Configuring Industrial Wireless Mesh Network via Dual-Mind Reasoning

Aitian Ma[†], Dongsheng Luo[†], Ali Maatouk[‡], Rex Ying[‡], Mo Sha[†]
[†]Knight Foundation School of Computing and Information Sciences, Florida International University
[‡]Department of Computer Science, Yale University
[†]{aima, dluo, msha}@fiu.edu, [‡]{ali.maatouk, rex.ying}@yale.edu

***Abstract*—Configuring industrial Wireless Mesh Networks (WMNs) requires jointly optimizing many settings, such as transmission power, channel allocation, and retransmission policy, to meet stringent performance requirements. Existing approaches rely on simulation-based optimization, but configurations that perform well in simulators often fail after deployment due to unmodeled interference, hardware diversity, and environmental dynamics. Prior domain adaptation techniques mitigate this simulation-to-reality gap by incorporating physical measurements but require extensive data collection in industrial facilities, imposing substantial data collection overhead. We present LLMNET, a framework that configures industrial WMNs without requiring expensive data from real-world deployments. LLMNETintroduces three key innovations: a multipath self-consistent Chain-of-Thought mechanism with consensus aggregation that stabilizes configuration decisions across diverse network conditions; a domain knowledge editing pipeline that corrects mismatched feature correlations between simulation and real deployments using expert-derived priors; and a cognitive-inspired dual-mind architecture comprising a lightweight Fast Mind for rapid heuristic inference and a deliberative Slow Mind for structured optimization, enabling adaptive reasoning under resource constraints. We evaluate LLMNETusing four widely used simulators (NS-3, Cooja, TOSSIM, and OMNeT++) and a 50-node WirelessHART testbed. Experimental results show that LLMNETimproves configuration accuracy by up to 40% over state-of-the-art baselines.**

***Index Terms*—Wireless Mesh Networks, Network Configuration, Simulation-to-Reality Gap, Large Language Models, Chain-of-Thought reasoning, Domain Knowledge Editing**

I. Introduction

Industrial Wireless Mesh Networks (WMNs) have become a critical communication infrastructure for modern industrial automation, enabling reliable sensing, monitoring, and control across manufacturing plants, chemical facilities, and power systems [1], [2], [3]. IEEE 802.15.4-based WMNs offer cost-effective deployment, flexible installation, and seamless integration with legacy infrastructure [4]. Standardized industrial wireless protocols including WirelessHART [5], ISA100.11a [6], IEC 62591 [7], and ZigBee [8] have driven widespread adoption, including more than 54,835 WirelessHART networks deployed globally [9], supporting millions of safety-critical sensor readings each day.

Despite this success, configuring industrial WMNs remains a challenging and labor-intensive task. Network performance depends on complex interactions among routing, scheduling, channel allocation, transmission power, and topology, making manual tuning error-prone and simulator-based optimization unreliable. Existing approaches largely fall into two categories. Simulation-only methods suffer severe simulation-to-reality gaps, leading to poor real-world performance. Hybrid approaches combining simulation with real-world labeled data improve accuracy but incur substantial deployment overhead: prior work [10] reports that achieving acceptable performance requires hundreds of physical measurements, significantly increasing commissioning time and operational cost.

Recent advances in Large Language Models (LLMs) with Chain-of-Thought (CoT) reasoning [11] offer a promising alternative by enabling structured reasoning over complex decision spaces. However, our empirical study demonstrates that directly applying LLMs to industrial WMN configuration fails in practice due to three fundamental challenges. First, LLMs lack domain-specific understanding of tightly coupled wireless parameters. Second, LLMs cannot reliably distinguish between simulated and real-world environments, leading to catastrophic deployment errors. Third, conventional prompting exhibits high variance across topologies, resulting in unpredictable configuration quality.

To address these challenges, we developed LLMNET, a framework that uses domain knowledge to guide reasoning for configuring industrial WMNs in real-world deployments. LLMNET stands out through three main features. First, it uses a multipath self-consistent CoT approach, which looks at different reasoning paths and uses majority voting to pick the best one, making the system much more reliable. Second, it employs a dual-layer architecture inspired by human cognition: a Fast Mind handles quick, heuristic-based decisions, while a Slow Mind takes over for complex optimization, allowing the system to balance speed and depth based on available resources. Finally, it features a dynamic knowledge editing tool that feeds expert networking insights directly into the LLM during inference. This helps the model bridge the gap between the simulated network and the physical deployment without needing a massive amount of data collected from the physical site.

We evaluate LLMNET using four widely used simulators (NS-3, Cooja, TOSSIM, OMNeT++) and a 50-node WirelessHART testbed. Experimental results demonstrate that LLMNET achieves up to 40% higher configuration accuracy than state-of-the-art baselines, while generalizing robustly to unseen network topologies and deployment conditions. More-

over, LLMNET produces interpretable reasoning traces, enabling practitioners to understand and validate system decisions. Specifically, this work makes the following contributions:

- We perform the first systematic study on the use of LLMs in industrial WMN configuration, identifying task comprehension deficits, domain gap blindness, and reasoning instability as key failure modes.
- We introduce LLMNET, an efficient dualmind reasoning system for Industrial WMN that includes MapReduce-based multipath CoT reasoning, dynamic domain knowledge editing, and dual mind collaboration.
- We develop a knowledge editing mechanism that enables deployment-ready WMN configuration without requiring high-cost data from real-world deployment.
- We validate LLMNET using multiple simulators and a testbed, LLMNET achieves new performance benchmarks with up to 40% accuracy improvement over existing methods.

