NETLLMBENCH: A Benchmark Framework for Large Language Models in Network Configuration Tasks

Kaan Aykurt*, Andreas Blenk†, Wolfgang Kellerer*
*Chair of Communication Networks, Technical University of Munich, Germany
†Siemens AG, Munich, Germany

***Abstract*—Traditional network management techniques often struggle with the scale and dynamism of modern networks, requiring significant human oversight and being prone to high error rates. Large Language Models (LLMs) present a promising alternative to conventional approaches by automating network configuration and management. However, a systematic way to evaluate their performance is lacking in the literature.**

This paper introduces NETLLMBENCH, a novel framework designed to rigorously assess the performance of LLMs in managing computer networks. By integrating prompt engineering and network emulation in a closed loop, NETLLMBENCH benchmarks and validates LLMs' responses in various configuration scenarios. The findings establish foundational benchmarks to guide future applications of LLMs in enhancing network management efficiency.

***Index Terms*—Large Language Models (LLMs), Autonomous Network Management, Benchmark**

I. INTRODUCTION

The increasing complexity of modern network infrastructures necessitates sophisticated management solutions that can rapidly adapt to dynamic changes and scale according to emerging demands. Traditional network management techniques often rely heavily on human expertise, which can be time-consuming and prone to errors, particularly in large-scale or highly dynamic environments [1]. The advent of Artificial Intelligence (AI) and Machine Learning (ML) technologies, especially Generative AI and Large Language Models (LLMs), introduces a promising paradigm for enhancing automation and intelligent decision-making within network management.

The integration of AI into network management has attracted considerable interest in recent years. Among the most promising developments in this area is the application of LLMs for automating and optimizing networking tasks [2]. These models, with their advanced natural language processing capabilities, are well-suited for automating complex network management tasks, potentially reducing the dependency on human intervention and increasing efficiency.

Recent literature has explored the use of LLMs for network management through various applications, ranging from incident management [3], [4] to providing technical insights for human users [5]. These studies underscore the potential of LLMs to generate contextually relevant insights for networking tasks. However, despite the growing application areas and implementations of LLMs, there is no clear methodology to assess the suitability and performance of these models for network management.

The implementation of LLMs in network management is still in its developmental stages, with a significant insufficiency in comprehensive evaluation frameworks. Existing research [3]–[7] tends to focus on isolated applications, lacking a comprehensive assessment of LLM performance across diverse network management scenarios. This research gap raises the need for a standardized benchmarking framework capable of systematically evaluating the performance of different LLMs in real-world network settings. The real-world complexity of network systems, characterized by dynamic topologies and diverse protocols and configurations, calls for a robust methodology to evaluate the practical utility of LLMs in this domain.

To address this need, we introduce a novel framework NETLLMBENCH. NETLLMBENCH is specifically designed to evaluate various LLMs across a range of predefined networking tasks, such as IP address and default gateway configuration. The proposed framework provides a systematic methodology for testing and validating LLMs within emulated network environments, thereby offering essential insights into their practical utility. Additionally, NETLLMBENCH solves the problem of LLM model comparison by establishing a quantitative metric to compare different models through automation, as human-performed model verification is not feasible and prone to errors. By delivering a systematic approach for testing and validating LLMs, NETLLMBENCH offers crucial insights into their applicability for networking and paves the way for future advancements in AI-driven network management.

Through the NETLLMBENCH framework, this work establishes a foundational step towards comprehensively analyzing the performance of AI technologies in network management. As part of our primary analyses with NETLLMBENCH, we benchmark four of the popular LLMs in the Ollama [8] environment, including their variations with different numbers of model parameters. Our analyses include Meta AI's LLama3 [9], Mistral AI's Mistral [10], Google's Gemma [11] and Alibaba Cloud's Qwen [12]. We evaluate each model's performance systematically within an emulated network scenario to provide a comparative understanding of their capabilities in real-world network management tasks. As

a contribution to the research community and in support of reproducible research, we are making our framework publicly available [13].

The remainder of this paper is organized as follows: Sec. II describes NETLLMBENCH in detail. Sec. III presents the empirical evaluation. Sec. IV reviews related work on LLM applications for networking, and Sec. V concludes the paper and outlines future research directions.

II. FRAMEWORK

This section discusses the challenges involved in developing an LLM benchmarking framework and introduces NETLLMBENCH.

