# AI-assisted Closed-loop Network Optimization Using Network Simulator 3 (NS-3) and Large Language Models

Aziz Alshehri [1,*] and Jebreel Omar Alamri [2]

[1] Department of Cybersecurity, College of Computing, Umm Al-Qura University, Makkah, Saudi Arabia
[2] Department of Computer Science, College of Engineering and Computing in Al-Qunfudhah, Umm Al-Qura University, Makkah, Saudi Arabia
Email: aaashehri@uqu.edu.sa (A.A.); joammarey@uqu.edu.sa (J.O.A.)
*Corresponding author

***Abstract*—This study presents a Large Language Model (LLM)-assisted closed-loop network optimization framework that integrates Network Simulator 3 (NS-3) v3.37 with a large language model accessed through the OpenAI Application Programming Interface (API) and a deterministic Python controller. The framework executes simulations, extracts performance metrics, validates LLM-generated candidate configurations, and applies controller-side safeguards before launching the next NS-3 run. The revised Wireless Fidelity (Wi-Fi) experiments use a reconstructed IEEE 802.11 g model and evaluate overloaded User Datagram Protocol (UDP) scenarios with 3 and 10 station nodes. In addition to the initial LLM-guided optimization, the study compares the LLM controller with Bayesian Optimization under the same search space, evaluation budget, simulator, metrics, and utility function. The results are scenario-dependent rather than universally favoring either optimizer. In 3 station nodes (3-STA) validation, Bayesian Optimization achieved the higher utility, 112.587 ± 0.011 versus 89.200 ± 0.012 for the LLM. In 10-STA validation, the LLM-guided configuration achieved 24.996 Mbps aggregate throughput with 0.004% packet loss and 0.424 ms latency, compared with 27.151 Mbps, 9.482% loss, and 180.562 ms for Bayesian Optimization under the same evaluation setting. Reliability analysis showed that controller-side safeguards were essential: 57 API calls produced 36 valid JavaScript Object Notation (JSON) responses, 21 malformed outputs, 29 duplicate proposals, and zero manual interventions.**

***Keywords*—Large Language Models (LLMs), Network Simulator 3 (NS-3) simulation, adaptive optimization, self-organizing networks, Wireless Fidelity (Wi-Fi) performance, Artificial Intelligence (AI)-assisted network management**

## I. Introduction

The rapid evolution of communication networks and intelligent systems has expanded the complexity of network design, control, and optimization. Traditional static configurations and rule-based methods often struggle to adapt to dynamic, heterogeneous conditions where traffic loads, link capacities, and user demands vary over time [1, 2]. This growing complexity motivates intelligent optimization mechanisms that can reason over measurements and update parameters during iterative closed-loop evaluation.

Network simulation plays a vital role in developing and validating adaptive systems. Network Simulator 3 (NS-3), a discrete-event simulator widely adopted in research and academia, provides a robust platform for testing communication protocols and performance metrics under controlled conditions [3–6]. However, simulation-guided optimization still requires careful configuration design, repeated runs, and interpretation of output metrics, particularly in high-dimensional design spaces [7–10].

In this study, ChatGPT is treated as a user-facing product/interface rather than as the language model itself. The experiments access an OpenAI Application Programming Interface (API) model through a fixed controller configuration; the Phase-3 artifact records gpt-4o-mini as the model used for the comparison run. This distinction is important because ChatGPT may route user interactions to different model families, whereas the reported experiments require a fixed API configuration for reproducibility. GPT-series technical literature is cited only as background for the model family and not as evidence that the ChatGPT product itself was used.

Recent advances in Large Language Models (LLMs), including GPT-series models accessed through products and API interfaces, have introduced a new paradigm for intelligent automation and decision support [11–15]. LLMs can interpret structured or unstructured data, generate context-aware recommendations, and support adaptive optimization without extensive pre-training in a specific domain. In this paper, the LLM is positioned as a semantic assistant for simulation-guided configuration rather than as a replacement for formal optimization algorithms [16, 17].

In this study, we propose an LLM-assisted closed-loop network optimization framework that integrates NS-3 with

Manuscript received December 30, 2025; revised February 13, 2026; accepted August 17, 2026; published November 12, 2026.

a validated controller and an API-accessed LLM. The framework is designed for reproducible parameter selection under explicit bounds, not for unconstrained modification of physical-layer capacity. It evaluates wired and reconstructed IEEE 802.11 g Wireless Fidelity (Wi-Fi) scenarios and compares LLM-guided search with Bayesian Optimization under equal budgets.

The framework was experimentally validated on a previously established wired topology and on newly reconstructed IEEE 802.11 g Wi-Fi experiments. The Wi-Fi results show that LLM-guided search can find useful congestion-reducing configurations, while the LLM-vs-Bayesian comparison demonstrates scenario-dependent performance rather than universal LLM superiority.