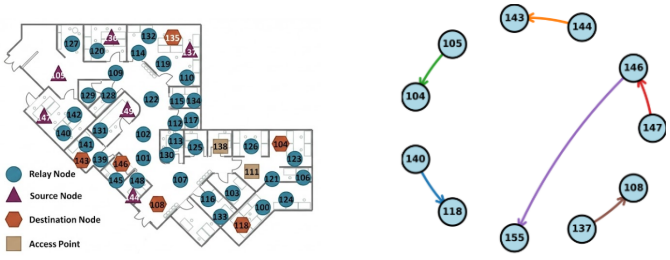
The remainder of this paper is organized as follows. Section II presents an empirical analysis of LLM behavior in industrial WMN configuration. Section III details the design of LLMNET. Section IV evaluates LLMNET. Section V reviews related work. Section VI concludes the paper.

II. EMPIRICAL STUDY

In this section, we first formulate the industrial WMN configuration problem from an LLM perspective. We then investigate the feasibility of leveraging LLMs to configure industrial WMN and narrow the simulation-to-reality gap. Next, we explore whether CoT reasoning can further close this gap. Finally, we analyze the factors limiting LLM generalization from a feature-correlation perspective.

A. Experimental Setup

We adopt the open-source WirelessHART implementation developed by Li et al., deploying six data flows across a physical testbed of 50 TelosB motes [12]. To manage network traffic, we utilized rate-monotonic scheduling—an optimal fixed-priority policy—to generate transmission schedules where each flow's delivery deadline is aligned with its period. Within this topology, the devices identified as 111 and 138 serve as the network access points. The physical arrangement of these nodes and the specific paths of the six data flows are illustrated in Figure 1a and Figure 1b, respectively.



(a) Deployment of TelosB motes. (b) Data flows.

Fig. 1: Deployment details and data flows for WirelessHART experiments.

The network configuration parameters include: Packet Reception Rate (PRR) threshold R, number of channels C, and transmission attempts A. Network performance metrics are end-to-end latency L, battery lifetime B, and reliability E. The parameter ranges are $R \in \{0.7, 0.71, \ldots, 0.9\}$, $C \in \{1, \ldots, 8\}$, and $A \in \{1, 2, 3\}$, producing 744 total configurations. After removing redundancies that yield identical routes and schedules, 88 distinct configurations remain.

To evaluate each network configuration, we replicate the environment in four distinct simulators: TOSSIM, Cooja, OMNeT++, and NS-3. We feed these simulators with PRR and noise traces, as well as the specific routes and schedules collected directly from the testbed. Reliability is quantified throughout these experiments using the Packet Delivery Ratio (PDR). For every unique configuration, each simulator generates 75 traces, resulting in a total of 6,600 traces per simulator.

We denote this dataset collected from simulators as D_S, and the dataset collected from real-world deployment is denoted as D_P. While D_S is relatively easy to generate at scale, the collection of D_P involves significant overhead. Collecting the 6,600 physical traces required approximately four days of continuous operation and consumed $23kJ$ of device energy. This comparison highlights the substantial resource gap and the high cost of physical data collection compared to the efficiency of simulation-based exploration.

B. Configuring Industrial WMNs from a LLM Perspective

From an algorithmic perspective, the goal of industrial WMN configuration is to identify a parameter vector $y = \{R, C, A\}$, representing the reception threshold, channel count, and transmission attempts, that satisfies a specific set of application-level performance requirements $x = \{L, B, E\}$ regarding latency, battery life, and reliability. Mathematically, this can be viewed as an inverse problem where we seek a mapping function that translates desired performance outcomes into the configuration settings necessary to achieve $y = f(x)$.

In this paper, we leverage a large dataset of simulation results and a small set of physical measurements to train our models. This allows us to define an LLM-based configuration function:

$$y^* = \text{LLM}(f_S(x), \mathcal{K}) \tag{1}$$

where $f_S(x)$ is the initial simulator outputs while $\mathcal{K}$ represents the injected domain knowledge. This structure enables the model to bridge the gap and generate deployment-ready configurations without requiring massive amounts of data from real-world deployment.

C. Feasibility of Leveraging LLMs for WMN Configuration

LLMs have demonstrated strong generalization capabilities across Natural Language Processing (NLP) and Computer vision (CV) tasks. To explore whether LLMs is capable of performing industrial WMN configuration well and close the

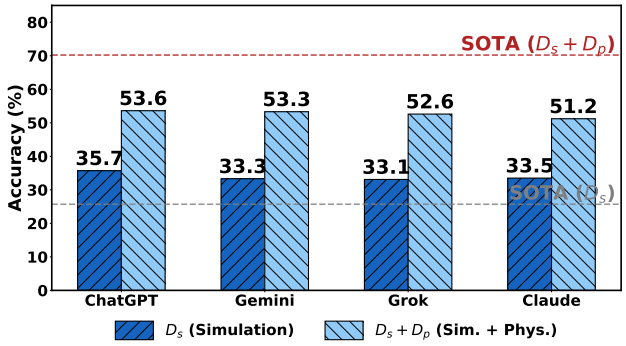


Fig. 2: Testing accuracy of LLMs under two training regimes: simulation data only versus hybrid data. The red dashed line indicates the performance of [13].

simulation-to-reality gap[1], we conduct two sets of experiments: providing LLMs with D_S alone, and with both D_S and D_P. We evaluate four leading LLMs (ChatGPT, Gemini, Grok, and Claude) against a state-of-the-art (SOTA) WMN configuration method [13]. We employ a simple task-oriented prompt consisting of a problem description, summaries of the training and testing datasets, and the objective of maximizing prediction accuracy on the test set using the training data.

