POLITECNICO DI TORINO
Repository ISTITUZIONALE

LLM-Driven Optical Network-as-a-Service (ONaaS) for Next Generation Metro-Access Convergence

Original
LLM-Driven Optical Network-as-a-Service (ONaaS) for Next Generation Metro-Access Convergence / Dipto, I.C., Zeb, S., Masood, M.U., Khan, I., Costa, N., Pedro, J., Napoli, A., Curri, V.. - (2026). (2026 IEEE International Conference on Machine Learning for Communication and Networking Abu Dhabi (UAE) 8–11 June 2026).

Availability:
This version is available at: 11583/3015690 since: 2026-10-20T09:58:06Z

Publisher:
IEEE

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

10 November 2026

LLM-Driven Optical Network-as-a-Service (ONaaS) for Next Generation Metro-Access Convergence

Imran Chowdhury Dipto
Politecnico di Torino, Italy
imran.dipto@polito.it

Sanwal Zeb
Politecnico di Torino, Italy
sanwal.zeb@polito.it

Muhammad Umar Masood
Politecnico di Torino, Italy
muhammad.masood@polito.it

Ihtesham Khan
Nokia Bell Labs, Murray Hill, US
ihtesham.khan@nokia-bell-labs.com

Nelson Costa
Infinera, Lisbon, Portugal
nelson.costa@nokia.com

João Pedro
Nokia Optical Networks, Carnaxide, Portugal
joao.pedro@nokia.com

Antonio Napoli
Nokia Optical Networks, Munich, Germany
antonio.napoli@nokia.com

Vittorio Curri
Politecnico di Torino, Italy
vittorio.curri@polito.it

***Abstract*—The convergence of metro and access domains and the shift towards intent-driven, autonomous optical infrastructures are accelerating the demand for intelligent control systems. This paper investigates the feasibility of deploying Large Language Models (LLMs) as agents to support an Optical-Network-as-a-Service (ONaaS) paradigm for next-generation metro–access convergence. We introduce an LLM-driven framework that integrates natural-language intent interpretation, topology generation, and network equipment allocation into a unified agent architecture. Eight state-of-the-art open-source LLMs, including LLaMA-3, Mistral-7b, Gemma-2/3, and Qwen-2.5, are benchmarked under Zero-Shot, One-Shot, and Few-Shot prompting strategies using a standardized evaluation criterion based on tool-call validity, hallucination rate, and inference latency. The results show that example-driven prompting improves reliability for most models, with the strongest and most consistent gains observed for Mistral-7b and LLaMA-3-8b, while some families (e.g., Gemma and Qwen variants) showed limited or mixed improvements under Few-Shot prompts. Mistral-7b achieved the highest success rate, however, with higher inference latency, whereas LLaMA-3-8b offers the best success-to-latency trade-off. Overall, our ONaaS scenarios demonstrate that open-source LLMs can robustly orchestrate network planning functions from natural-language intents, paving the way for the proposed architecture to integrate designed impairment-aware modules in future extensions, thereby enabling tighter integration with digital-twin environments and impairment-aware planning workflows.**

***Index Terms*—Large Language Model, Prompting, Metro-Access convergence, ONaaS, Network Design**

I. Introduction

The rapid growth of data-intensive and AI-driven applications, together with 5G and 6G services, are increasing the performance and scalability requirements of converged metro–access optical networks. These networks must support large traffic growth, ultra-low-latency services, dynamic network slicing, integrated fronthaul/backhaul/midhaul (xHaul) transport, and the rapid expansion of edge computing [1], [2]. As operators transition from static, manually engineered infrastructures to highly dynamic service-level agreement-aware provisioning, network design and planning processes are becoming significantly more complex. Traditional offline tools and static configuration workflows struggle to capture heterogeneous constraints such as evolving topology structures, cost models, and multi-service coexistence. These challenges highlight the need for advanced automation frameworks capable of efficiently generating, analysing, and adapting future optical network topologies.