The main contributions of this paper are summarized as follows:

- A reproducible NS-3-LLM closed-loop architecture for controller-mediated parameter selection.
- Controller-side validation and safety mechanisms for LLM-generated configuration proposals, including parsing, bounds checks, duplicate detection, retries, and deterministic fallback.
- Controlled Wi-Fi evaluation under congestion conditions with 3 station nodes (3 STAs) and 10 station nodes (10 STAs) using a reconstructed IEEE 802.11 g model.
- Equal-budget comparison against Bayesian Optimization and quantitative characterization of LLM reliability, including malformed responses, duplicate proposals, retries, fallback usage, and API latency.

The remainder of this paper is structured as follows: Section II reviews related literature on AI-assisted and LLM-based network optimization. Section III presents the proposed methodology, covering the system architecture, the closed-loop adaptive workflow, and the implementation and controller-validation details for both wired and Wi-Fi scenarios. Section IV describes the experimental design and setup. Section V reports the results and evaluation, including computational overhead. Section VI discusses the findings, limitations, and implications. Section VII concludes the paper and outlines future research directions.

## II. Literature Review

The growing complexity of modern wireless and communication networks has motivated significant progress in simulation methodologies and Artificial Intelligence (AI)-assisted optimization techniques. This section reviews existing literature in three areas relevant to our work: network simulation frameworks, AI-driven optimization, and the emerging application of LLMs for intelligent system control. We also identify key research gaps that motivate the proposed adaptive framework.

### A. Network Simulation and Modeling

Discrete-event simulators such as NS-3 have become foundational tools in modeling wired, wireless, and next-generation network behavior. NS-3 supports modular extensions for diverse protocols and technologies, and prior work has also connected NS-3 with AI frameworks for networking research [3–6]. However, such simulation workflows often rely on predefined parameter choices or external learning pipelines rather than LLM-generated, controller-validated parameter proposals.

Prior surveys show that network simulators are widely used to train, validate, and evaluate learning-based wireless-network methods [7]. Deep learning and mathematical optimization have also been applied to wireless resource allocation, power control, scheduling, and network slicing [8–10]. These methods are important baselines, but they do not directly evaluate LLM-guided closed-loop parameter selection inside NS-3.

These works [3–7] advance simulation fidelity but do not systematically evaluate LLM-guided parameter selection between successive NS-3 simulation runs. This gap motivates the integration of AI-assisted reasoning directly within an iterative simulation loop.

### B. AI-Assisted Network Optimization

Artificial intelligence has been widely explored to improve network efficiency through learning-based parameter tuning. Reinforcement learning, deep neural methods, and mathematical optimization have been applied to routing, power control, resource allocation, and network slicing [8–10]. However, these methods often require training data, model engineering, or numerical surrogate models, which differs from the controller-mediated LLM proposal workflow evaluated here.

Recent studies and tools have begun connecting AI methods with network simulators, including NS-3 interfaces for AI algorithms and surveys of simulator-assisted reinforcement learning [5–7]. These efforts support the broader feasibility of simulation-guided AI workflows, while the present study focuses specifically on validated LLM-generated candidate configurations and an equal-budget Bayesian Optimization comparison.

Unlike conventional machine learning models [8–10], the proposed framework uses a pretrained LLM accessed through an API to interpret performance feedback and recommend parameter updates between successive NS-3 simulation runs. This reduces the need for data-intensive model training, while the resulting recommendations are still validated and bounded by the external Python controller.

### C. Large Language Models in Technical Domains

LLMs and GPT-series systems have demonstrated strong potential in technical reasoning, software engineering, telecom, and network-management tasks [11–15]. Recent work has explored LLMs in technical automation and network-related tasks; however, systematic evaluation of LLM-guided closed-loop parameter selection within NS-3 remains limited [13, 14, 18–20].

This work contributes (relative to [13, 14, 18–20]) an experimentally grounded NS-3 study of LLM-guided closed-loop parameter selection with deterministic controller-side safeguards. The claim is not novelty by

being first, but a reproducible evaluation of how such a workflow behaves under controlled wired and Wi-Fi simulation settings.

Recent research has explored the use of agentic LLM-based systems for scientific discovery and complex problem-solving, where models act as autonomous agents capable of generating hypotheses, executing computational steps, and iteratively refining solutions. For example, the SPARKS framework demonstrates a multi-agent system that performs scientific discovery tasks, including hypothesis generation, experimentation, and refinement [21]. Similarly, ProtAgents introduces a collaborative multi-agent LLM framework that integrates knowledge retrieval, simulation, and analysis to solve complex protein design problems [22].

These approaches reflect a broader trend toward using LLMs as active decision-making agents interacting with computational environments. In this context, the proposed framework can be viewed as a domain-specific instance of agentic AI, where the LLM interacts with the NS-3 simulation environment to iteratively analyze network performance and generate configuration updates. Unlike general-purpose agentic systems, this work focuses specifically on adaptive network optimization through closed-loop simulation feedback, emphasizing interpretability and parameter tuning within communication systems.

The proposed framework is also related to AutoML and hyperparameter optimization because it searches over configuration parameters through repeated evaluation. However, the objective in this work is not to claim superiority over all search algorithms. Instead, the contribution is to demonstrate an interpretable, low-training-cost control loop in which an LLM proposes candidate configurations from simulation feedback and a deterministic controller validates and applies them. This positioning clarifies the relationship between the proposed method, conventional heuristic tuning, and broader automated optimization methods. Table I summarizes the key differences among related works and clarifies how the present study is positioned relative to prior simulation, AI, and LLM-based network-optimization research.