As Figure 2 shows, when providing the LLM with simulation data only, LLMs achieved an accuracy ranging between 33% and 35%. While this is notably better than the simulation-only baseline of roughly 26%, it is not yet deployment-ready. When we add physical data, LLM performance improved significantly to the 51–53% range. However, this still falls short of our handcrafted baseline, which reached over 70% accuracy with the same data. This suggests that while LLMs possess strong reasoning heuristics, they currently lack the fine-grained optimization capabilities inherent in dedicated industrial mesh protocols.

D. Feasibility of CoT for Closing Simulation-to-reality Gap

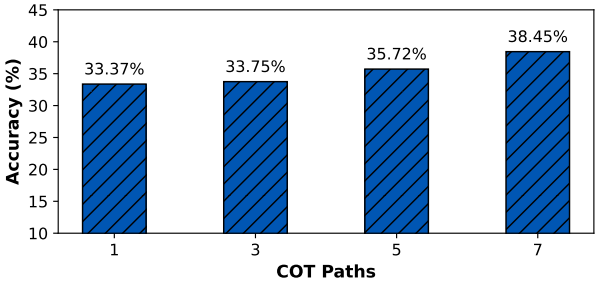


Fig. 3: Effect of the number of CoT reasoning paths on configuration accuracy.

We next investigate whether performance can be improved by enhancing the model's reasoning process alone. We provide LLMs with simulation data D_S and apply more sophisticated prompting strategies. Instead of relying on a single reasoning trajectory, we require the model to generate multiple reasoning paths and select the best-performing outcome among them.

As Figure 3 shows, the best prediction performance varies with different numbers of CoT reasoning paths. Accuracy increases monotonically with the number of paths, rising from approximately 33% to 38% with seven paths, while incurring increased reasoning overhead. This corresponds to a relative improvement of more than 15%, achieved without incorporating additional physical data. These results suggest that for industrial WMN configuration, self-consistency mechanisms in multipath CoT enable LLMs to effectively filter suboptimal configurations. Although LLMs still lag behind models trained on extensive physical datasets, scaling reasoning depth offers a cost-effective approach to close the simulation-to-reality gap without the need of collecting expensive data from real-world deployment.

E. Simulation-to-Reality Gap from a Feature Correlation Perspective

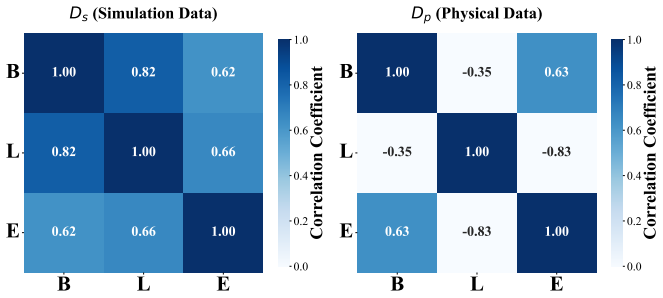


Fig. 4: Comparative correlation analysis between simulation and real-world deployment.

To better understand why performance degrades when moving from simulation to real-world deployment, we analyzed the difference in feature correlation between simulation data and real-world data. Figure 4 illustrates this simulation-to-reality gap by comparing the Pearson correlation matrices on the simulation data (D_s) and the physical Data (D_p). In the idealized environment of the NS-3 simulator, network metrics exhibit strong, monotonic relationships. The D_s matrix reveals a high degree of coupling between features, particularly between Battery (B) and Latency (L), with a correlation coefficient of 0.82. These deterministic scaling laws allow LLM to converge on simplified rules. However, these rules are based on a level of predictability that is rarely present in practical deployments.

The transition to the physical domain (D_p) introduces environmental stochasticity, such as multipath fading, shadow fading, and non-stationary interference. These factors decouple previously related parameters. The strong positive correlation (0.82) in simulation collapses to a negative correlation (-0.35) in reality. Furthermore, the relationship between Latency (L) and Reliability (E) shifts from a moderate positive correlation (0.66) to a strong negative correlation (-0.83), suggesting that

[1]Prediction accuracy decreases a lot when we apply the model trained on the simulation dataset to real-world deployments.

real-world retransmission mechanisms or power-saving states introduce trade-offs not captured by the simulator.

III. Design

This section introduces LLMNet, an efficient framework that performs dual-mind reasoning on simulation data. It is designed for immediate real-world application without using expensive real-world data.

A. System Overview

LLMNet serves as an end-to-end pipeline that leverages simulation-derived datasets D_S to map WMN performance requirements, including Latency, Battery Lifetime, and Reliability, onto optimal network configurations D_P, including PRR, channel assignments, and transmission attempt limits. As Figure 5 shows, LLMNet is orchestrated through three integrated components: Map-Reduce CoT Reasoning Module, Knowledge Editing Module, and Dual-Mind Collaboration Module.