To address these demands, the optical transport layer is transitioning from static configuration toward fully automated networking environments. In parallel, the traditional separation between metro and access segments is progressively fading due to the adoption of high-rate coherent optics and flexible wavelength routing [3], [4]. This convergence increases control-plane complexity as networks become more heterogeneous and impairment awareness must extend across both domains. Consequently, operators are moving toward the Optical Network-as-a-Service (ONaaS) paradigm, in which optical connectivity is provisioned dynamically through intent-based abstractions rather than manual configuration [5], [6]. ONaaS environments require intelligent orchestration systems capable of translating high-level service intents into structured workflows involving topology selection, resource allocation, and configuration actions across multiple tools and control interfaces. However, conventional SDN/NFV controllers primarily execute predefined logic and cannot typically interpret ambiguous or incomplete user intents, resolve missing parameters, or coordinate multi-step planning interactions in real time.

Recent advances in Large Language Models (LLMs) provide a potential mechanism to address this orchestration gap. LLMs have demonstrated strong capabilities in natural-language understanding, context reasoning, and structured tool usage, enabling them to translate high-level operator requests into executable workflows that combine multiple domain-specific functions [7]. In our context, these models can assist with intent interpretation, parameter completion, and workflow

sequencing while relying on deterministic back-end modules for accurate computation. Early studies such as OptiComm-GPT have shown that LLMs can guide physical-layer simulation and multi-stage decision processes within optical communication environments [8]. However, most existing work remains limited to isolated link-level or configuration tasks, and the feasibility of employing LLMs as orchestration agents for the converged metro–access network design workflows is largely unexplored.

In this work, we propose an LLM-driven ONaaS framework in which the LLM operates as an intent-to-workflow agent that interprets high-level user requests and coordinates domain-specific Python modules responsible for deterministic network design functions such as topology generation, equipment placement, and path analysis. The objective of this study is to investigate how open-source LLMs can reliably mediate between natural-language intents and existing network engineering tools. To assess the effectiveness of current models in this orchestration role, we evaluate several state-of-the-art open-source LLM families, including LLaMA-3, Mistral-7B, Gemma-2/3, and Qwen-2.5, under Zero-Shot, One-Shot, and Few-Shot prompting strategies using a unified benchmarking framework where we gave the models the tasks of network designing, path calculations and equipment deployments.

The remainder of this paper is organized as follows. Section II provides an overview of LLM architectures and the selected open-source models. Section III presents the design of the proposed LLM-driven ONaaS framework and benchmarking pipeline. Section IV reports the experimental results across different prompting strategies and analyzes the observed trade-offs. Finally, Section V concludes the paper and outlines future research directions.

II. Large Language Models

LLMs are a type of AI model capable of general-purpose reasoning. They can understand natural human language to generate responses and can also understand code and run it [9]. Unlike traditional ML models trained for a fixed task, LLMs can understand different domains and run chain-of-thought reasoning, making them ideal for applications that require structured decision-making [10]. This logical understanding capability is crucial for our ONaaS architecture, where an LLM acts as the core agent mediating between user intents and Python modules that implement network modelling and equipment deployment decisions.

A. Types of Open-Source LLMs

LLMs are built upon the transformer architecture introduced in [11], where the self-attention mechanism was proposed. This method allows tokens in an input sequence to capture the relevance of all other tokens allowing the model to capture long-range dependencies, contextual meaning and structured relationships. Additional variants, such as multi-head attention and grouped-query attention (GQA), further improved the scalability of these models.

In particular, these models operate on tokens, which are subword units generated by a tokenizer. Instead of processing the whole network queries, these models encode text into a vocabulary of frequent word fragments [12], [13]. This process makes the models more robust to linguistic variation and enables efficient representation of technical expressions and network-specific terminology critical for optical network tasks.

1) LLaMA-3 (8b): This Meta's LLaMA-3 series model introduces improved tokenizer efficiency, improved attention mechanism and refined instruction tuning. The 8b version offers a strong balance between reasoning ability and computational efficiency [14], potentially making it ideal for network automation tasks.

2) Mistral-7b: The Mistral(M)-7b version uses the GQA type of the attention mechanism coupled with sliding-window attention and optimized training setups. It is known for its robust few-shot performance and reduced hallucination rates [15]. The model's architecture makes it effective in network tasks like query reasoning and executing valid Python method calls.