Key gaps identified include:

- Limited integration between simulation and AI reasoning.
- Static configuration in most simulation frameworks.
- Heavy reliance on manual parameter tuning.
- Underutilization of LLMs for autonomous system control.

By integrating NS-3 with an API-accessed LLM through a validated controller, this research introduces an LLM-assisted optimization workflow for simulation-based network studies. The evaluation remains limited to the tested wired and IEEE 802.11 g Wi-Fi scenarios, selected bounds, User Datagram Protocol (UDP) traffic, static topology, and finite optimization budgets.

TABLE I. SUMMARIZES KEY DIFFERENCES AMONG RELATED WORKS AND POSITIONS THE CONTRIBUTION OF THIS STUDY

| Research Work | Simulation Tool | AI Technique | Automation | Adaptivity | LLM Integration |
|---|---|---|---|---|---|
| Yin *et al*. [6] | NS-3 (ns3-ai interface) | Generic AI/RL interface | Framework only | ✗ | ✗ |
| Ergun *et al*. [7] | N/A (survey) | Reinforcement Learning (surveyed) | ✓ | ✓ | ✗ |
| Zhou *et al*. [14] | N/A (survey) | LLM (surveyed) | Limited | ✗ | ✓ |
| Angi *et al*. [20] | Softwarized network devices | Small LM (intent-driven) | ✓ | Limited | ✓ |
| This Work | NS-3 + OpenAI API | LLM (gpt-4o-mini) | ✓ | ✓ | ✓ |

## III. METHODOLOGY

The proposed framework integrates the NS-3 network simulator with an LLM through a deterministic controller to support validated, closed-loop parameter selection. The system is composed of four main components: the simulation environment, the Python controller, the LLM interface, and the logging/validation layer. Together, these components allow the workflow to observe simulation results, request candidate configurations, validate them, and re-evaluate accepted settings.

### A. Overall Architecture

Fig. 1 illustrates the end-to-end workflow of the proposed adaptive optimization system. The process begins with NS-3 executing a network simulation—either wired or wireless—based on parameters specified in an external configuration file (config.yaml). After each run, NS-3 outputs Key Performance Indicators (KPIs) such as throughput (Mbps), latency (ms), and packet loss (%) to a results file (results.csv). These metrics are then processed by a Python-based controller, which acts as an interface between the simulator and the LLM.

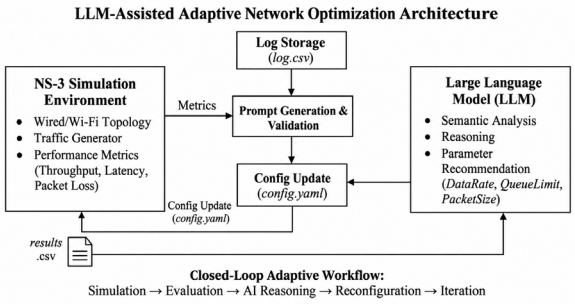



Fig. 1. System architecture of the LLM-assisted adaptive network.

The Python controller serves as the bridge between NS-3 and the LLM. It extracts throughput, latency, and packet-loss metrics, formats them into structured prompts, validates returned candidate parameters, and launches the next simulation only after parser, bounds, and duplicate checks. For Wi-Fi experiments, the tunable application-layer parameters are OfferedRatePerSta, QueueLimit, and PacketSize; the IEEE 802.11 g Physical Layer (PHY) mode remains fixed. Fig. 2 illustrates the adaptive feedback loop among the NS-3 simulator, Python controller, and Large Language Model (LLM), showing how simulation outputs are converted into prompts,

validated suggestions, and updated configuration files for the next iteration.

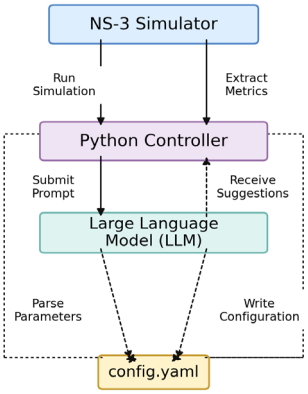



Fig. 2. Adaptive feedback loop illustrating the interaction between the NS-3 simulator, Python controller, and LLM between successive NS-3 simulation runs.

Algorithm 1 formalizes the proposed closed-loop optimization procedure, in which the Python controller coordinates NS-3 simulation runs, structures prompts for the LLM, validates and applies candidate configurations, and terminates upon exhausting the evaluation budget $N$.