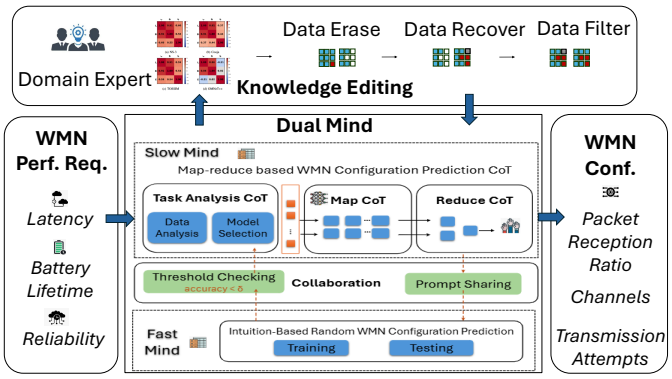


Fig. 5: LLMNet architecture overview.

The process begins with the Map-Reduce CoT Reasoning Module, which functions as a high-capacity inference engine designed to distill structural network insights from D_S by decomposing complex, non-linear wireless dependencies into parallelized reasoning paths. To bridge the gap between simulation environments and physical deployments, the Knowledge Editing Module employs an automated feature correlation adaptation mechanism. This component refines the initial simulation-based knowledge K_S into a more robust, generalized representation K_N that accounts for real-world environmental stochasticity. Finally, the Dual-Mind Collaboration Module acts as the decision-making core, harmonizing adapted knowledge with real-time constraints to generate accurate configuration predictions. By synthesizing theoretical simulation bounds with empirical adaptation, LLMNet ensures that the predicted configurations remain both performance-optimal and physically viable in practical WMN deployments.

B. Map-Reduce CoT Reasoning

The Map-Reduce CoT framework is designed to address the scalability limitations of conventional CoT reasoning when applied to industrial WMN configuration problems. Rather than relying on a single reasoning trajectory, LLMNet explores multiple candidate reasoning paths in parallel and aggregates their outputs using performance-aware weighting. This approach improves robustness against individual model errors, simulation data overfitting, and domain-specific blind spots, while remaining computationally tractable for real-time use.

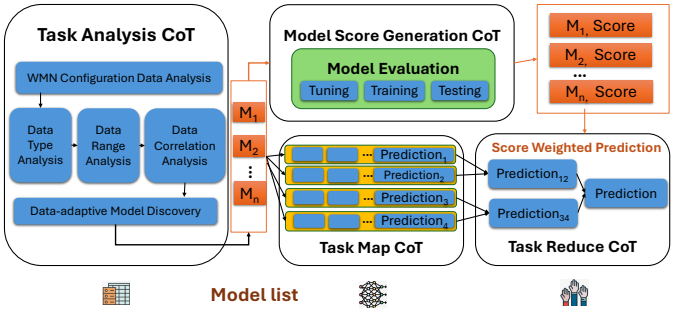


Fig. 6: Map-Reduce CoT pipeline.

As Figure 6 shows, the pipeline begins with task analysis, where the system inspects the input configuration request and associated performance metrics (L, B, E) to identify dominant constraints, detect anomalous correlations, and estimate task difficulty. Based on this analysis, the system constructs a candidate set of reasoning models $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$ tailored to the task structure. Model selection is implemented using complexity-based $O(n \log n)$ priority ranking, allowing rapid pruning of weak or redundant candidates.

Each selected model then undergoes performance scoring using validation traces aligned by the knowledge editing module. The resulting scores capture a combination of predictive accuracy, robustness under domain shift, and computational cost, and are normalized to form a weighted ranking.

$$\mathcal{S} = \{(M_1, s_1), (M_2, s_2), \ldots, (M_n, s_n)\}. \tag{2}$$

The Map stage executes all models in parallel on the target configuration instance, producing a set of candidate predictions $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. This parallel execution exploits model diversity to explore multiple reasoning hypotheses simultaneously and achieves near-linear speedup under moderate system load.

In the Reduce stage, predictions are aggregated using score-weighted averaging,

$$p_{\text{final}} = \frac{\sum_{i=1}^{n} s_i \cdot p_i}{\sum_{i=1}^{n} s_i}, \tag{3}$$

which biases the final decision toward consistently reliable models while retaining robustness to outliers. This aggregation strategy provides stability guarantees similar to ensemble learning while preserving the interpretability and structure of explicit reasoning chains.

The Map-Reduce CoT framework is formulated as a multi-objective optimization problem that balances predictive accuracy, computational efficiency, and robustness to domain shifts:

$$\min_{\mathcal{M}} \left[\alpha_1 \cdot \mathcal{L}_{\text{pred}}(\mathcal{M}) + \alpha_2 \cdot \mathcal{C}_{\text{comp}}(\mathcal{M}) + \alpha_3 \cdot \mathcal{R}_{\text{robust}}(\mathcal{M})\right], \tag{4}$$

where $\mathcal{L}_{\text{pred}}$ denotes prediction loss, $\mathcal{C}_{\text{comp}}$ captures inference cost, and $\mathcal{R}_{\text{robust}}$ measures sensitivity to domain mismatch across simulators and physical deployments. The coefficients α_1, α_2, and α_3 enable explicit trade-offs between accuracy, latency, and robustness, allowing the system to adapt to operational constraints in real deployments.

To enable online adaptation, model weights are updated using an exponentially smoothed performance estimator,

$$s_i(t) = \eta \cdot s_i(t-1) + (1-\eta) \cdot \text{Performance}_i(t), \tag{5}$$

where $\eta \in [0, 1]$ controls the temporal discounting of historical performance. This formulation allows LLMNet to track non-stationary network conditions while avoiding oscillatory behavior in model selection.