A. Challenges

Developing a framework for autonomously benchmarking LLMs for networking tasks involves several complex challenges that must be addressed to ensure the reliability of the framework. One of the primary challenges lies in defining a comprehensive and representative set of networking tasks that encapsulate a wide range of real-world scenarios. The creation of these tasks requires knowledge in the networking field and potential capabilities of LLMs. Currently, the task creation for NETLLMBENCH is done manually.

Ensuring that the LLMs' outputs are consistently structured and easily parsable is also crucial. For the validation stage, the outputs of the LLMs must adhere to the predefined JSON format. This necessitates the presence of a JSON verification stage in NETLLMBENCH.

The practical applicability of LLM outputs needs to be verified within an emulated network environment. This requires an emulator capable of providing feedback to the LLMs. A significant challenge is the provision of precise error feedback to the LLMs. This feedback is essential in refining the LLMs' responses in subsequent iterations. Designing a feedback mechanism that effectively communicates with LLMs to improve their response accuracy is one of the major challenges of NETLLMBENCH.

Finally, establishing clear and measurable benchmarking criteria and evaluation metrics is also essential to systematically assess the performance of LLMs. By addressing these challenges, the NETLLMBENCH framework aims to provide a rigorous, systematic evaluation of LLMs in network management, setting a benchmark for future developments in AI-driven network management technologies.

B. NETLLMBENCH

NETLLMBENCH introduces a novel method to autonomously benchmark LLMs for networking tasks. This benchmarking process is designed to be systematic and replicable. Fig. 1 provides a block diagram of the NETLLMBENCH, illustrating its structured approach. NETLLMBENCH involves four phases: task definition, testing, format verification, and emulation.

The initial task definition phase involves specifying particular networking tasks to evaluate the capabilities of LLMs in network settings. These tasks, derived from real-world scenarios, are formulated as questions that resemble typical or critical networking challenges. Each task should be clearly defined with specified input conditions and expected outcomes. The outcome requires the LLM to generate precise network configuration commands in JSON format, where the keys are "command" and "machine" (network entity where the command is to be executed).

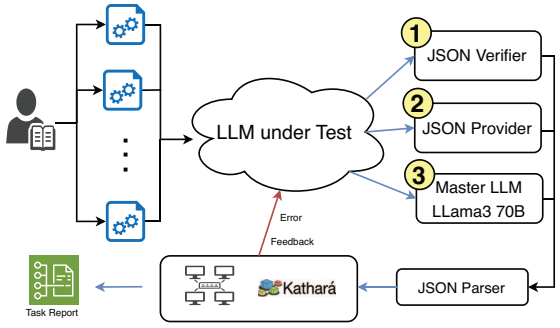

Fig. 1. Block diagram of NETLLMBENCH. It autonomously evaluates LLMs in networking tasks through a sequence of JSON validation and error feedback integration. NETLLMBENCH systematically evaluates the LLM under Test's performance in network management tasks.

In the testing phase, the LLM under Test is queried using these predefined tasks, and responses are systematically collected. In the format verification phase, these responses undergo a three-stage verification process to ensure adherence to the JSON format. Initially, a JSON verifier checks the syntactic correctness and structure. If valid, the JSON is parsed and validated in the network emulator. If the verifier rejects the structure, the JSON Provider component employs prompt engineering by providing an example of the correct JSON format for the LLM under Test to improve its response. If corrections fail, a *Master LLM*, specifically a LLama3 model with 70 billion parameters, is prompted to output the correctly formatted JSON string.

Finally, in the emulation phase, once the LLM under Test's outputs pass format verification, they are input into a custom-built emulator, developed using Kathará [14], which replicates a networking environment to practically test these outputs. This emulation phase is crucial for verifying the practical applicability of the LLM under Test's responses. It provides error feedback that is used to guide the iterative refinement process. This feedback is crucial for adjusting responses to ensure they are practically applicable. The LLM under Test is allowed a predefined maximum number of iterations to solve the task. Each iteration is used either for verifying the JSON structure or for refining the response based on the error feedback from the network emulator. After emulation, the framework evaluates the LLM under Test's responses against the feedback from the network emulator. Comprehensive performance metrics, including task-solving iterations,

accuracy, and model throughput, are analyzed, and the results are presented in Sec. III.

III. Evaluation

This section presents the measurement testbed, topology, and tasks used for the analysis, as well as the findings from NetLLMBench.

