dense template reports no iteration extension despite nonzero packet latency, whereas the MoE and inference templates report 19.601 μs and 0.696 μs of exposed delay. Thus, the existing results answer Q1 for the evaluated templates: network delay and iteration overhead are distinct quantities, and their relationship depends on the modeled execution context.

Table 2: **Packet latency and iteration-level effects across three AI workload templates.** Packet P99 and exposed iteration delay are reported in μs. Iteration slowdown is the iteration-time difference relative to the reference configuration, expressed as a percentage of the reference iteration time.

Template	Packet P99 ↓	Exposed Delay ↓	Iteration Slowdown ↓
Dense Transformer	3.835	0	0
MoE Transformer	19.393	19.601	0.049
Inference Prefill/Decode	1.412	0.696	0.002

We evaluate 24 paired comparisons across six workloads and four ESUN variants: ESUN-Header, ESUN-FC, ESUN-Congestion, and ESUN-Full. Each sample compares a completed variant run with the matching Ethernet baseline under the same workload. For each run, iteration overhead is computed as $O = 1000(T_{\text{iter}} - T_{\text{comp}})$ in microseconds, using the recorded iteration time and configured compute time in milliseconds; the evaluation target is the paired difference $\Delta O = O_{\text{variant}} - O_{\text{Ethernet}}$. The EIO predictor computes the corresponding difference in exposed communication time from packet-to-operation associations, communication timings reconstructed from packet events and phase records, execution-DAG dependencies, overlap parameters, and the run's network configuration. MAE (Hogg et al., 2013) is evaluated using leave-one-workload-out cross-validation: each fold holds out all four variants of one workload and uses the remaining 20 samples for calibration. For calibrated metrics, a scalar linear model $\widehat{\Delta O} = ax + b$, where x is the predictor's paired difference, is fitted by least squares on the training samples and applied to the held-out workload. Direct metrics are evaluated in microseconds without calibration. The results are shown in Table 3, which distinguishes configuration ranking from overhead estimation. Packet P99 achieves perfect SRCC but substantially higher MAE than EIO, showing that correct ordering

does not ensure accurate prediction of iteration-level effects. Aggregating matched communication-duration changes improves estimation, but treats delays additively without accounting for overlap or dependency placement. EIO further reduces MAE from 0.264 to 0.099 μs by incorporating execution structure. This result supports the importance of modeling how communication changes are exposed to execution, beyond measuring their aggregate duration.

Table 3: **Correlation and prediction accuracy of iteration-overhead predictors.** ΔL_{pkt} and ΔL_{99} denote changes in mean and P99 packet latency; ΔT_{tx} and ΔT_{op} denote changes in mean transfer and operation duration; ΔG denotes the decrease in goodput; and $\sum \Delta T_{\text{comm}}$ sums duration changes across matched communication operations. oPLCC and oSRCC measure Pearson and Spearman correlations with observed overhead (Hogg et al., 2013). MAE (μs) is evaluated using leave-one-workload-out cross-validation. Average reports the mean rank across the three metrics, with higher values indicating better overall performance.

Metric	Information Retained	oPLCC ↑	oSRCC ↑	MAE ↓	Average ↑
ΔL_{pkt}	Packet timing	0.487	0.829	1.365	2.333
ΔL_{99}	Packet tail	0.860	1.000	1.221	5.167
ΔT_{tx}	Transfer completion	0.525	0.829	1.390	2.333
ΔT_{op}	Communication semantics	0.661	1.000	1.317	4.500
ΔG	Delivered payload rate	0.983	0.829	3.793	3.000
$\sum \Delta T_{\text{comm}}$	Matched operations	0.988	0.829	0.264	5.167
ΔT_{iter}	Mapping & execution dependencies	0.988	0.829	0.099	5.500

4.3 Which Workload and Network Conditions Produce the Timing Changes?

This section connects workload conditions to mechanism effects and then asks whether those effects reach the AI iteration. The relevant comparison changes a declared mechanism configuration from θ_0 to θ under the same workload and execution assumptions, and evaluates ΔT_{iter} alongside network and operation timing.