**Algorithm 1:** Controller-Mediated LLM-Assisted Adaptive Optimization Loop

**Input:** NS-3 scenario $S$, initial configuration $C_0$, parameter bounds $B$, maximum evaluation budget $N$, and performance objective $U$
**Output:** Validated configuration trajectory $T$, performance metrics, and best-observed configuration $C^*$
$C \leftarrow C_0$; $T \leftarrow \emptyset$; $C^* \leftarrow C_0$
**for** $i \leftarrow 1$ **to** $N$ **do**
Execute NS-3 simulation using $C$
Extract throughput, packet loss, latency, and utility
Store $C$ and the measured performance metrics in $T$
Construct structured prompt using $C$, metrics, $B$, and $T$
Request candidate configuration $C'$ from the LLM/optimizer
Parse and validate $C'$
**if** $C'$ is malformed, out-of-bound, or duplicate **then**
Apply the documented retry procedure
**if** the retry fails **then**
$C' \leftarrow$ documented deterministic fallback configuration
**end if**
**end if**
Evaluate/update the accepted configuration according to the controller workflow
Update the best-observed configuration $C^*$
**end for**
**return** $T$, $C^*$

## B. Closed-Loop Adaptive Workflow

The adaptive workflow consists of the following stages:

(1) Initialization: The system loads an initial configuration file defining scenario-specific baseline parameters. In the wired experiment, these include link DataRate, QueueLimit, and PacketSize; in the Wi-Fi experiments, the IEEE 802.11 g PHY remains fixed and the controller tunes OfferedRatePerSta, QueueLimit, and PacketSize.
(2) Simulation Execution: NS-3 runs the simulation and records performance metrics in CSV format.
(3) Metric Extraction: The Python interface reads the results, ensuring data consistency and filtering incomplete or corrupted entries.
(4) AI-Based Analysis: The extracted metrics are sent to the LLM, which interprets performance trends and proposes optimized parameters in structured JavaScript Object Notation (JSON) or YAML Ain't Markup Language (YAML) format.
(5) Configuration Update: The system validates the syntax of the AI response, logs it for reproducibility, and updates the config.yaml file.
(6) Iteration: The process repeats for several cycles until performance convergence is reached or a predefined number of iterations is completed.

This design allows the simulation environment to evolve dynamically based on previous results rather than relying only on a fixed configuration. The framework therefore extends NS-3 from a static testbed into an adaptive simulation platform in which candidate parameter changes are proposed by the LLM and checked by the controller.

## C. Implementation Details

The system was implemented using Python 3.12 for the optimization controller and NS-3 v3.37 for network simulation. The OpenAI API served as the LLM interface, while the controller enforced deterministic validation of every candidate configuration.

The LLM was configured using low-variance parameters to improve reproducibility and consistency across runs. Specifically, the Phase-3 controller records gpt-4o-mini with temperature set to 0.2. All prompts, responses, accepted parameter updates, rejected malformed outputs, duplicate proposals, retries, and fallback-assisted evaluations were logged to support traceability and output-reliability analysis.

Prompt construction, response formatting, and output-handling details are provided in Section III-F to avoid repeating the same implementation description here.

Table II summarizes the initial configuration parameters for the wired demonstration; the reconstructed Wi-Fi parameters are reported separately in Section IV.

TABLE II. INITIAL NETWORK CONFIGURATION PARAMETERS

| Parameter | Value | Description |
|---|---|---|
| Wired DataRate | 50 Mbps initial | Initial wired link rate used in the previously validated wired demonstration. |

The baseline parameters and NS-3 configuration adopted for the Wi-Fi experiments are summarized in Table III (Section IV). During execution, the system automatically generated and maintained log.csv, capturing the iteration number, throughput, latency, and updated parameters. During execution, the system automatically generated and maintained log.csv, capturing the iteration number, throughput, latency, and the updated parameters. This log was later used for visualization and performance analysis. All communication between modules occurred

through lightweight text files to preserve reproducibility and minimize code dependencies.

### D. Advantages of the Framework

The architecture offers three main advantages:

- Interpretability: Each adaptation is traceable through logged prompts and configuration changes, allowing transparent analysis of AI reasoning.
- Scalability: The modular structure supports extension to multi-node or heterogeneous topologies, including Wi-Fi and IoT scenarios.
- Controller-mediated autonomy: No prior training is required, and every accepted LLM output has passed the validation and safeguard checks described in Section III.G.

### E. Wi-Fi and Controller Implementation

The implementation combines a C++ NS-3 Wi-Fi simulation script with the Python controller described above, supporting three execution modes: baseline execution, LLM-guided search, and Bayesian Optimization. For each evaluated configuration, the controller records the raw NS-3 CSV output, the prompt and response text, the parsed proposal, validation status, duplicate/retry/fallback behavior, the computed utility score, and the LLM API latency.

### F. Prompt Design and Documentation

For Wi-Fi optimization, each prompt reports the current iteration number, scenario, station count, current configuration, current measured metrics (throughput, latency, packet loss, utility), the complete compact history, the best configuration found so far, the allowed parameter bounds, and the fixed IEEE 802.11 g physical constraints. The LLM is instructed to return a structured JSON object containing OfferedRatePerSta, QueueLimit, PacketSize, and a reason_code. A representative Phase-3 proposal is: {“OfferedRatePerSta”: 7.5, “QueueLimit”: 100, “PacketSize”: 1300, “reason_code”: “explore_lower_load_and_queue”}.