A. Measurement Testbed

We use four high-performance servers running Ubuntu 22.04 with the 5.15.0-107-generic kernel. Two servers are equipped with dual NVIDIA A40 GPUs, which are used for model inference. The models are managed using Ollama, a tool that is capable of deploying LLMs on NVIDIA GPUs with model parallelism. One GPU server consistently runs the *llama3-70b* model, acting as the *Master LLM*, and the LLM under Test is deployed on the second GPU server.

The third server hosts the Kathará network emulator. Finally, an orchestration computer oversees the workflow, managing the exchange of information between the LLM under Test and the network emulator. It ensures synchronous operation throughout the testing process. This setup tests the LLMs' ability to accurately manage and configure networks with NetLLMBench.

B. Topology and Tasks

We create an example topology for the evaluation of NetLLMBench. The network topology for the validation of LLM under Test's responses includes a router, two switches, and three hosts, as illustrated in Fig. 2. Each of the hosts is connected to the switches, which operate in Layer 2 forwarding mode by default, hence requiring no specific configuration steps. A router is integrated within this setup to facilitate all-to-all connectivity among the network components.

The entire topology is emulated with active links and connections, yet initially, none of the IP addresses are configured. The initial tasks assigned to the LLM under Test involve configuring the IP addresses of the respective interfaces of hosts and the router, as highlighted in blue in the figure (TASKS 1-5). The secondary tasks include configuring the correct default gateways in the hosts for correct routing (TASKS 6-8), before finally performing a ping test between the different subnets (TASK 9). The complete list of tasks and prompts is accessible via [13]. For the evaluation, we set the maximum allowed number of iterations to solve the task to 3. This gives the LLM under Test the opportunity to utilize all the provided format verification components within NetLLMBench.

C. Evaluated LLMs and Metrics

We evaluate NetLLMBench with the following LLMs:
llama3-70b [9], one of the largest and most recent models with 70 billion parameters developed by Meta Inc. Our evaluation focuses on the 8-bit quantized version of the model tuned for instruction use cases. With 70 billion parameters, and each parameter represented with a byte, the model occupies 70 GB of GPU memory.

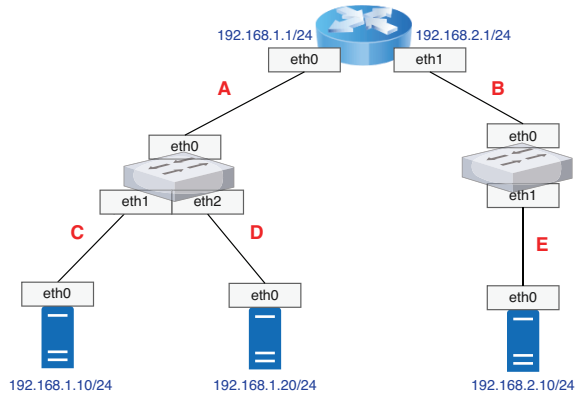

Fig. 2. Network topology for testing the LLMs. The considered topology includes a router, two switches, and three hosts.

llama3-8b [9], a smaller variant in the Llama environment with 8 billion parameters. It balances performance and computational efficiency, making it deployable in small-scale environments. Our evaluation focuses on the 16-bit floating point version of the model tuned for instruction use cases. The model occupies 16 GB of GPU memory.
mistral-7b [10], a model developed by Mistral AI with 7 billion parameters. Our evaluation focuses on the 16-bit floating point version of the model tuned for instruction use cases. The model occupies 14 GB of GPU memory.
gemma-7b [11], Google's state-of-the-art model with 7 billion parameters. Our evaluation focuses on the 16-bit floating point version of the model tuned for instruction use cases. The model occupies 14 GB of GPU memory.
gemma-2b [11], the lightweight version of *gemma-7b* with only 2 billion parameters. It is ideal for applications where resource efficiency is crucial. Our evaluation focuses on the 16-bit floating point version of the model tuned for instruction use cases. The model occupies 4 GB of GPU memory.
qwen-4b [12], a model developed by Alibaba Cloud featuring 4 billion parameters. Our evaluation focuses on the 16-bit floating point version of the model tuned for chat use cases, as tuning for instruction use cases do not exist for this specific model. The model occupies 8 GB of GPU memory.