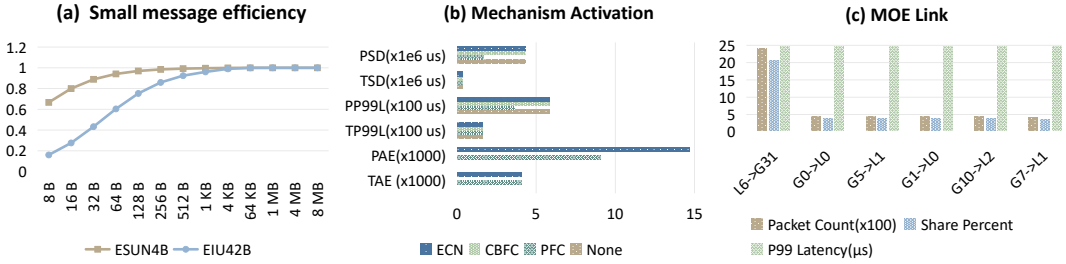

Figure 3: **Effects of message granularity, congestion, and traffic placement on network behavior.** (a) Payload efficiency under simplified header-accounting models: ESUN4B and EIU42B denote 4-byte ESUN and 42-byte Ethernet/IP/UDP overhead, respectively. (b) Activation-event counts (PAE/TAE), packet P99 latency (PP99L/TP99L), and scheduler delay (PSD/TSD) under persistent/transient congestion. Scaling factors are indicated in the plot. (c) Bottleneck-observation counts and shares across the highest-ranked links in the MoE workload, shown alongside the workload-level packet P99 latency. G and L denote GPU and leaf-switch identifiers, respectively.

Figure 3 examines how message granularity, congestion pressure, and spatial contention shape communication service. In subfigure (a), an 8-byte payload achieves an efficiency of 0.667 under the declared header-accounting model, whereas efficiency approaches unity for multi-megabyte transfers. This behavior indicates that the relative cost of the modeled overhead depends on message granularity, motivating workload-specific evaluation of header compaction. The resulting serialization changes affect transfer timing, but their iteration-level significance depends on which communication completions constrain execution. The subfigure (b) compares transient and persistent congestion using mechanism-event counts, packet P99, and scheduler waiting time. The scenarios produce up to 14,660 ECN marks and 9,071 PFC events, identifying conditions under which the

Table 4: **Network, communication-operation, and iteration-level effects of mechanism changes.** FL denotes flow labels; Eff denotes payload efficiency; PFC, ECN, and CBFC report the corresponding mechanism-event counts; LLR reports link-level retries; ΔL^p_{99} (%) and ΔT^p_{op} (%) denote the percentage change in P99 packet latency and communication-operation duration. For clarity, selected zero-valued event counts are omitted from the Activation Evidence column.

Case / Mechanism	$\Delta L^p_{99} \downarrow$	$\Delta T^p_{op} \downarrow$	$\Delta T_{\mathrm{iter}} \downarrow$	$\Delta T^{nor}_{\mathrm{iter}} \downarrow$	**Activation Evidence**
Header / 8B	-0.40	-0.40	0	0	Eff: $0.16 \rightarrow 0.67$, FL=1
Header / 1MB	0	0	0	0	Eff: $1.00 \rightarrow 1.00$, FL=1
PFC / persistent	-38.40	-37.80	-227.49	-0.56	PFC=9071
ECN / persistent	0	0	0	0	ECN=14660
CBFC / persistent	0	0	0	0	–
Header / AllReduce	-50.10	-41.10	-4.30	-0.01	FL=159
Full-vs-Header / MoE	0	0	0	0	FL=192
Full-vs-Header / All-to-All	0	0	0	0	FL=188

corresponding mechanisms become active. These counts describe a different aspect of network behavior from latency and waiting time: activation records show that control conditions are reached, while timing measurements reveal the resulting communication service. Finally, subfigure (c) adds a spatial perspective. Bottleneck observations in the MoE workload concentrate on LEAF6→GPU31, while the case reports a packet P99 of 24.55 μs. The concentration identifies where contention occurs, information that the aggregate percentile alone cannot provide. It also motivates examining how traffic is distributed across destinations and links, beyond its total volume. Together, these results answer Q2 by locating communication-service differences in message overhead, congestion conditions, and traffic placement.