Output variability was controlled through this low-temperature configuration and by repeating simulations under the same initial conditions. The accepted optimization trajectory showed low variation across repeated runs; nevertheless, the use of an external LLM introduces stochasticity and API-version dependence, which is identified as a limitation for deployment beyond offline iterative simulation studies.

### G. Validation and Safeguard Mechanism

The controller does not directly trust raw LLM text. It parses the JSON response, validates numeric fields, enforces the parameter bounds in Table IV, detects duplicate configurations, issues one retries for duplicates, and applies a documented deterministic fallback exploration point when needed. This design supports a controller-mediated autonomous workflow rather than an unconstrained LLM optimizer: outputs are accepted only after these parser, bounds, duplicate, retry, and fallback checks succeed.

## IV. Experimental Design and Setup

### A. Simulation Environment

The simulation environment (NS-3 v3.37, Python 3.12 controller, OpenAI API interface) is described in Section III (Methodology).

### B. Network Topologies

The wired LAN experiment is retained as a separate validated demonstration of closed-loop operation. The reconstructed Wi-Fi experiment uses the IEEE 802.11 g configuration and the one-AP, multi-STA topology summarized in Table III.

### C. Wi-Fi Baseline and Tunable Parameters

Table III summarizes the reconstructed Wi-Fi baseline configuration. Phase 3 uses the same physical setup while tuning only OfferedRatePerSta, QueueLimit, and PacketSize within the bounds listed in Table IV. Additional supplementary configuration details are provided in Table A1.

TABLE III. Reconstructed Wi-Fi Methodology Parameters

| Parameter | Value |
|---|---|
| NS-3 version | 3.37 |
| Wi-Fi standard | IEEE 802.11 g/WIFI_STANDARD_80211g |
| Rate manager | ns3: ConstantRateWifiManager |
| PHY data/control mode | ErpOfdmRate54Mbps |
| Traffic | UDP |
| Topology | One AP at origin; STAs on 5 m radius circle |
| Channel/propagation | YansWifiChannel; ConstantSpeedPropagationDelayModel; LogDistancePropagationLossModel |
| Baseline offered load | 20 Mbps per STA |
| Baseline packet/queue | 1024 bytes; WifiMacQueue MaxSize = 100 packets |
| Duration/measurement | 20 s simulation; 19 s measurement interval |
| Seed/runs | Seed 20260809; runs 1–5 |

TABLE IV. Phase-3 Search Bounds

| Parameter | Bounds |
|---|---|
| OfferedRatePerSta | 1.0–20.0 Mbps |
| QueueLimit | 20–500 packets |
| PacketSize | 512–1500 bytes |

### D. Evaluation Procedure

The evaluation is reported in three stages: reconstructed Wi-Fi baseline, initial LLM-guided Wi-Fi optimization, and controlled LLM-vs-Bayesian comparison. Phase 3 uses 20 evaluated configurations per optimizer per scenario, followed by five independent validation runs for each method’s best configuration. Baseline, optimization, and validation outputs are preserved separately.

## V. Results and Evaluation

### A. Wired Results

The previously validated wired experiment is retained as a separate demonstration of the NS-3-to-controller feedback loop. Wired results are not mixed with Wi-Fi packet-loss measurements because the physical models, traffic conditions, and bottlenecks are different.

### B. Reconstructed Wi-Fi Baseline

Using the reconstructed IEEE 802.11 g configuration summarized in Table III, the baseline results are reported in Table V. The severe packet loss and latency in Table V arise from an intentionally overloaded UDP scenario in which the aggregate offered load exceeds the effective shared-medium service capacity. These values should not be interpreted as typical IEEE 802.11g Wireless Local Area Network (WLAN) performance.

### C. Initial LLM-Guided Wi-Fi Optimization

Phase 2 used 15 LLM-guided iterations for both the 3-STA and 10-STA scenarios. The LLM repeatedly selected OfferedRatePerSta = 8.0 Mbps, QueueLimit = 150 packets, and PacketSize = 1200 bytes.

For 3 STA, the final run achieved 23.996 Mbps throughput, 0.015% packet loss, and 0.493 ms latency; the best observed values were 23.998 Mbps, 0.004% packet loss, and 0.483 ms latency.

For 10 STA, Phase 2 improved the overloaded baseline but did not adequately resolve congestion: the final run achieved 24.792 Mbps throughput, 69.004% packet loss, and 2538.221 ms latency, with best observed values of 24.876 Mbps, 68.886% packet loss, and 2522.969 ms latency. The repeated proposal indicates limited exploration at this initial optimization stage.

### D. Controlled LLM vs Bayesian Optimization

Phase 3 compared LLM-guided optimization and Bayesian Optimization under the same NS-3 Wi-Fi model, parameter bounds, evaluation budget, performance metrics, and deterministic utility function. Each optimizer received 20 NS-3 evaluations for 3 STA and 20 for 10 STA. The objective was fixed before observing the outcome:

$$\begin{aligned}\text{Utility} = 100\Big[&1.00\min\left(\frac{T}{T_{\text{ref}}}, 1.2\right) - 1.25\left(\frac{L_p}{100}\right)\\ &- 1.00\min\left(\frac{D}{D_{\text{ref}}}, 1.0\right)\\ &- 3.00\max\left(0, \frac{T_{\min} - T}{T_{\min}}\right)\Big]\end{aligned}$$

The tunable bounds were OfferedRatePerSta = 1.0–20.0 Mbps, QueueLimit = 20–500 packets, and PacketSize = 512–1500 bytes. The minimum useful aggregate-throughput constraints were 20 Mbps for 3 STA and 18 Mbps for 10 STA. The best configurations identified during the controlled Phase-3 search are summarized in Table VI.