3) Gemma-2 (7b) and Gemma-3 (4b): This is Google's Gemma family built from the wider Gemini architecture and designed for lightweight, high-throughput inference. The 4b and 7b versions provide high decoding speed with compact memory usage [16]. This characteristic makes them ideal for latency-constrained ONaaS deployment.

4) Qwen-2.5 (7b and 14b): Qwen(Q)-2.5 is Alibaba's LLM model designed with high context length (token size up to 32k), strong language understanding capabilities, and alignment for instruction following. The 14b version offers higher accuracy, but having 14 billion parameters provides higher accuracy with the cost of higher use of computational resources compared to the 7b variant of the model [17]. For this experiment, this model helps in understanding long network questions.

B. Prompting

Prompting is the process of guiding an LLM using human language or examples so that it can produce accurate and task-aligned outputs. This is important because LLMs are highly sensitive to input structure and the quality and style of a prompt can greatly impact the models' performance, reasoning depth, and hallucination tendencies [18]. In our investigation, we have used three widely used prompting paradigms to assess the LLM models for robustness.

Figure 1 illustrates the prompting workflow used in this study. The user submits a network query through the application and selects one of three prompting regimes – Zero-Shot (ZS), One-Shot (OS), or Few-Shot (FS) – each mapped to a corresponding back-end prompt template. In the ZS mode, the LLM receives only a minimal system instruction describing the available required function-call format; no examples are provided, so the model must rely solely on its pre-trained knowledge to infer the correct Python command. In the OS mode, the system instruction is supplemented with a single demonstration showing how a representative network intent should

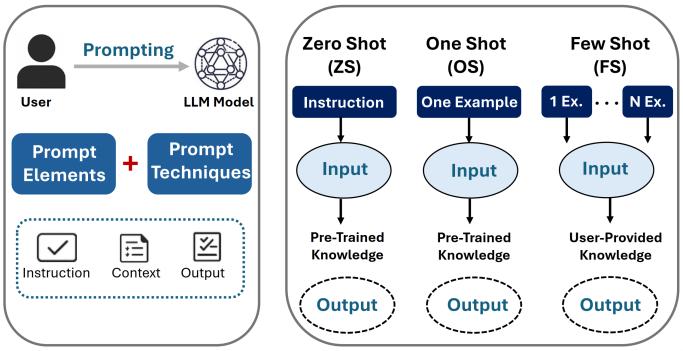


Fig. 1: Prompting techniques [8]

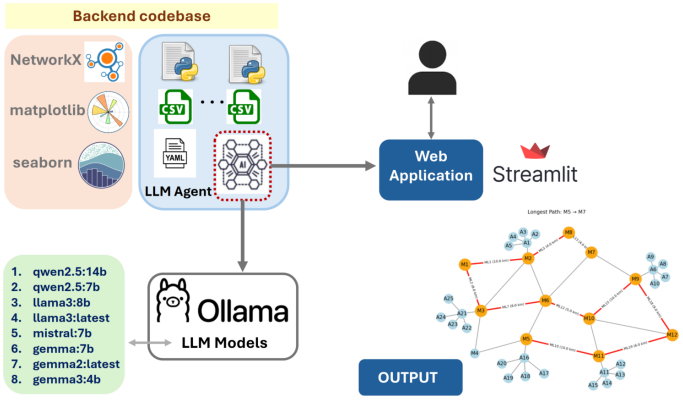


Fig. 2: LLM based ONaaS design framework

be converted into a valid tool call, enabling limited in-context learning. In the FS mode, the instruction is further enriched with multiple curated examples spanning topology generation, equipment deployment, and QoT analysis, providing the model with a structured pattern from which it can generalize more reliably. Once the selected prompt template and user query are combined, the resulting message is passed to the LLM, which produces a deterministic Python function call executed by the back-end interpreter. This structured prompting mirrors prior evaluation techniques such as FEACI [19], enabling systematic study of contextual reasoning improvements across prompting modes. In this paper, we evaluate these open-source LLM models having a varying range of transformer designs, scaling behaviours and alignment strategies. The selection aims to capture the trade-offs between model size, inference speed, hallucination rate, and computational resource utilization.