4.4 How should network mechanism gains be interpreted at the AI iteration level?

Table 4 shows that network-mechanism gains must be assessed through their effects on communication completion and execution dependencies. Across these workloads, the larger percentage improvement in packet latency therefore does not imply the larger application benefit. The baseline timing and execution context also determine the outcome. For 8-byte messages, packet and operation timing improve by 0.40, with no iteration-time change at the reported precision. ECN records 14,660 marks without a timing improvement, showing that marking activity alone does not establish a performance benefit. Full-vs-Header comparisons likewise show no additional timing benefit for the evaluated MoE and All-to-All workloads. Together, these results answer Q3: EIO and its normalized form quantify the modeled application benefit, while operation timing and mechanism records identify the intermediate changes through which that benefit must arise.

5 Conclusion

This paper presents ESUN-Bench, a workload-aware framework for evaluating Ethernet scale-up fabrics through their effects on AI iteration performance. By connecting packet events, communication completion, and execution dependencies, the framework quantifies modeled iteration-time changes using exposed iteration overhead (EIO). Experiments show that preserving communication semantics and execution structure improves overhead estimation over aggregate network metrics, while workload conditions shape mechanism activity and communication timing. Mechanism comparisons further demonstrate that application benefits depend on how communication improvements propagate through execution dependencies. These findings establish the importance of interpreting network mechanisms across packet, operation, and iteration levels. Future work will extend validation with hardware measurements and incorporate more measured AI execution traces.

AI USE STATEMENT

In this work, we use artificial intelligence tools for code writing and debugging, manuscript polishing to improve readability, screening of relevant references, standardization of reference formats, as well as refinement of paper titles and keywords. We do not use generative artificial intelligence tools for theoretical model derivation, mathematical formulation, and overall research scheme design. The core research content of this paper involves the design of AI-related data simulation models. All experimental data are generated exclusively by the simulation models proposed in this work without the use of generative AI tools. The remaining disclosure items are not applicable to this work. We review all AI-assisted work. Specifically, all AI-polished manuscripts are manually revised and cross-checked by all authors to ensure technical accuracy and logical rigor. All literature summarized by AI is verified against original publications to prevent misinterpretation and omission of key research findings. All experimental conclusions and framework claims presented in this paper are independently validated by the authors based on the actual benchmark results of the ESUN-Bench framework. We take full responsibility for the final content of this work, including all texts, research claims, and experimental outcomes produced with the assistance of generative artificial intelligence.

ETHICS STATEMENT

This work does not involve human subjects, human data collection, animal experiments, sensitive personal information, or field studies requiring ethical approval. All authors of this study have strictly adhered to the ICLR Code of Ethics throughout the research and writing process. No real user data, confidential industrial datasets, or regulated biological materials are utilized. Therefore, no additional ethical review procedures are required for this research.

REPRODUCIBILITY STATEMENT

This work guarantees the reproducibility of all proposed research results and experimental findings. All experimental settings, implementation details, parameter configurations, and evaluation metrics required for reproducing the benchmark experiments and framework validation are fully elaborated in the main text, appendix, and supplementary materials of this paper. Readers can strictly follow the documented experimental procedures and step-by-step guidelines presented in the above contents to replicate all performance tests, and functional verifications of the proposed ESUN-Bench framework. Furthermore, the complete source code of the proposed method and the corresponding experimental scripts will be fully released and made publicly available upon the official acceptance of this paper, to facilitate independent verification, repeated validation, and further extension of this work by the academic community.

REFERENCES

Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 Conference*, pp. 63–74, 2010. doi: 10.1145/1851182.1851192.

Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. *Site reliability engineering: how Google runs production systems*. " O'Reilly Media, Inc.", 2016.

Eric Ding, Byungsoo Oh, Bhaskar Kataria, Kaiwen Guo, Jelena Gvero, Abhishek Vijaya Kumar, Arjun Devraj, Lindsey Bowen, Atharv Sonwane, Emaad Manzoor, et al. Ccl-bench 1.0: A trace-based benchmark for llm infrastructure. *arXiv preprint arXiv:2605.06544*, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Robert V Hogg, Joseph W McKean, Allen T Craig, et al. *Introduction to mathematical statistics*. Pearson Education India, 2013.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Karamjeet Kaur, Japinder Singh, and Navtej Singh Ghumman. Mininet as software defined networking testing platform. In *International conference on communication, computing & systems (ICCCS)*, pp. 139–42, 2014.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–15, 2021.