TABLE V. RECONSTRUCTED WI-FI BASELINE

| Scenario | Aggregate Throughput | Packet Loss | Latency | Mean Per-STA Throughput |
|---|---|---|---|---|
| 3 STA | 25.216873 ± 0.024448 Mbps | 57.970019% ± 0.040749% | 4109.405677 ± 6.623224 ms | 8.405624 Mbps |
| 10 STA | 23.339353 ± 0.018004 Mbps | 88.327677% ± 0.009004% | 7065.379178 ± 9.747501 ms | 2.333935 Mbps |

TABLE VI. PHASE-3 BEST CONFIGURATIONS

| Scenario | Method | Offered Rate/STA | Queue | Packet Size | Throughput | Loss | Latency | Utility |
|---|---|---|---|---|---|---|---|---|
| 3 STA | LLM | 7.5 Mbps | 100 | 1300 bytes | 22.495 Mbps | 0.019 % | 0.282 ms | 89.176 |
| 3 STA | Bayesian | 9.472 Mbps | 224 | 1385 bytes | 28.411 Mbps | 0.012 % | 1.636 ms | 112.613 |
| 10 STA | LLM | 2.5 Mbps | 25 | 1500 bytes | 24.996 Mbps | 0.005 % | 0.429 ms | 107.086 |
| 10 STA | Bayesian | 3.0 Mbps | 40 | 1500 bytes | 27.198 Mbps | 9.324 % | 171.142 ms | 102.457 |

TABLE VII. FIVE-RUN VALIDATION

| Scenario | Method | Utility Mean ± SD | Throughput Mean ± SD | Loss Mean ± SD | Latency Mean ± SD |
|---|---|---|---|---|---|
| 3 STA | LLM | 89.200 ± 0.012 | 22.498 ± 0.001 Mbps | 0.008% ± 0.006% | 0.283 ± 0.0005 ms |
| 3 STA | Bayesian | 112.587 ± 0.011 | 28.410 ± 0.002 Mbps | 0.017% ± 0.006% | 2.255 ± 0.765 ms |
| 10 STA | LLM | 107.089 ± 0.003 | 24.996 ± 0.000 Mbps | 0.004% ± 0.001% | 0.424 ± 0.003 ms |
| 10 STA | Bayesian | 101.924 ± 0.268 | 27.151 ± 0.028 Mbps | 9.482% ± 0.093% | 180.562 ± 2.543 ms |

### E. Independent Five-Run Validation

After the search, each method's best configuration in each scenario was validated over five independent NS-3 runs. These validation runs did not influence either optimizer.

The independent five-run validation statistics for each selected configuration are summarized in Table VII.

The comparison is scenario-dependent. Bayesian Optimization achieved the higher validated utility in the 3-STA case (112.587 vs. 89.200) and higher throughput (28.410 Mbps vs. 22.498 Mbps). In the 10-STA case, the LLM achieved the higher validated utility (107.089 vs. 101.924) by selecting a lower-congestion operating point with 24.996 Mbps aggregate throughput, 0.004% packet loss, and 0.424 ms latency. Bayesian Optimization achieved higher 10-STA throughput, 27.151 Mbps, but with 9.482% packet loss and 180.562 ms latency.

The 10-STA optimization trajectories further explain this difference. Bayesian Optimization reached its strongest search result early, at iteration 4, with OfferedRatePerSta = 3.0 Mbps, QueueLimit = 40, PacketSize = 1500 bytes, utility = 102.457, throughput = 27.198 Mbps, packet loss = 9.324%, and latency = 171.142 ms. The LLM improved more gradually: its best-so-far utility increased at iterations 1, 2, 5, 7, and 11, and its strongest configuration appeared at iteration 11 with OfferedRatePerSta = 2.5 Mbps, QueueLimit = 25, PacketSize = 1500 bytes, utility = 107.086, throughput = 24.996 Mbps, packet loss = 0.005%, and latency = 0.429 ms.

The logs also show different exploration patterns. BO evaluated 20 unique 10-STA configurations without retry or fallback events and covered the offered-rate range from 1.010 to 19.509 Mbps. The LLM also produced 20 evaluated configurations after controller handling, but the

raw trajectory required 13 duplicate-proposal detections, 13 retries, and 17 fallback-assisted evaluations in the 10-STA case. Thus, the LLM did not converge faster; instead, controller safeguards converted repetitive or invalid suggestions into valid exploration points. The final performance advantage of the LLM in 10 STA is therefore best interpreted as the result of eventually selecting a more conservative low-load, small-queue configuration that strongly reduced packet loss and latency, rather than as evidence of faster or more exhaustive search.