- Zero-shot (ZS) prompting: The model is given only the system directive specifying the allowable functions in the codebase and must infer the correct mapping purely from the query.
- One-shot (OS) prompting: Adds a single example demonstrating how a natural query is mapped to a valid function call. This enables minimal context learning while preserving generality.
- Few-shot (FS) prompting: Introduces multiple example "seeds" covering diverse network reasoning tasks, topology generation, and path queries to enhance contextual learning and reduce prompt ambiguity.

III. Methodology

This section presents the overall design and workflow of the proposed LLM-driven Network Assistant which is capable of designing network and compute path metrics based on user questions. We describe the framework we are using to evaluate and test the LLMS based system. The framework integrates a domain-specific graph modelling, topology generation, and equipment deployment functionalities into a combined environment supporting the different prompting tasks: Zero-Shot, One-Shot, and Few-Shot prompting strategies. Fig. 2 shows the overall representation of the framework for ONaaS design and planning.

A. System Overview

The framework follows an Agent-Oriented Design, where an LLM model acts as the core reasoning agent interpreting a group of specialized functional Python modules stored in the backend codebase. Here, the NetworkX library is used for designing and path calculations, and matplotlib and seaborn are used for visualizations. Inside the codebase, the two CSV files provide the link, latitude, and longitude information to the LLM models for plotting. Each of these models abstracts a real network operation, such as topology generation and equipment deployment. The user interface layer is implemented in Streamlit, providing interactive access for querying models and visualizing results. The LLM Agent file is the central controller that manages the model selection, message construction, prompt injection, and function dispatch for the LLM models and establishes the connection with Ollama to call the models to interpret the created code base. All the selected models were run locally using `Ollama` software for reproducibility and uniform inference control.