The framework collects three key metrics to comprehensively evaluate the LLMs' performance in networking tasks: Chat History, LLM Performance Statistics, and Task Performance Statistics. Chat History consists of the log files of the interactions between the prompts and LLM responses. It captures the sequence of interactions between the LLM and the task prompts, providing insights into the LLM's reasoning process and response patterns. LLM Performance Statistics include metrics such as task completion time, resource utilization, and model throughput. Task Performance Statistics evaluate the practical effectiveness of the LLM's solutions within the network emulator, measuring syntactical correctness and accuracy of task completion. These metrics

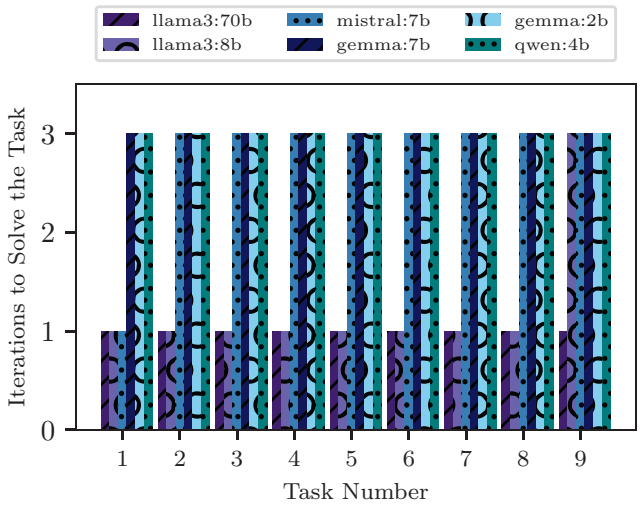

Fig. 3. Number of iterations required by each LLM to complete network management tasks.

together enable a thorough analysis of both the conversational and technical capabilities of the LLMs in a networking context.

D. Results

1) Iterations to Solve a Task: Fig. 3 shows the bar plot of each LLM's number of iterations to solve the task against the Task ID. It categorizes the performance of each LLM by the number of iterations needed to complete the tasks. The evaluated LLMs are shown with various colors in each bar. A single iteration to solve the task means that the task is successfully completed, whereas more than one iteration to solve the task does not convey information on the correctness of the completed task. Since each LLM is given a maximum of three iterations to solve a single task, regardless of its successful completion, our framework continues with the next task. In general, NETLLMBENCH evaluates each LLM's capability to solve a given task, and it is dependent on the LLM whether the iterations are being used for receiving an error feedback from the emulator or fixing its response to a JSON-compatible format.

The figure shows that *llama3-70b* is capable of completing all of the tasks in a single iteration. *llama3-8b* is capable of completing all of the tasks except the last one in a single iteration. Although all of the configuration tasks are done correctly by *llama3-8b*, the final ping test fails. This indicates that while all tasks were passed in the network emulator, suggesting the correctness of the issued commands, a configuration error exists. Inspection of the Chat History logs indicates that *llama3-8b* misconfigures the default gateway of the hosts. However, since the generated commands are syntactically correct, they are issued in the emulator without any errors. The final ping test, which covers cases for semantic misconfigurations, reveals that although *llama3-8b* produces correct commands, it cannot pass the final benchmark.

mistral-7b is the only other LLM that can solve Task 1 in a single iteration. However, its performance in the subsequent tasks also requires three iterations.

gemma-7b, *gemma-2b*, and *qwen-4b* cannot solve the tasks in a single iteration. The investigation of the conversation history shows that these LLMs are incapable of producing JSON-compliant outputs even after two stages of prompting. For these models, NETLLMBENCH relies on the *Master LLM* to extract the correctly formatted output.

2) Model Throughput and Task Performance: High throughput indicates an LLM's ability to process queries faster, thereby significantly reducing the response time for each individual request. This metric is particularly important in investigating an application's real-time capabilities.

Fig. 4 shows the throughput and task completion performance of various language models across different tasks. Each model is plotted in separate subplots. The horizontal axes represent the different tasks, labeled by task IDs. The tasks are numbered sequentially, with the iteration number indicated by a subscore. The vertical axes display the throughput in tokens per second, showing how many tokens each model outputs per second. The color of the scattered points indicates the task completion status. Green means the task was completed successfully, yellow indicates a needed iteration, and red shows the task has failed.

Overall, the first subplot shows that *llama3-70b* is capable of solving all the tasks without any iterations. The model also achieves a throughput around the 7.45 to 7.50 token/s range. The second subplot, with *llama3-8b*, indicates almost 5 times higher throughput than its bulkier counterpart with 70 billion parameters. However, despite the computational efficiency of the model, *llama3-8b* fails the final task.

