Large Language Models for Zero Touch Network Configuration Management

Oscar G. Lira, Oscar M. Caicedo, and Nelson L. S. da Fonseca

The authors introduce the network configuration generator that architects zero-touch network and service management configuration agents by large language models by taking advantage of recent advances in generative artificial intelligence.

Abstract

The zero-touch network and service management (ZSM) paradigm, a direct response to the increasing complexity of communication networks, is a problem-solving approach. In this article, taking advantage of recent advances in generative artificial intelligence, we introduce the network configuration generator (LLM-NetCFG) that architects ZSM configuration agents by large language models. LLM-NetCFG can automatically generate configurations, verify them, and configure network devices based on intents expressed in natural language. We also show the automation and verification of network configurations with minimum human intervention. Moreover, we explore the opportunities and challenges of integrating LLM in functional areas of network management to fully achieve ZSM.

Introduction

The European Telecommunications Standards Institute (ETSI) defined a network management paradigm to realize network self-operation, self-maintenance, and self-optimization with minimal (or without) human intervention, named Zero-touch Network & Service Management (ZSM) [1]. Self-configuration, the ability to configure network parameters without human intervention, is an essential building block for ZSM. In addition, the Resource Configuration Management Service (RCMS) is a crucial component of ZSM for coordinating network configuration, improving operational agility, and reducing policy errors. RCMS oversees the network configuration management across network domains, empowering the network with autonomous configuration generation and deployment.

Intent-driven networks (IDN) [2] allow operators to describe their business objectives and requirements using natural language to reduce the complexity of network administration. ETSI defined intents to enhance and assist the RCMS by adopting actionable directives and translating them into policies and precise configuration parameters. IDN operation relies on effectively resolving intents to achieve operators' goals. Translating and solving intents requires advanced language processing and network knowledge.

A large language model (LLM) can create human-like text by processing large amounts of data. LLMs can be integrated into IDN networks to achieve ZSM, especially to generate network and device configuration recommendations. In this regard, [3] proposed NETBUDDY, a system that aims to simplify and automate the configuration of network devices using GPT-4. The authors in [4] explored LLMs to create recommendations on network device configuration using instruction prompts in GPT-4. However, these models demand large amounts of computational resources due to their size and consequently many accesses to remote platforms. Such an approach implies sharing sensitive information over the Internet, increasing the vulnerability to information leakage and unauthorized access to devices, even when sent over secure channels.

This article introduces the Large Language Model-based Network ConFiguration Generator (LLM-NetCFG) to address the threat of disclosing network configuration information. LLM-NetCFG employs a local LLM and avoids using an Internet-accessible LLM platform to translate high-level intents into specific network device configurations by locally executing the Zephyr-7b beta model. This approach ensures the privacy of information about network devices and infrastructure. Additional original contributions in this article are:

- A showcase of leveraging a local LLM for configuring Zero-touch network
- A showcase of verification of network configurations with minimum human intervention in a local environment
- A discussion of research opportunities and challenges to build LLM-based ZSM agents.

This article is organized as follows: The next section reviews large language models. We then discuss the application of LLMs in network management. Following that, we cover LLMs in ZSM. Then we present the operation of LLM-NetCFG. We then present a use case, followed by a discussion of future research. Finally, we conclude the article.

Large Language Models

Architectures

LLMs are AI applications with Natural Language Processing (NLP) capabilities, which allow machines to understand and mimic human language. Those capabilities come from using deep neural networks with a self-attention mechanism

Oscar G. Lira and Nelson L. S. da Fonseca are with State University of Campinas, Brazil;
Oscar M. Caicedo is with Universidad del Cauca, Colombia.

Digital Object Identifier: 10.1109/MCOM.001.2400368

that enables the model to dynamically weigh the relevance of each word to contextualize and understand the input text meaning word by word.

A transformer is a deep neural network (DNN) architecture (model) with attention mechanisms. Transformers efficiently enable self-attention mechanisms by processing multiple input sequences simultaneously, speeding up training and inference processes. Encoder-only transformers use self-attention mechanisms to capture dependencies between different words regardless of their position by considering only the input data. They produce a meaningful, fixed-size output representation that can be used for various downstream tasks (e.g., classification tasks). Decoder-only transformers generate coherent text by predicting the following words in a sequence, considering previously generated words using the self-attention mechanisms (e.g., summarization and translation tasks). Encoder-decoder transformers combine the encoder to process the input and the decoder to generate the output.