Open Compute Project (OCP). OCP ESUN – Network Operator Requirements – Base Specification V1.0, 2026. URL https://www.opencompute.org/documents/ocp-esun-network-operator-requirements-base-specification-rev-1-0-final-pdf.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

Saeed Rashidi, Srinivas Sridharan, Sudarshan Srinivasan, and Tushar Krishna. Astra-sim: Enabling sw/hw co-design exploration for distributed dl training platforms. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 81–92. IEEE, 2020.

George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pp. 15–34. Springer, 2010.

Md Ghulam Saber and Zhiping Jiang. Physical layer standardization for ai data centers: Challenges, progress, and perspectives. *IEEE Network*, 40(2):147–155, 2025.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

Aashaka Shah, Vijay Chidambaram, Meghan Cowan, Saeed Maleki, Madan Musuvathi, Todd Mytkowicz, Jacob Nelson, Olli Saarikivi, and Rachee Singh. TACCL: Guiding collective algorithm synthesis using communication sketches. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 593–612, 2023.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Srinivas Sridharan, Taekyung Heo, Louis Feng, Zhaodong Wang, Matt Bergeron, Wenyin Fu, Shengbao Zheng, Brian Coutinho, Saeed Rashidi, Changhai Man, et al. Chakra: Advancing performance benchmarking and co-design using standardized execution traces. *arXiv preprint arXiv:2305.14516*, 2023.

Srinivas Sridharan, Andy Balogh, Bradford M Beckmann, Brian Coutinho, Louis Feng, Sheng Fu, Sanshan Gao, Mehryar Garakani, Taekyung Heo, David Kanter, et al. Mlcommons chakra: Advancing performance benchmarking and co-design using standardized execution traces. *Proceedings of Machine Learning and Systems*, 8:451–468, 2026.

UALink Consortium. UALink 200G 1.0 Specification, 2025. URL https://ualinkconsortium.org/wp-content/uploads/2025/04/UALink-1.0-Specification-PR_FINAL.pdf.

Ultra Ethernet Consortium. Ultra Ethernet Specification v1.0, 2025. URL https://ultraethernet.org/wp-content/uploads/sites/20/2025/06/UE-Specification-6.11.25.pdf.

Xizheng Wang, Qingxu Li, Yichi Xu, Gang Lu, Dan Li, Li Chen, Heyang Zhou, Linkang Zheng, Sen Zhang, Yikai Zhu, et al. SimAI: Unifying architecture design and performance tuning for large-scale large language model training with scalability and precision. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pp. 541–558, 2025.

Ertza Warraich, Ali Imran, Annus Zulfiqar, Shay Vargaftik, Sonia Fahmy, and Muhammad Shahbaz. Optinic: A resilient and tail-optimal rdma nic for distributed ml workloads. *arXiv preprint arXiv:2512.22743*, 2025.

William Won, Taekyung Heo, Saeed Rashidi, Srinivas Sridharan, Sudarshan Srinivasan, and Tushar Krishna. Astra-sim2. 0: Modeling hierarchical networks and disaggregated systems for large-model training at scale. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 283–294. IEEE, 2023.

William Won, Jinsun Yoo, Tuan Ta, Moumita Dey, Andy Balogh, Pradosh Datta, Furkan Eris, Conor Green, Winston Liu, Changhai Man, et al. Astra-sim 3.0: Next-level distributed machine learning simulations via high-fidelity gpu and infrastructure modeling. *arXiv preprint arXiv:2606.10440*, 2026.

Dingyu Yan, Yaping Liu, Shuo Zhang, Zhikai Yang, and Yingzhen Wang. A survey of rocev2 congestion control. In *International Conference on Information Science, Communication and Computing*, pp. 42–56. Springer, 2023.

Jinsun Yoo, William Won, Meghan Cowan, Nan Jiang, Benjamin Klenk, Srinivas Sridharan, and Tushar Krishna. Toward a standardized representation for deep learning collective algorithms. *IEEE Micro*, 45(2):46–55, 2025.

Hongyu Zhu, Amar Phanishayee, and Gennady Pekhimenko. Daydream: Accurately estimating the efficacy of optimizations for DNN training. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pp. 337–352, 2020.

Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. pp. 523–536, 2015. doi: 10.1145/2785956.2787484.