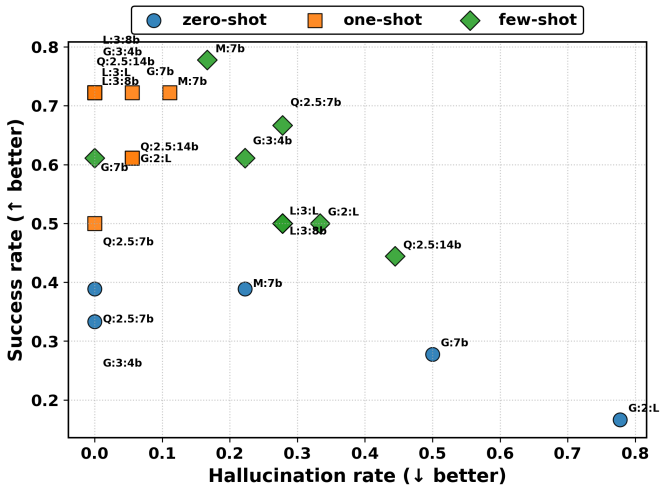


Fig. 3: Success and Hallucination rates

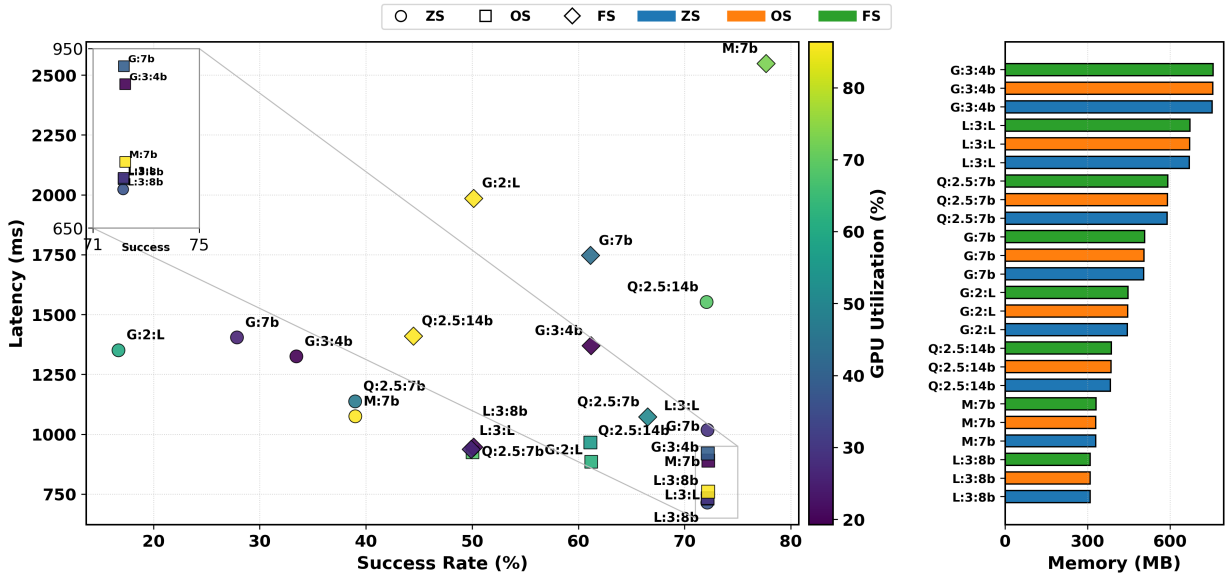


Fig. 4: Success vs Resource utilization

B. Evaluation and Benchmarking Framework

We evaluated the models using a standardized evaluation metric for all models and prompting modes. Benchmark queries are drawn from a separate YAML file. Each benchmark run captures the following parameters:

$$\text{Success Rate} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\text{valid}(i)],$$
$$\text{Hallucination Rate} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\text{hallucinated}(i)], \quad (1)$$
$$\text{Average Latency} = \frac{1}{N}\sum_{i=1}^{N} t_i,$$

where N shows the total number of evaluated commands and $\mathbf{1}[\cdot]$ is the indicator function that returns 1 when the condition is true and 0 otherwise. valid(i) indicates that the command i produced a correct and executable function call, while hallucinated(i) denotes that the model generated an incorrect function call.

IV. Performance and Evaluation

The performance analysis of the LLMs focuses on 4 primary performance metrics: (i) success rate, (ii) hallucination rate, (iii) inference latency, and the overall success vs hallucination and success vs latency trade-offs. The results highlight key behavioral differences among the 8 LLM models under realistic network-engineering tasks.

Fig. 3 illustrates the joint distribution of success rate and hallucination rate for all evaluated models under the three prompting regimes. The results reveal a clear separation in model behaviour. In the zero-shot setting (blue markers), the models exhibit substantial variability, with several points concentrated in the lower accuracy region and showing hallucination rates up to 0.80. This confirms that, without prior examples, most models struggle to consistently generate valid tool calls for network-domain tasks.

Introducing a single demonstration through One-Shot prompting (orange markers) generally stabilizes the outputs for most models, shifting several points toward higher success and reduced hallucination. However, the effect of Few-Shot prompting (green markers) is not uniform across all model families. While Mistral-7b and LLaMA-3-8b show clear improvements and move closer to the desirable upper-left region, some models, particularly the Gemma and Qwen variants, exhibit mixed or limited gains compared to One-Shot prompting. In these cases, the longer and more structured prompts may introduce format sensitivity or example overfitting, leading to occasional degradation in function-call accuracy. This observation indicates that Few-Shot prompting does not universally improve reliability and that model alignment and instruction-following behaviour strongly influence the benefit obtained from additional contextual examples.

Among the evaluated models, LLaMA3:8b (L:3:8b) and Mistral-7b (M:7b) achieve the most favourable positions, approaching the Pareto-optimal boundary with success rates exceeding 0.75 and hallucination rates below 0.18 in the few-shot scenario. Qwen2.5:14b (Q:2.5:14b) also benefits significantly from example-based prompting but remains slightly down-shifted due to occasional overgeneralization in argument construction. In contrast, the compact Gemma (G) models remain clustered in the lower half of the plot across all regimes, indicating that their efficiency comes at the cost of reduced adherence to the domain-specific tool-use format.