### F. LLM Output Reliability and Controller Safeguards

Phase 3 also measured raw LLM output reliability. The workflow was autonomous at the controller-mediated system level: LLM-generated candidate configurations were parsed, checked for JSON validity, checked against bounds, checked for duplicates, retried when necessary, and replaced by deterministic fallback exploration points when proposals remained malformed or duplicate.

Table VIII reports the LLM-output reliability and controller-safeguard metrics collected during Phase 3.

TABLE VIII. LLM RELIABILITY

| Metric | Value |
|---|---|
| Total API calls | 57 |
| Valid JSON | 36 (63.16%) |
| Malformed outputs | 21 (36.84%) |
| Out-of-bound proposals | 0 (0.00%) |
| Duplicate proposals | 29 (50.88%) |
| Retries | 17 |
| Rejected proposals | 33 |
| Fallback-assisted evaluations | 33 |
| Manual interventions | 0 |
| Mean API latency | 1.418 s |

These results show that the raw LLM should not be described as a fully reliable standalone optimizer. Reliable operation required deterministic controller-side safeguards, although no manual intervention was required during the Phase 3 trajectory.

### G. Computational Overhead

Phase 3 measured the LLM API latency separately from NS-3 simulation execution and controller/file-processing overhead. The mean LLM API latency was 1.418 s across 57 API calls. NS-3 runtime varies by scenario and station count; because the new artifacts preserve per-evaluation outputs but do not provide a consolidated runtime table for every NS-3 call, no unsupported NS-3 runtime average is introduced here.

The computational overhead should therefore be interpreted as three separable components: NS-3 simulation runtime, LLM API latency, and local controller overhead for prompt generation, parsing, validation, CSV writing, and logging. Future work should report full wall-clock runtime distributions for each component across larger topologies.

## VI. DISCUSSION

The revised experiments correct earlier physically unsupported Wi-Fi claims by reconstructing the wireless setup as an explicit IEEE 802.11 g Network Simulator 3 (NS-3) model. The severe baseline loss and latency are not typical WLAN performance; they result from an intentionally overloaded User Datagram Protocol (UDP) scenario in which aggregate offered load exceeds the effective shared-medium service capacity.

The experiments show that LLM-guided optimization can reduce congestion dramatically by reasoning over application-layer offered load, queue capacity, and packet size. However, the controlled comparison does not establish universal LLM superiority. In the 3-STA scenario, Bayesian Optimization achieved the higher validated utility and higher throughput. In the 10-STA scenario, BO reached a strong-performing configuration earlier, at iteration 4, whereas the LLM reached its best configuration at iteration 11 after substantial controller-mediated duplicate and fallback handling. The LLM's final 10-STA advantage came from a more conservative operating point that sacrificed some throughput relative to BO but sharply reduced packet loss and latency.

LLM reliability remains a central limitation. Phase 3 produced malformed outputs and duplicate proposals at non-negligible rates, so deterministic parsing, validation, duplicate handling, retry logic, and fallback exploration were essential. The system achieved zero manual interventions during Phase 3, but that autonomy occurred at the controller-mediated workflow level rather than through an unsupervised raw LLM acting alone.

The evaluation remains limited to IEEE 802.11 g, 3-STA and 10-STA static topologies, UDP traffic, selected parameter bounds, one LLM/model configuration, and a limited optimization budget. Larger networks, heterogeneous traffic, mobility, additional wireless standards, and broader optimizer comparisons remain future work.

## VII. CONCLUSION

This study presented an LLM-assisted closed-loop optimization framework that integrates NS-3 with a validated controller and an API-accessed large language model. The architecture successfully supports simulation execution, metric extraction, LLM-generated candidate proposals, deterministic validation, and iterative re-evaluation.

The reconstructed IEEE 802.11g experiments show that useful congestion-reducing configurations can be found through LLM-guided parameter selection, but the results are scenario-dependent. Bayesian Optimization was stronger under the fixed utility in the 3-STA validation, whereas LLM-guided optimization achieved the higher utility in the 10-STA validation by producing a low-loss, low-latency operating point.

The results support the value of controller-mediated LLM assistance for network simulation workflows, not a claim of universally scalable autonomous network optimization. Reliable operation required parser, bounds, duplicate, retry, and fallback safeguards. Further evaluation is required before drawing conclusions about real-world deployment or larger-scale heterogeneous networks.

## APPENDIX: EXPERIMENTAL CONFIGURATION AND OPTIMIZATION WORKFLOW

Table A1 summarizes the principal configuration settings used in the revised experimental evaluation.

TABLE A1. SUMMARY OF EXPERIMENTAL CONFIGURATION

| Parameter | Value |
|---|---|
| NS-3 version | 3.37 |
| Wired experiment | Previously validated point-to-point closed-loop demonstration |
| Wi-Fi topology | One AP plus 3 or 10 STA nodes |
| Wi-Fi standard | IEEE 802.11g / WIFI_STANDARD_80211g |
| Baseline offered load | 20 Mbps per STA |
| Phase-3 tunables | OfferedRatePerSta, QueueLimit, PacketSize |
| Phase-3 budget | 20 evaluations per optimizer per scenario plus 5-run validation of each best configuration |

## DATA AVAILABILITY STATEMENT

The NS-3 scripts, Python controllers, configuration files, and experiment logs used in this study are publicly available at: https://github.com/aaashehri/LLM-NS3-Optimization.