A Appendix

A.1 Simulator Credibility Validation

We adopt ns-3 version 3.36 with core modules including point-to-point, internet (TCP stack), traffic-control, applications, and flow-monitor; the network follows a spine–leaf topology consisting of 2 spine switches, 8 leaf switches and 4 hosts connected to each leaf. Each link operates at 100 Gbps with a propagation delay of 0.5 μs, where switches deploy PFIFO queues with a depth of 1024 packets and weighted fair queuing scheduling, an ECN marking threshold of 200 packets, and three independent priority buffers per port for PFC. The default packet size is set to 1500 B, and we generate TCP bulk flows as well as AllReduce collective traffic, with incast workloads used for persistent congestion test cases. Each simulation runs for 10 seconds in total, discarding the initial

Table 5: **Cross-validation of ESUN-Bench against ns-3.** This table compares ESUN-Bench with ns-3 at the packet and communication-operation levels, where Relative Error $= |\text{ESUN-Bench} - \text{ns-3}|/\text{ns-3} \times 100\%$.

Scenario	Metric	Relative Error (%) ↓
Header / 8B	P99 latency (μs)	1.01
Header / 1MB	P99 latency (μs)	0.33
PFC / persistent	P99 latency (μs)	0.75
ECN / persistent	P99 latency (μs)	0.03
CBFC / persistent	P99 latency (μs)	0.03
Header / AllReduce	P99 latency (μs)	0.65
Full-vs-Header / MoE	Op. completion (μs)	0.70
Full-vs-Header / All-to-All	Op. completion (μs)	0.82

1-second warm-up phase, and all performance metrics are collected within the steady-state window covering the remaining 9 seconds.

Table 5 shows that, across the eight reported comparisons, relative errors range from 0.03% to 1.01%. The packet-latency comparisons assess forwarding and queueing behavior under header and persistent-congestion configurations, including the reported PFC and ECN settings. The MoE and All-to-All comparisons additionally assess communication-operation completion timing. These results support timing agreement for the evaluated configurations.

Table 6: **Overview of six workload generators in ESUN-Bench.** Baseline-control instantiation used for the six workloads: $num_gpus = 32$, $message_size = 64$, $packet_size = 1$, $offered_load = 0.85$, and $compute_time_ms = 40$.

Workload	Source	Nodes	Comm. type & volume	Compute, DAG, overlap	Link to test suite
Point-to-point	Artificial pattern template	2	Single unicast flow, total 64 KiB payload	40 ms compute; DAG: forward → comm → backward → optimizer; overlap=0.35	Small-message, latency, multihop, LLR, baseline control cases
Incast	Artificial pattern template	17	16-to-1 fan-in incast, 16 flows, total 1.000 MiB	40 ms compute; same sequential DAG; overlap=0.0	Incast stress, baseline control
AllReduce	Artificial AI collective template	32	Log-based all-reduce, 160 flows / 5 phases, total 5.000 MiB	40 ms compute; comm after backward; overlap=0.15	Collective performance, baseline control
All-to-All	Artificial collective-like template	32	Full pairwise exchange, 992 flows, total 62.000 MiB	MoE-style dispatch/combine DAG; overlap=0.05	All-to-all scaling, collective comparison, baseline control
MoE routing	Artificial MoE template	32	Skewed expert dispatch, 320 flows, total 20.000 MiB	MoE dispatch-combine DAG; overlap=0.05	MoE hotspot/routing skew, baseline control
Mixed AI traffic	Composite (Read+AR+MoE) template	32	Heterogeneous READ/AR/MoE flows (488 total), 12.050 MiB	Multi-phase comm after backward; overlap=0.10	Mixed traffic, topology scaling, scheduler/buffer ablation, baseline control

A.2 Network Model Implementation Details

ESUN-Bench uses event-driven packet simulation over directional links with per-class queues. Packet timing includes queueing, serialization, propagation, and modeled flow-control and recovery delays. Configurable header overhead determines serialization size, buffer occupancy, and credit consumption. Scheduling supports FIFO, strict priority, and deficit weighted round-robin. ESUN header fields control traffic classification, deterministic path selection, and TTL checks.

PFC is approximated by occupancy-dependent local pause delays, while CBFC uses rate-replenished per-link, per-class credit windows. ECN performs threshold marking without a default sender-response loop. Link errors follow a size-, hop-, and BER-dependent probability; FEC scales the residual error probability, and recovery adds a configured replay delay. Packet and mechanism events are recorded for analysis. Communication-phase timings are reconstructed by flow and traffic class and projected onto an iteration DAG to estimate exposed communication and iteration time.