C. Knowledge Editing

The Knowledge Editing module forms the foundation of the LLMNet pipeline by explicitly correcting distributional mismatches between simulated and real network data. Instead of relying solely on prompt engineering or downstream fine-tuning, this module reshapes the statistical structure of input features before any LLM reasoning occurs. This design choice reflects the observation that most failures under domain shift arise from corrupted feature relationships rather than isolated prediction errors.

The module incorporates correlation priors derived from domain experts, capturing qualitative physical laws such as latency–reliability trade-offs and energy–throughput coupling under contention. These priors are encoded as dual-state correlation coefficients that distinguish between high-confidence and low-confidence operating regimes. Given partially observed features, missing values are reconstructed using

$$\begin{aligned} L' &= \sum_{k \in \{B,E\}} (\rho_{Lk}^{\text{high}} \cdot k + \rho_{Lk}^{\text{low}} \cdot k) + \epsilon_L, \\ B' &= \sum_{k \in \{L,E\}} (\rho_{Bk}^{\text{high}} \cdot k + \rho_{Bk}^{\text{low}} \cdot k) + \epsilon_B, \\ E' &= \sum_{k \in \{L,B\}} (\rho_{Ek}^{\text{high}} \cdot k + \rho_{Ek}^{\text{low}} \cdot k) + \epsilon_E, \end{aligned} \tag{6}$$

where ϵ represents residual noise sampled from empirical variance to preserve stochastic realism.

Operationally, knowledge editing proceeds in three stages. First, the system applies controlled stochastic erasure by masking a fraction of feature values, forcing the model to infer missing information from relational structure rather than memorizing simulator artifacts. Next, the recovery stage reconstructs masked entries using the expert-informed priors above, preserving both dominant correlations and weaker cross-metric dependencies. Finally, a filtering stage removes samples that violate physical feasibility constraints, such as negative latency or implausible battery lifetimes. This process yields a refined dataset whose statistical structure more closely resembles real deployments, thereby improving downstream reasoning stability.

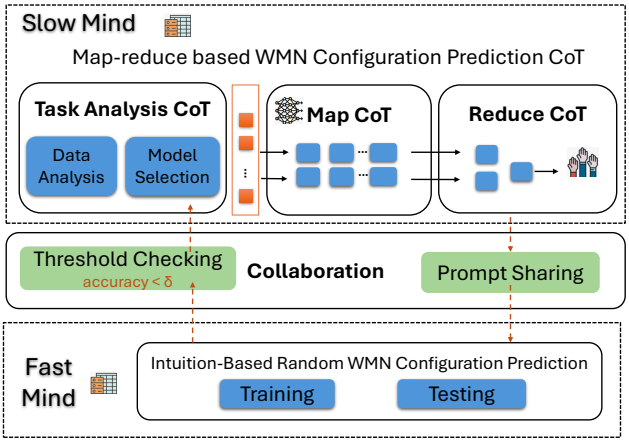


Fig. 7: Dual-mind collaboration mechanism.

D. Dual-Mind Collaboration

The core inference engine of LLMNet is built around a Dual-Mind collaboration protocol inspired by cognitive dual-process theory and adapted to the operational constraints of network control systems.

The Fast Mind serves as the low-latency execution path. It operates on pre-aligned features produced by the knowledge editing module and relies on lightweight heuristics and cached reasoning templates to produce rapid configuration decisions. In common operating regimes, where traffic patterns, interference conditions, and topology dynamics exhibit familiar structure, the Fast Mind can classify tasks and produce decisions faster, making it suitable for latency-sensitive reasoning tasks.

The Slow Mind is activated only when the Fast Mind detects ambiguity or low confidence. In such cases, the system escalates the execution to a deliberative reasoning pathway that invokes the full Map-Reduce CoT pipeline. This pathway performs multi-step analysis, explores multiple candidate reasoning trajectories, and aggregates results to maximize robustness under uncertainty. Although computationally more expensive, the Slow Mind is used sparingly and only when accuracy gains justify the additional latency.

The interaction between the two minds is governed by an adaptive threshold mechanism,

$$\delta(t) = \delta_0 + \kappa \cdot \text{PerformanceGap}(t-1), \tag{7}$$

where PerformanceGap measures recent accuracy differences between the Fast and Slow paths and κ controls the adaptation rate. This formulation allows the system to dynamically adjust

its escalation policy in response to workload characteristics and evolving network conditions.

In addition to escalation control, LLMNet supports bidirectional knowledge transfer between the two reasoning modes. When the Slow Mind successfully resolves a complex task, it extracts compact reasoning patterns from its internal traces and injects them into the Fast Mind's prompt templates,

$$\mathbf{P}_f(t+1) = \beta \cdot \mathbf{P}_f(t) + (1-\beta) \cdot \Phi\left(\mathcal{T}_s(t)\right) \tag{8}$$

where β controls the learning rate of this transfer process. Over time, this mechanism enables the Fast Mind to absorb previously complex reasoning patterns, reducing future escalation frequency and improving overall system efficiency without requiring full retraining.

IV. Evaluation

We evaluate LLMNet against state-of-the-art baselines across multiple simulation environments and a physical testbed. Experimental results show that LLMNet provides good network configuration models without the need to collect data from physical deployments, improving accuracy by up to 40% compared to existing approaches. We also explore the effect of key design parameters, including knowledge editing ratios and CoT width, to identify the configurations that best balance accuracy with computational efficiency.