The second row of subplots shows the performance of *mistral-7b* and *gemma-7b* respectively. While having the same number of model parameters, *gemma-7b* consistently achieves lower throughput than *mistral-7b*. This indicates that the Gemma model architecture is bulkier. Task completion wise, *gemma-7b* performs better, completing 5 out of the 9 tasks correctly in comparison to 4 out of 9.

The final row of subplots indicates the performance of *gemma-2b* and *qwen-4b*, which are two of the smallest analyzed LLMs. *gemma-2b* exhibits a significantly higher throughput than all the other models between 70 and 80 tokens/s. Considering that the number of correctly completed tasks between Gemma's 2 billion and 7 billion parameter models is the same, *gemma-2b* benefits from its lightweight structure when evaluated with NETLLMBENCH. Finally, our benchmarks indicate that *qwen-4b* achieves the worst task completion benchmark across all the models.

3) Model Size: When examining the model sizes, the *llama3-70b* model stands out by requiring 70 GB of GPU memory, compared to 16 GB or less for the other models. Consequently, it is not astonishing that *llama3-70b* shows the best performance, successfully completing all tasks without iterations. However, our evaluation reveals that significantly smaller LLMs also exhibit substantial potential, even if they do

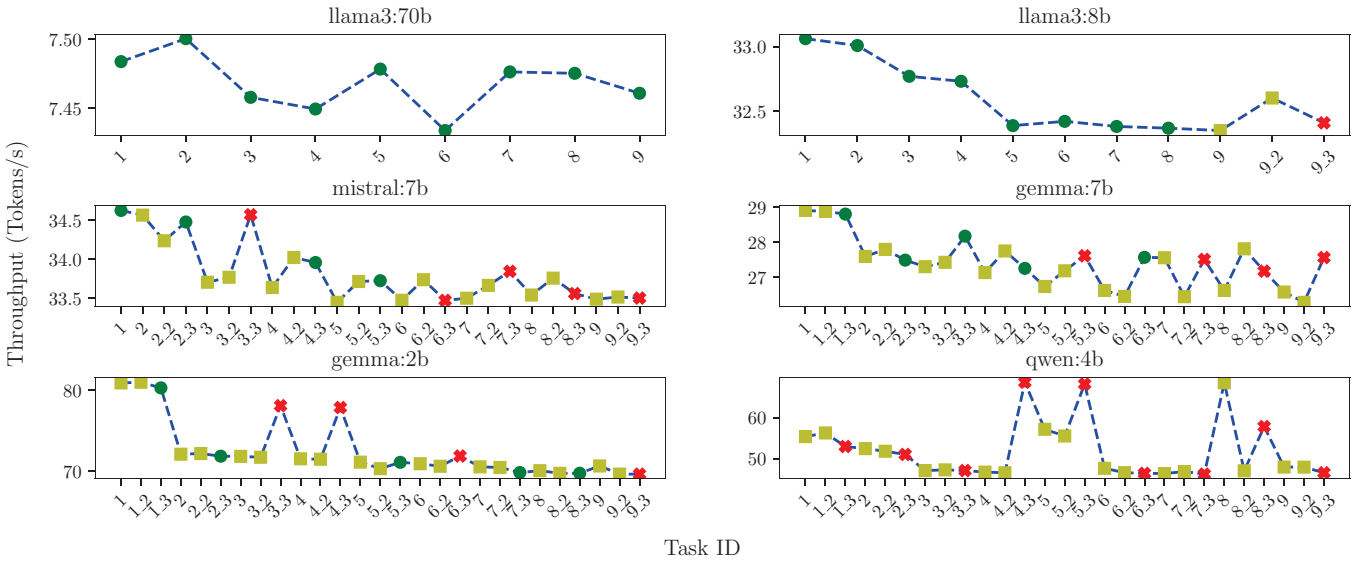

Fig. 4. Throughput performance of various LLMs across different network management tasks, measured in tokens per second. Correctly solved tasks are marked with green, whereas yellow shows that an iteration for the task is needed, and red indicates that the task has been unsuccessful.

not always succeed in completing the tasks. This observation raises a critical research question: what is the minimum size of an LLM necessary to effectively perform meaningful network management tasks? We believe that our initial findings, together with NETLLMBENCH will enable future research on benchmarking and improving the performance of promising smaller LLMs.