Building on the reliability trends observed in the success–hallucination plot, Fig. 4 extends the evaluation by incorporating the computational dimension of model behaviour. While the previous analysis established how prompting structures influence correctness and stability, this figure examines how these gains translate in terms of inference latency, GPU

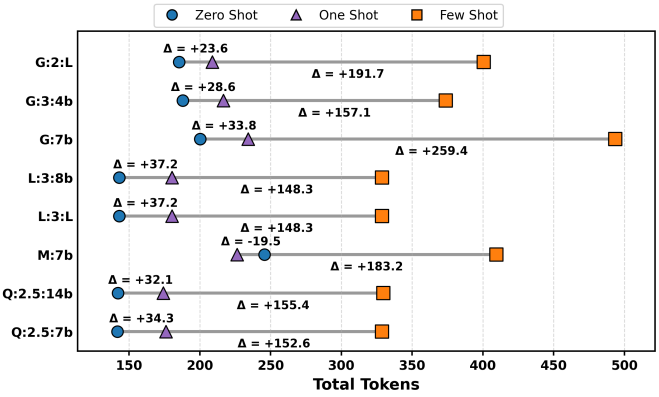


Fig. 5: LLMs Token usage

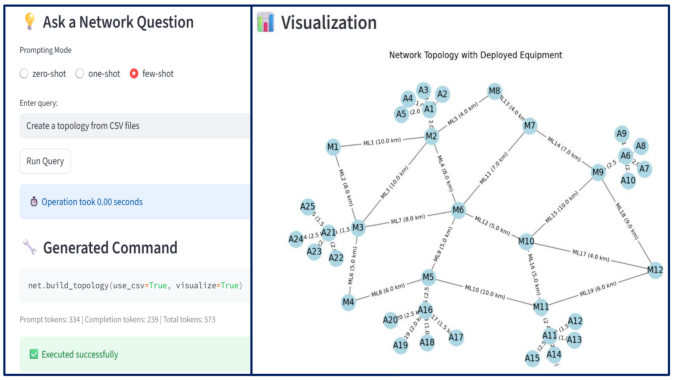
utilization, and memory footprint, three critical factors for deploying ONaaS reasoning agents in practical environments.

The scatter plot relates success rate to latency, with marker color representing GPU load. A consistent pattern emerges: models that benefited most from few-shot prompting in the earlier analysis generally maintain higher success rates here as well. However, an important distinction becomes visible in their runtime behaviour. While L:3:8b (low GPU utilization) and Q:2.5:14b (moderate GPU utilization) cluster in a favourable region of high accuracy and 0.8-1.55 sec latencies, M:7b (moderate GPU utilization), despite being among the most accurate models, consistently exhibits the highest inference times, reaching up to 2.3–2.5 s in the few-shot regime. This reflects the heavier intermediate computation introduced by its grouped-query attention mechanism, making it the slowest model in the comparison even when producing highly reliable tool calls.

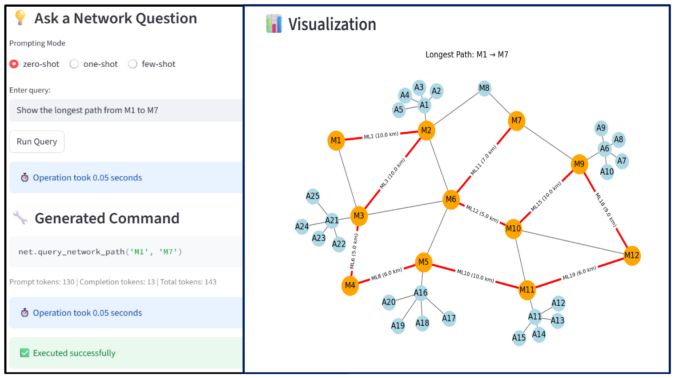
The Gemma models, despite their compact parameter counts, consistently exhibit higher memory consumption than most other models in the evaluation, yet their performance remains confined to the mid-accuracy and moderate-latency region. This behaviour reinforces the previous observation that architectural compactness does not necessarily translate into efficient tool-following capability, especially for domain-specific function-call reasoning. The memory–utilization bars on the right further highlight this discrepancy: whereas the top-performing models such as L:3:8b, M:7b, and Q:2.5:14b fall within a balanced low to mid-memory tier (≈300–500 MB), the Gemma variants often occupy the upper range of memory usage while still delivering moderate-to-low success rates.