A. Implementation and Experimental Setups

We build LLMNet using ChatGPT as the foundational language model and compare it against five baselines: Teacher-Student adaptation [10], WMN-CDA [14], LLM-CoT, LLM-ToT, and LLM-ToT+KE (incorporating knowledge editing). We implement all baselines in PyTorch [15] and run our evaluations on an NVIDIA A100 GPU with 80 GB of memory. We use the data collected from a physical testbed consisting of 50 TelosB motes as the ground truth data [13]. This testbed captures real-world network behavior across 88 distinct configurations, defined by variations in PRR thresholds, channels, and transmission attempts. By measuring latency, battery lifetime, and reliability under each configuration, 6,600 real-world traces have been collected. To rigorously test cross-domain generalization, we replicate this exact network topology and parameter space in four different simulators (NS-3, Cooja, TOSSIM, and OMNeT++) and generate an equivalent dataset for each.

B. Performance in Physical Deployment

We first assess the generalization ability of LLMNet. We provide the model with simulation data and test them directly on the held-out physical testbed data. This mirrors the practical challenge where engineers use simulators for initial exploration but ultimately deploy to a real network. Figure 8 illustrates the performance across all four simulators.

The results show that LLMNet consistently improves performance in every environment. When train on NS-3, LLMNet reaches 81% accuracy, significantly outpacing Teacher-Student (28%), WMN-CDA (32%), LLM-CoT (32%), LLM-ToT (41%), and LLM-ToT+KE (50%). This represents a 31-percentage-point gain over the strongest baseline. We observe similar margins on TOSSIM (82% vs. 48%), Cooja (64% vs. 34%), and OMNeT++ (63% vs. 33%).

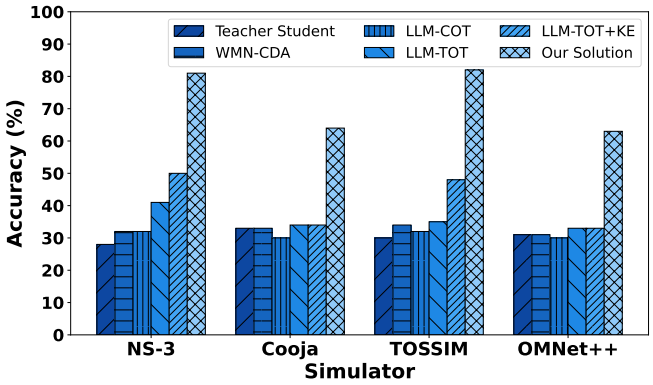


Fig. 8: Performance across simulators in the real-world testbed without data collected from real-world deployment.

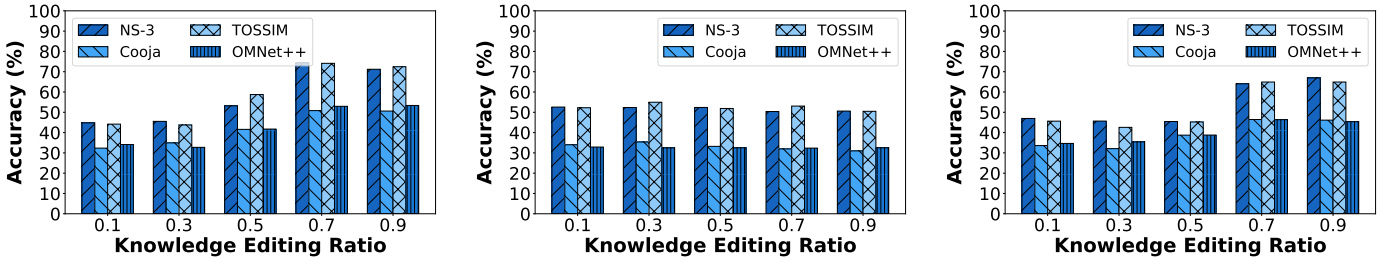
While the step up from LLM-CoT to LLM-ToT demonstrates the value of multipath reasoning, and adding knowledge editing provides further gains, LLMNet achieves the most dramatic improvements. By integrating the Dual-Mind architecture with adaptive knowledge editing, it effectively bridges the gap between simulation and real-world deployment.

C. Effect of Knowledge Editing Ratio

We then explore the effect of Knowledge Editing Ratio on the performance of LLMNet. We vary the knowledge editing ratio (γ) from 0.1 to 0.9 and repeat our experiments. As Figure 9 shows, the prediction performance is highly sensitive to the degree of editing applied. For example, under NS-3 and TOSSIM, accuracy jumps significantly as the ratio increases from 0.1 to 0.7. In the case of NS-3, accuracy rises from roughly 45% to 74%, while TOSSIM sees a similar increase. However, pushing the ratio to 0.9 causes a slight decrease in performance, suggesting that excessive editing can lead to overfitting. Cooja and OMNeT++ show smaller but steady improvements. The results reveal that different metrics respond differently to editing intensity.

The prediction accuracy follows a different pattern when varying the battery lifetime knowledge editing ratio. Here, the model performs best with minimal intervention. NS-3 achieves its peak accuracy of 52.6% at the lowest ratio of 0.1 and does not improve with higher ratios. TOSSIM peaks slightly higher at 0.3, but degrades if we edit too aggressively. This suggests that energy metrics are delicate; over-correcting them can introduce artifacts that pollute the physical energy consumption patterns learned from the simulation data.

Prediction accuracy benefits the most from aggressive editing on reliability. NS-3 accuracy increases from 47% to 67% at the maximum editing ratio, and TOSSIM sees a similar increase. Even lower-fidelity simulators show significant gains here, indicating that knowledge editing helps compensate for



(a) Latency (b) Battery (c) Reliability

Fig. 9: Effect of knowledge editing ratio (0.1–0.9) on LLMNET's performance.

fundamental limitations in how wireless channels are modeled. These divergent patterns suggest that the optimal strategy depends on the metric: latency and reliability favor moderate-to-high editing ratios, while battery lifetime requires a smaller editing ratio.