IV. RELATED WORK

Many AI applications in the field of networking are proposed and explored in the literature. Beurer-Kellner et al. [15] introduce a neural algorithmic reasoning approach for finding and generating scalable network configurations. The use of Natural Language Processing (NLP) methods for network configuration tasks is investigated by Ben-Houdi et al. [16] as an alternative to manually configured networks.

As LLMs have advanced, traditional ML and NLP approaches are often found inadequate for handling complex tasks or processing extensive contexts. LLMs have demonstrated promising capabilities in the last few years for generating human-like text. Although LLMs are originally designed for tasks such as text generation or language translation, their potential applications have rapidly expanded across various domains. In network management, these models are now utilized to automate complex processes such as predictive maintenance, anomaly detection, and dynamic resource allocation [2]. Such applications not only enhance operational efficiency but also significantly reduce the likelihood of human error and downtime, proving that LLMs can be essential in the optimization of network operations.

The survey by Zhou et al. [2] overviews the applications of LLMs in telecommunications. It highlights the use of LLMs for various tasks including code and network configuration generation, traffic classification, resource optimization, and traffic load prediction within the telecom sector. Additionally, the survey addresses the challenges associated with training these models and the necessity of prompt engineering for effective deployment in telecommunication tasks. This analysis emphasizes the potential of LLMs to enhance automation, improve efficiency, and overcome complexities of network management and configuration. However, the survey's analysis lacks a structured methodology, relying predominantly on subjective human evaluation.

Moreover, additional LLM use cases are emerging, including AI-driven network incident management [3], [4], synthesizing router configurations [17], and serving as co-pilots for network managers [5]. Recent advancements by Mondal et al. [17] highlight the challenges LLMs face in synthesizing correct router configurations. This work emphasizes the necessity for verified prompt programming to enhance accuracy and reduce manual oversight. Similarly, Wang et al. [7] investigate whether LLMs can facilitate network configuration and management, proposing a methodological approach to evaluate their effectiveness. These studies reflect a growing consensus on the potential of LLMs to manage network configurations but also underline the complexity and error-proneness of current models without sufficient verification mechanisms.

PROSPER [18] serves as a model that leverages LLMs to extract protocol specifications from Internet RFCs. Their approach demonstrates how combining textual and non-textual components from RFCs can enhance the extraction accuracy. In a similar approach, Xiang et al. [19] explore the reproduction of network research results using LLMs, pushing forward the discussion on the replicability and reliability of AI-driven network research.

While LLMs are finding their place in a variety of appli-

cations, the reliability of their performance remains uncertain. There is no unified and automated way of evaluating LLMs' performance. The NETLLMBENCH framework addresses the gap in the literature by providing a robust and systematic evaluation of LLMs for network management tasks.

V. CONCLUSION

This paper addresses the lack of systematic evaluation frameworks for LLMs in network management by introducing NETLLMBENCH, which provides a structured approach to test LLMs in realistic network scenarios using a custom-built network emulator. Our foundational analysis with NETLLMBENCH shows significant performance differences among the benchmarked models. Not surprisingly, the largest model, *llama3-70b*, performs the best, solving all tasks without iterations. Our evaluation reveals significant performance differences in smaller models, demonstrating their potential even if they are unsuccessful in the completion of the benchmark.

Future research should expand NETLLMBENCH to more tasks and diverse environments, improve feedback mechanisms, and explore integrating LLMs with mechanisms to improve task accuracy. We believe that this framework will lay the groundwork for systematic benchmarking of newly produced LLM architectures for network management tasks.

ACKNOWLEDGMENTS

We acknowledge the financial support by the Federal Ministry of Education and Research of Germany (BMBF) in the programme of "Souverän. Digital. Vernetzt." joint project 6G-life, project identification number 16KISK002. This work is partially funded by the German Federal Ministry of Economics and Climate Action (BMWK) within the funding measure "5G campus-networks" under the funding code 01MC22001B and supervised by the DLR Projektträger | Division Society, Innovation, Technology at the German Aerospace Center.

REFERENCES

[1] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: High-availability design principles drawn from googles network infrastructure," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 58–72. [Online]. Available: https://doi.org/10.1145/2934872.2934891

[2] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu *et al.*, "Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *arXiv preprint arXiv:2405.10825*, 2024.