Overall, this plot shows that improvements in reliability do not uniformly translate to faster inference. In particular, M:7b offers strong accuracy but at the cost of the highest latency, whereas L:3:8b strikes the best balance between correctness and runtime efficiency. This reinforces that prompting structure and architectural design must be jointly considered when selecting LLMs for real-time ONaaS applications. Given these observations and considering the accuracy–latency trade-offs across all models, the subsequent analysis of example ONaaS queries is carried out using M:7b as the representative model.

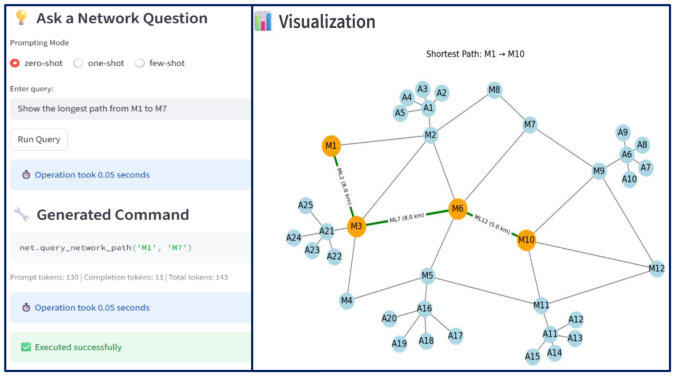
Fig. 5 shows the usage of tokens of all the models for the



(a) Create Metro–Access topology.



(b) Longest path from M1→M7.



(c) Shortest path from M1→M10.

Fig. 6: LLM-assisted Metro-Access network design

prompting settings used in this experience. From this graph we can observe that Mistral-7b generates more tokens in the ZS prompt setup compared to the OS prompt. When no examples are given, the model produces additional explanations before generating a function call, which increases sequence length and contributes to higher latency. In contrast, models like LLaMA-3 and Qwen-2.5 already follow the expected function call structure in ZS mode, producing shorter outputs, demonstrating the rise in token usage from Zero-One-Few-shot driven mainly by prompt length. These results show that token efficiency is largely influenced by model alignment and the decoding nature of these LLM models with regard to inference performance in the ONaaS workflow, and token-efficient models like LLaMA-3 might be more suitable for

latency-strict constraints.

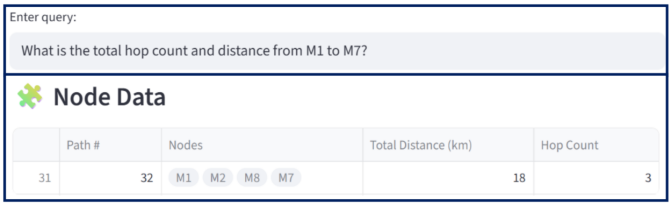


Fig. 7: LLM-assisted hop count and distance for path M1→M7.

Fig. 6 shows the visual representation of how queries are executed by the Mistral-7b model under the few-shot prompt environment. Fig. 6(a) shows that when the user Queries the network to generate the network topology from the CSV files the LLM model executes the "net.build_topology()" function with input parameters as use_csv = True and sets the *visualize=True* which is the correct function call based on the query. Then the topology is correctly generated and shown on the screen to the end user. We have also tested the model with a different prompting paradigm, as can be seen in Fig. 6(b & c). We chose this setup to challenge the model to test its success rate under a zero-shot setup by asking the model to display the longest and shortest paths from different sources and destinations. In this case, the model again executed the correct Python function method and displayed the correct results. This shows the strong understanding capability of the Mistral-7b model as it was able to understand the context of the user even without any examples.

The Mistral-7b was tested with a more difficult network query under the ZS condition, where we queried this model to show the total number of nodes, hop count, and distance between metro nodes M1 and M7. Despite having ZS prompts in the backend with no examples to map, it again demonstrated an accurate Python method call to retrieve the correct information as shown in Fig. 7. This further verifies the strong success rate with low hallucination of this model and its potential to be deployed in real-time Digital-twin-based network management systems.