D. CoT Width Analysis

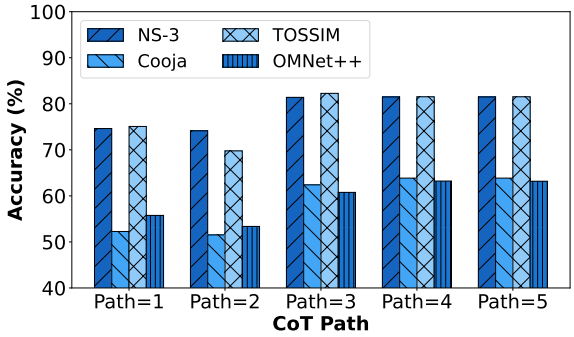


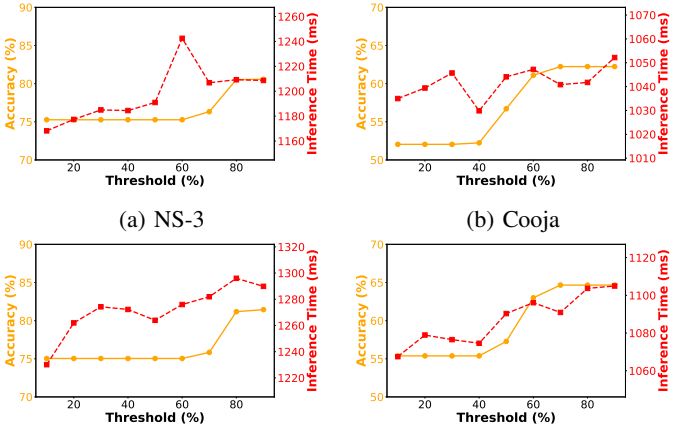
Fig. 10: Effect of CoT width on LLMNET's performance in physical deployment.

The CoT width determines how many parallel reasoning paths the model explores before aggregating the results. Figure 10 shows how accuracy changes as we increase the width from a single path to five parallel paths. Moving from single-path to multipath reasoning clearly boosts performance. NS-3 accuracy improves from 74.6% to 81.5% when using 4 or 5 paths. A similar improvement can be observed from TOSSIM, where the prediction accuracy reaches 82.3% on three paths. Lower-fidelity simulators also benefit, with OMNeT++ and Cooja both showing noticeable improvements.

However, we observe diminishing returns beyond three paths. Accuracy tends to plateau at 4 or 5 paths across all simulators. This indicates that adding more trajectories beyond this point adds little value and may simply introduce noise. We conclude that a width of 3 to 4 paths offers the best balance, providing sufficient reasoning diversity without wasting computational resources.

E. Runtime Scalability Analysis

We analyze accuracy-runtime trade-offs as the confidence threshold for Fast Mind versus Slow Mind selection varies from 10% to 90%. This adaptive mechanism uses an efficient



(a) NS-3 (b) Cooja

(c) TOSSIM (d) OMNeT++

Fig. 11: Accuracy-runtime trade-offs as confidence threshold varies from 10% to 90%.

Fast Mind for high-confidence predictions while reserving a sophisticated Slow Mind for complex cases. Figure 11 shows the trade-offs across all simulators. Under NS-3, accuracy improves from 74.8% (10% threshold) to 81.5% (70% threshold), with runtime increasing modestly from 1.17s to 1.24s (6% overhead). Similar patterns occur across simulators: Cooja (52.1% to 63.9% accuracy, 1.03s to 1.05s, 2% overhead), TOSSIM (75.3% to 82.3% accuracy, 1.23s to 1.30s, 6% overhead), and OMNeT++ (55.2% to 63.2% accuracy, 1.07s to 1.11s, 4% overhead). Higher confidence thresholds consistently enhance accuracy across all simulators while maintaining minimal computational overhead below 10%.

Confidence thresholds above 70% achieve an optimal balance between performance and efficiency across all environments. The adaptive selection maintains accuracy within 1-2% of peak performance while achieving substantial runtime reductions, making LLMNET suitable for real-time network configuration in resource-constrained scenarios.

V. Related Work

A. Industrial WMN Configuration and Optimization

Industrial WMN configuration involves optimizing interdependent parameters—packet reception thresholds, channel allocation, and transmission policies—to achieve stringent

Quality of Service (QoS) requirements [16], [17]. Traditional handcrafted approaches achieve limited scalability in dynamic environments. Recent machine learning based methods have shown significant improvements. Park et al. [18] achieved PDR improvement using RL for WirelessHART, while Silva et al. [3] demonstrated significant energy reduction through genetic algorithms. Deep learning based approaches [19] enable real-time adaptation but require extensive training data and lack interpretability—critical barriers for safety-critical industrial deployments [20].

B. Large Language Models for Technical Reasoning

LLMs demonstrate remarkable technical capabilities through CoT reasoning [11]. CoT enables step-by-step problem decomposition with enhanced interpretability. Recent advances include Self-Consistency CoT with majority voting, Tree of Thoughts exploring structured trajectories [21], and Graph of Thoughts enabling complex reasoning exploration. LLMs achieve human-expert performance in mathematical reasoning, competitive programming, and geometric theorem proving [22]. Networking applications focus on code generation and troubleshooting. However, existing approaches treat LLMs as passive tools, lacking systematic frameworks for technical domain adaptation and multi-agent coordination.