A.3 Additional Experimental Details

Unless a scenario group overrides them, experiments use a two-tier Clos topology with radix 64, 800 Gbps link bandwidth, 100 ns link propagation latency, shared 16 MB link buffers, 4 MB class buffers, strict-priority scheduling, and a BER of 10^{-12}. The default AI-side iteration model uses a 40 ms compute baseline. Packet service time includes configured header accounting: 4 bytes for the ESUN header and 42 bytes for the Ethernet/IP/UDP baseline. The default ESUN configuration enables flow labels, EH-ECN, PFC, CBFC, LLR, and FEC, with 256 KB credit windows, 800 Gbps credit refill, PFC high/low watermarks of 0.86/0.58, an ECN threshold of 0.62, and an FEC uncorrectable-error scaling factor of 0.01. Individual benchmark groups vary workload type, message size, packet size, offered load, hop count, topology oversubscription, routing skew, congestion-control settings, scheduler policy, and baseline mechanism toggles according to the scenario being evaluated.

Table 7: **Configurations of baseline Ethernet and layered ESUN mechanisms.** PFC is enabled in the Ethernet baseline used by the ablation matrix, so Ethernet is not a no-PFC baseline. Flow labels are available only when the ESUN header is enabled; they are used as deterministic routing entropy for path selection.

Config	Header accounting	PFC	CBFC	ECN	Sender response	LLR/FEC	Flow labels
Ethernet	Ethernet/IP/UDP accounting, 42 B; ESUN header disabled	On	Off	Off	None	Off / Off	Off
ESUN-Header	ESUN compact header accounting, 4 B	On	Off	Off	None	Off / Off	On
ESUN-FC	ESUN compact header accounting, 4 B	On	On	Off	None	Off / Off	On
ESUN-Congestion	ESUN compact header accounting, 4 B	On	On	On	No closed-loop sender-rate response; ECN marking is recorded as congestion feedback	Off / Off	On
ESUN-Full	ESUN compact header accounting, 4 B	On	On	On	No closed-loop sender-rate response; ECN marking is recorded as congestion feedback	On / On	On

For reproducibility, each case is run as a self-contained configuration and capped by the simulator's configured packet budget. The primary benchmark suite described in the paper contains 102 test

cases covering small-message behavior, congestion stress, incast, all-to-all and collective communication, MoE routing skew, mixed traffic, topology scaling, scheduler and buffer effects, AI trace-like workloads, and mechanism ablations.

Table 6 summarizes the six synthetic workload generators supported by ESUN-Bench, covering point-to-point, incast, AllReduce, all-to-all, MoE routing, and mixed composite AI traffic patterns. For each workload, we list its internal generator identifier, template origin, number of participating GPU nodes, communication characteristics including flow count and total payload volume, fixed computation baseline, execution DAG structure and communication-computation overlap ratio, as well as the corresponding subsets of the 102-case benchmark suite where this workload is deployed. All workloads adopt a unified computation time while differing in communication topology, traffic skew and DAG dependency logic, enabling controlled evaluation of diverse Ethernet fabric congestion and iteration overhead behaviors under representative distributed AI training loads.

Table 7 defines five comparative network configurations ranging from standard baseline Ethernet to the complete ESUN-Full stack. Each configuration incrementally enables ESUN-specific optimizations, including compact header compression, CBFC congestion control, ECN feedback recording, LLR/FEC error correction, and flow labels, while tracking the activation state of PFC as the base lossless transport primitive.

A.4 Visualization Interface and Video Demonstration

ESUN-Bench provides a visualization interface for inspecting workload configurations, packet movement, and network-to-iteration timing. The interface exposes workload and topology settings, scheduling policies, and mechanism toggles, alongside packet latency, goodput, congestion events, and iteration-level metrics. Packet animations and link-level summaries help locate traffic concentration, while the execution panel displays communication exposure and iteration progress.

An anonymous video demonstration is available at `https://anonymous.4open.science/r/bsun-67B7/index.html`. The video illustrates workload selection, configuration comparisons, and inspection of network and execution indicators across representative scenarios. Values shown during playback reflect the current demonstration state, including intermediate or approximate indicators labeled by the interface; the quantitative results in the paper are obtained from completed benchmark runs under the specified evaluation configurations. The video therefore illustrates the analysis workflow rather than serving as a reproduction of the reported tables.