V. Conclusion and Future Work

In this paper, we propose an LLM-driven ONaaS framework for converged metro-access networks by integrating intent interpretation, topology generation, and equipment deployment within a unified agent-based architecture. We benchmarked eight state-of-the-art open-source LLMs across ZS, OS, and FS prompting strategies to assess their capability for autonomous reasoning in optical network design. The results demonstrate that LLMs are feasible candidates for automating design and planning tasks. FS prompting consistently improved reliability across most models by providing structured examples that reduced hallucinations and increased functional correctness. Among the evaluated models, Mistral (7b) achieved the highest accuracy, albeit with higher latency and moderate resource utilization, while compact models such as Gemma-2 and Gemma-3 were more efficient but required stronger prompt guidance to reliably generate valid function calls.

Future work will focus on strengthening LLM reliability through reinforcement learning, tool-integrated fine-tuning, and constrained decoding. We will extend the evaluation framework towards multi-step reasoning, intent negotiation, and cross-layer decision making. Integration with GNPy, an open-source Python-based library for planning and simulating optical networks, will further improve adaptation to operator environments. Finally, scaling the framework to larger topologies and real traffic conditions will enable more rigorous validation of the LLM-driven ONaaS.

Acknowledgments

Funded by the European Union's Horizon Europe research and innovation programme under project NESTOR G.A. No. 101119983 and ALLEGRO G.A. No. 101092766, in collaboration with Nokia Bell Labs (New Jersey, US), Nokia Optical Networks (Carnaxide, Portugal & Munich, Germany).

References

[1] J. G. A. et al., "What Will 5G Be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[2] S. V. et al., "Toward 6G Networks: A Survey on Enabling Technologies," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 3341–3379, 2020.

[3] G. Z. et al., "Toward Converged Optical Network Architectures for 5G," *Journal of Optical Communications and Networking*, vol. 12, no. 2, pp. A1–A12, 2020.

[4] Z. e. a. Zhu, "Converged Optical Access and Metro Networks: A Survey," *Journal of Lightwave Technology*, vol. 40, no. 10, pp. 3281–3297, 2022.

[5] N. B. Labs, "Optical Network as a Service: Concepts and Implementations," 2021, white Paper.

[6] S. Zeb, A. Ali, I. C. Dipto, A. Rosso, M. U. Masood, R. Schips, R. Ambrosone, S. Straullu, F. Aquilino, A. Nespola, J. Pedro, A. Napoli, A. Galardini, and V. Curri, "AI-driven converged metro-access Optical Network-as-a-Service with point-to-multipoint coherent optics for 6G X-hauling," *Computer Networks*, vol. 284, p. 112338, 2026.

[7] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, vol. 1, no. 2, 2023.

[8] X. Jiang, M. Zhang, Y. Song, Y. Zhang, Y. Wang, C. Ju, and D. Wang, "OptiComm-GPT: a GPT-based versatile research assistant for optical fiber communication systems," *Optics Express*, vol. 32, no. 12, pp. 20 776–20 796, 2024.

[9] L. Ouyang *et al.*, "Training Language Models to Follow Instructions," *arXiv preprint arXiv:2203.02155*, 2022.

[10] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[11] A. Vaswani *et al.*, "Attention is All You Need," *Advances in Neural Information Processing Systems*, 2017.

[12] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *ACL*, 2016.

[13] T. Kudo and J. Richardson, "Sentencepiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing," in *EMNLP*, 2018.

[14] H. Touvron *et al.*, "LLaMA 3: Open and Efficient Foundation Language Models," *arXiv preprint arXiv:2407.00088*, 2024.

[15] A. Jiang, L. Lambert, and Y. Jernite, "Mistral 7B: Efficient Open-Weight Language Models," *arXiv preprint arXiv:2310.06825*, 2024.

[16] G. DeepMind, "Gemma and Gemma 2: Open Models Based on Gemini Research," *arXiv preprint arXiv:2407.13850*, 2024.

[17] J. Bai, T. Chen *et al.*, "Qwen2.5: Scaling Open Language Models with Multilingual and Multi-Modal Reasoning," *arXiv preprint arXiv:2409.12191*, 2024.

[18] P. Liu *et al.*, "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *ACM Computing Surveys*, 2023.

[19] R. Khan, M. Santoro, and L. Zhao, "Benchmarking Open and Closed-Source LLMs for 5G/6G Network Configuration Using FEACI," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2024.