C. Knowledge Editing and Domain Adaptation

Knowledge editing enables modification of model behavior without full retraining, offering a lightweight alternative to costly end-to-end fine-tuning [23]. Existing approaches include parameter patching, retrieval-augmented generation, and prompt-based editing, each providing different trade-offs between flexibility, stability, and deployment complexity. Recent studies have begun exploring domain adaptation through knowledge editing, with Liu et al. [24] demonstrating prompt-based control of factual updates and Gao et al. [25] showing that targeted knowledge injection can improve cross-domain generalization.

VI. Conclusions

In this paper, we introduced LLMNet, a framework guided by domain knowledge that uses a dual-mind architecture to achieve robust WMN configuration. LLMNet effectively closes the gap between simulation and reality by combining three primary innovations. First, it utilizes a multipath self-consistent CoT reasoning engine. Second, it implements a hybrid Fast and Slow Mind architecture that balances computational efficiency with reasoning depth. Finally, it uses a dynamic knowledge editing mechanism to align correlations found in simulations with real-world physical constraints. Evaluations with four simulators (NS-3, Cooja, TOSSIM, OMNeT++) demonstrate LLMNet's ability to provide good network congigurations with the need of collecting costly data from physical deployments.

Acknowledgment

This work was supported in part by the National Science Foundation under grants CNS-2150010, ECCS-2242700, and IIS-2529283.

References

[1] X. Cheng, M. Sha, and D. Chen, "Configuring Industrial Wireless Mesh Networks via Multi-Source Domain Adaptation," in *EWSN*, 2024.

[2] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and Real-Time Communication in Industrial Wireless Mesh Networks," in *RTAS*, 2011.

[3] C. Silva, Y. Oliveira, C. Celes, R. Braga, and C. Oliveira, "Performance Evaluation of Wireless Mesh Networks in Smart Cities Scenarios," in *EATIS*, 2018.

[4] S. K. Gupta and P. Kuila, "Wireless Sensor Network Survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2015.

[5] "HART Communication Foundation: WirelessHART Standards," https://www.fieldcommgroup.org/technologies/hart/wirelesshart, 2009, accessed: 2025-07-23.

[6] "ISA100 Wireless Systems for Automation," https://www.isa.org/isa100, 2010, accessed: 2025-07-23.

[7] "IEC 62591:2011 - Industrial communication networks – Wireless communication network and communication profiles – WirelessHART," https://webstore.iec.ch/publication/11128, 2011, accessed: 2025-07-23.

[8] "ZigBee Specification Overview," https://csa-iot.org/technology/zigbee/, 2012, accessed: 2025-07-23.

[9] "WirelessHART Deployments by Emerson Process Management," https://www.emerson.com/en-us/catalog/emerson-wirelesshart, 2018, accessed: 2025-07-23.

[10] J. Shi, A. Ma, X. Cheng, M. Sha, and X. Peng, "Adapting Wireless Network Configuration from Simulation to Reality via Deep Learning based Domain Adaptation," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 1983–1998, 2024.

[11] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *NeurIPS*, 2022.

[12] Washington University, "WCPS: Wireless Cyber-Physical Simulator," http://wsn.cse.wustl.edu/index.php/WCPS:_Wireless_Cyber-Physical_Simulator, 2026, accessed: Feb. 04, 2026.

[13] J. Shi, M. Sha, and X. Peng, "Adapting Wireless Mesh Network Configuration from Simulation to Reality via Deep Learning based Domain Adaptation," in *NSDI*, 2021.

[14] A. Ma and M. Sha, "WMN-CDA: Contrastive Domain Adaptation for Wireless Mesh Network Configuration," in *SAC*, 2025.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An Imperative Style, High-Performance Deep Learning Library," *NeurIPS*, 2019.

[16] B. Milic, S. Brack, and R. Naumann, "Hierarchical Configuration System for Wireless Mesh Networks in Manufacturing Industry," in *WMNC*, 2014.

[17] H.-C. Lee and H.-H. Lin, "Design and Evaluation of An Open-source Wireless Mesh Networking Module for Environmental Monitoring," *IEEE Sensors Journal*, vol. 16, no. 7, pp. 2162–2171, 2015.

[18] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless Network Design for Control Systems: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 978–1013, 2017.

[19] A. Ma, J. T. Rodriguez, M. Sha, and D. Luo, "Sensorless Air Temperature Sensing Using LoRa Link Characteristics," in *DCOSS-IoT*, 2025.

[20] J. Shi and M. Sha, "Parameter Self-Configuration and Self-Adaptation in Industrial Wireless Sensor-Actuator Networks," in *INFOCOM*, 2019.

[21] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," in *NeurIPS*, 2023.

[22] Q. Liu, X. Zhao, Y. Wang, Z. Zhang, H. Zhong, C. Chen, X. Li, W. Huang, and F. Tian, "Bridge the Domains: Large Language Models Enhanced Cross-domain Sequential Recommendation," in *SIGIR*, 2025.

[23] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Fast Model Editing at Scale," in *ICLR*, 2022.

[24] S. Wang, Y. Zhu, H. Liu, Z. Zheng, C. Chen, and J. Li, "Knowledge editing for large language models: A survey," *ACM Computing Surveys*, vol. 57, no. 3, pp. 1–37, 2024.

[25] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint: 2312.10997*, 2024.