[3] P. Hamadanian, B. Arzani, S. Fouladi, S. K. R. Kakarla, R. Fonseca, D. Billor, A. Cheema, E. Nkposong, and R. Chandra, "A holistic view of ai-driven network incident management," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. Cambridge MA USA: ACM, Nov. 2023, p. 180–188. [Online]. Available: https://dl.acm.org/doi/10.1145/3626111.3628176

[4] Y. Zhou, N. Yu, and Z. Liu, "Towards interactive research agents for internet incident investigation," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. Cambridge MA USA: ACM, Nov. 2023, p. 33–40. [Online]. Available: https://dl.acm.org/doi/10.1145/3626111.3628212

[5] M. Kotaru, "Adapting foundation models for operator data analytics," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. Cambridge MA USA: ACM, Nov. 2023, p. 172–179. [Online]. Available: https://dl.acm.org/doi/10.1145/3626111.3628191

[6] S. K. Mani, Y. Zhou, K. Hsieh, S. Segarra, T. Eberl, E. Azulai, I. Frizler, R. Chandra, and S. Kandula, "Enhancing network management using code generated by large language models," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. Cambridge MA USA: ACM, Nov. 2023, p. 196–204. [Online]. Available: https://dl.acm.org/doi/10.1145/3626111.3628183

[7] C. Wang, M. Scazzariello, A. Farshin, S. Ferlin, D. Kostić, and M. Chiesa, "Netconfeval: Can llms facilitate network configuration?" *Proc. ACM Netw.*, vol. 2, no. CoNEXT2, Jun. 2024. [Online]. Available: https://doi.org/10.1145/3656296

[8] Ollama, "Ollama," https://www.ollama.com, 2024, accessed: 2024-06-25.

[9] A. D. et al., "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[10] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[11] G. Team and T. M. et al., "Gemma: Open models based on gemini research and technology," 2024. [Online]. Available: https://arxiv.org/abs/2403.08295

[12] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, B. Hui, L. Ji, M. Li, J. Lin, R. Lin, D. Liu, G. Liu, C. Lu, K. Lu, J. Ma, R. Men, X. Ren, X. Ren, C. Tan, S. Tan, J. Tu, P. Wang, S. Wang, W. Wang, S. Wu, B. Xu, J. Xu, A. Yang, H. Yang, J. Yang, S. Yang, Y. Yao, B. Yu, H. Yuan, Z. Yuan, J. Zhang, X. Zhang, Y. Zhang, Z. Zhang, C. Zhou, J. Zhou, X. Zhou, and T. Zhu, "Qwen technical report," 2023. [Online]. Available: https://arxiv.org/abs/2309.16609

[13] TUM-LKN, "Github - tum-lkn/netllmbench: A benchmark framework for large language models in network configuration tasks." [Online]. Available: https://github.com/tum-lkn/netllmbench

[14] G. Bonofiglio, V. Iovinella, G. Lospoto, and G. Di Battista, "Kathará: A container-based framework for implementing network function virtualization and software defined networks," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Apr. 2018, p. 1–9. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8406267

[15] L. Beurer-Kellner, M. Vechev, L. Vanbever, and P. Veličković, "Learning to configure computer networks with neural algorithmic reasoning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 730–742, 2022.

[16] Z. B. Houidi and D. Rossi, "Neural language models for network configuration: Opportunities and reality check," *Computer Communications*, vol. 193, pp. 118–125, 2022.

[17] R. Mondal, A. Tang, R. Beckett, T. Millstein, and G. Varghese, "What do llms need to synthesize correct router configurations?" in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. Cambridge MA USA: ACM, Nov. 2023, p. 189–195. [Online]. Available: https://dl.acm.org/doi/10.1145/3626111.3628194

[18] P. Sharma and V. Yegneswaran, "Prosper: Extracting protocol specifications using large language models," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. Cambridge MA USA: ACM, Nov. 2023, p. 41–47. [Online]. Available: https://dl.acm.org/doi/10.1145/3626111.3628205

[19] Q. Xiang, Y. Lin, M. Fang, B. Huang, S. Huang, R. Wen, F. Le, L. Kong, and J. Shu, "Toward reproducing network research results using large language models," in *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. Cambridge MA USA: ACM, Nov. 2023, p. 56–62. [Online]. Available: https://dl.acm.org/doi/10.1145/3626111.3628189