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## AUTHOR CONTRIBUTIONS

Aziz Alshehri contributed to conceptualization, methodology, software implementation, validation, formal analysis, and writing of the original draft. Jebreel Omar Alamri contributed to methodology, supervision, validation, and manuscript review and editing. All authors have read and approved the final version of the manuscript.

## FUNDING

This research work was funded by Umm Al-Qura University, Saudi Arabia under grant number: 26UQU4300013GSSR01.

## REFERENCES

[1] M. Z. Chowdhury, M. Shahjalal, S. Ahmed, and Y. M. Jang, "6G wireless communication systems: Applications, requirements, technologies, challenges, and research directions," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 957–975, 2020. doi: 10.1109/OJCOMS.2020.3010270

[2] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019. doi: 10.1109/MCOM.2019.1900271

[3] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network Simulations with the ns-3 Simulator," *SIGCOMM demonstration*, vol. 14, no. 14, 527, 2008.

[4] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Proc. Modeling and Tools for Network Simulation*, Berlin, Germany: Springer, 2010, pp. 15–34.

[5] L. Campanile, M. Gribaudo, M. Iacono, F. Marulli, and M. Mastroianni, "Computer Network Simulation with ns-3: A systematic literature review," *Electronics*, vol. 9, no. 2, 272, 2020. doi: 10.3390/electronics9020272

[6] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, "ns3-AI: Fostering artificial intelligence algorithms for networking research," in *Proc. 2020 Workshop on ns-3*, 2020, pp. 57–64. doi: 10.1145/3389400.3389404

[7] S. Ergun, I. Sammour, and G. Chalhoub, "A survey on how network simulators serve reinforcement learning in wireless networks," *Computer Networks*, vol. 234, 109934, 2023. doi: 10.1016/j.comnet.2023.109934

[8] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2595–2621, 2018. doi: 10.1109/COMST.2018.2846401

[9] Y.-F. Liu, T.-H. Chang, M. Hong, Z. Wu, A. M.-C. So, E. A. Jorswieck, and W. Yu, "A survey of recent advances in optimization methods for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 11, pp. 2992–3031, 2024. doi: 10.1109/JSAC.2024.3443759

[10] J. J. A. Esteves, A. Boubendir, F. Guillemin, and P. Sens, "A heuristically assisted deep reinforcement learning approach for network slice placement," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4794–4806, 2022. doi: 10.1109/TNSM.2021.3132103

[11] X. Hou *et al*., "Large language models for software engineering: A systematic literature review," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, pp. 1–79, 2024. doi: 10.1145/3695988

[12] J. Achiam *et al*., "GPT-4 technical report," arXiv preprint, arXiv:2303.08774, 2023.

[13] F. Liu, B. Farkiani, and P. Crowley, "Large Language Models for Computer Networking Operations and Management: A Survey on Applications, Key Techniques, and Opportunities," *Computer Networks*, vol. 271, 111614, 2025. doi: 10.1016/j.comnet.2025.111614

[14] H. Zhou *et al*., "Large Language Model (LLM) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 3, pp. 1955–2005 2024. doi: 10.1109/COMST.2024.3465447

[15] S. Hong *et al*., "A Comprehensive Survey on LLM-Based Network Management and Operations," *International Journal of Network Management*, vol. 35, no. 6, e70029, 2025. doi: 10.1002/nem.70029

[16] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, "Algorithms for hyper-parameter optimization," *Advances in Neural Information Processing Systems*, vol. 24, pp. 2546–2554, 2011.

[17] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," *Advances in Neural Information Processing Systems*, vol. 25, pp. 2951–2959, 2012.

[18] H. Luo *et al*., "AI reasoning for wireless communications and networking: A survey and perspectives," *ACM Computing Surveys*, vol. 58, no. 13, pp. 1–38, 2026. doi: 10.1145/3811822

[19] H. Zou *et al*., "Large language models in 6g from standard to on-device networks," *Nature Reviews Electrical Engineering*, vol. 3, no. 2, pp. 123–134, 2026.

[20] A. Angi, A. Sacco, and G. Marchetto, "LLNet: An intent-driven approach to instructing softwarized network devices using a small language model," *IEEE Transactions on Network and Service Management*, vol. 22, no. 4, pp. 3403–3418, 2025. doi: 10.1109/TNSM.2025.3570017

[21] A. Ghafarollahi and M. J. Buehler, "Sparks: Multi-agent artificial intelligence model discovers protein design principles," arXiv preprint, arXiv:2504.19017, 2025.

[22] A. Ghafarollahi and M. J. Buehler, "ProtAgents: Protein discovery via large language model multi-agent collaborations combining physics and machine learning," *Digital Discovery*, vol. 3, pp. 1389–1409, 2024. doi: 10.1039/D4DD00013G

Copyright © 2026 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).