LLMs are versatile in handling various NLP tasks. However, their performance is influenced by factors such as the model's comprehension of inputs, the efficacy of its training process, and the expertise with which users interact to accomplish tasks' objectives. The combination of data preprocessing (i.e., tokenization), structured learning (i.e., training and fine-tuning), and strategic guidance (i.e., prompts and prompt engineering) ensures the customization of the model to perform specialized tasks with high accuracy and efficiency.

Tokenization and Training

Tokenization is a crucial step in LLMs forming the foundation for data understanding. Tokens, the discrete parts of a text, are the building blocks that LLMs use to comprehend the data. Tokenization involves separating a text into discrete parts. Training enables LLMs to comprehend and generate human-like text with coherence by learning the probability distribution of the vocabulary from a vast corpus using diverse techniques such as next-token prediction, masked language, n-grams, and continuous space.

Prompts and Prompt Engineering

A prompt is a statement or question that guides an LLM output. It sets the context and specifies the task required by a system, an assistant, or a user role. In the system role, a well-designed prompt ensures that the LLM understands the user's request accurately to generate a relevant response. The assistant role carries out a desired task based on the user's input, understands the context of the user's request, and tailors the response accordingly. The user role encompasses individual interactions with the LLM by inputting queries, instructions, or commands to get responses or perform tasks.

Prompt engineering determines the relevance (applicability and usefulness) and quality (accuracy and time metrics) of the responses given by LLM-based models. An effective prompt not only comprises carefully chosen words, but also emphasizes their arrangement, inclusion of specific keywords or phrases, or even the incorporation of example responses. Priming, zero-shot, one-shot, few-shot, chain-of-thought, and self-consistency are prompt engineering techniques.

Fine-Tuning

Fine-tuning involves training a pre-trained model on a specific task or dataset, seeking to refine the model's general knowledge and adapt it to the particularities of the new task. The adaptation facilitated by fine-tuning and the capacity for language understanding and handling massive amounts of knowledge has positioned LLMs as effective assistants for humans and promoted their incorporation into complex and diverse domains. Transfer learning and reinforcement learning can be used in fine-tuning techniques.

LLMs for Network Management Functionalities: What Has Been Done?

In the literature, researchers have recently started constructing domain-adapted LLMs for the networking field [5] and LLM-based network management functional areas: fault, configuration, accounting, performance, and security (FCAPS).

Fault

NetLM [6] introduces a network generative AI-based architecture designed to understand the sequence structures in the network data packets for 6G networks, predict maintenance, and perform troubleshooting tasks. RCAgent [7] proposes a practical and privacy-aware industrial root cause analysis solution. It explores LLMs for building customized conversational interfaces in cloud environments.

Configuration

NETBUDDY [3] addresses the traditionally complex and error-prone task of configuring network devices by using GPT-4 to generate human-readable configurations and actionable code for border routers and programmable switches. [4] introduces pair programming and verified prompt programming as methodologies to achieve GPT-4-human collaboration to synthesize correct router configurations crafted by network administrators, translate configurations from one network device brand to another one, and verify such configurations with minimal human intervention.

Accounting

Reference [8] evaluates the usage of LLMs in conversational interfaces within the telecommunications domain. The experiments were applied to various chat interfaces with questions related to 4G and 5G networks and device and resource management. The findings demonstrate that these models exhibit strong baseline performance and highlight the need for fine-tuning LLMs on domain-specific tasks to ensure reliable enterprise-grade applications.

Performance

[9] introduces a framework based on Generative Pretrained Transformers (GPTs), which automatically organizes, adapts, and optimizes performance in edge systems. [10] investigates how to improve the user experience and promote efficient network management with an AI-enabled network management framework consisting of environment, perception, action, and brain units that make decisions considering a knowledge base built with an LLM module and a Retrieval

Transformers efficiently enable self-attention mechanisms by processing multiple input sequences simultaneously, speeding up training and inference processes. Encoder-only transformers use self-attention mechanisms to capture dependencies between different words regardless of their position by considering only the input data.

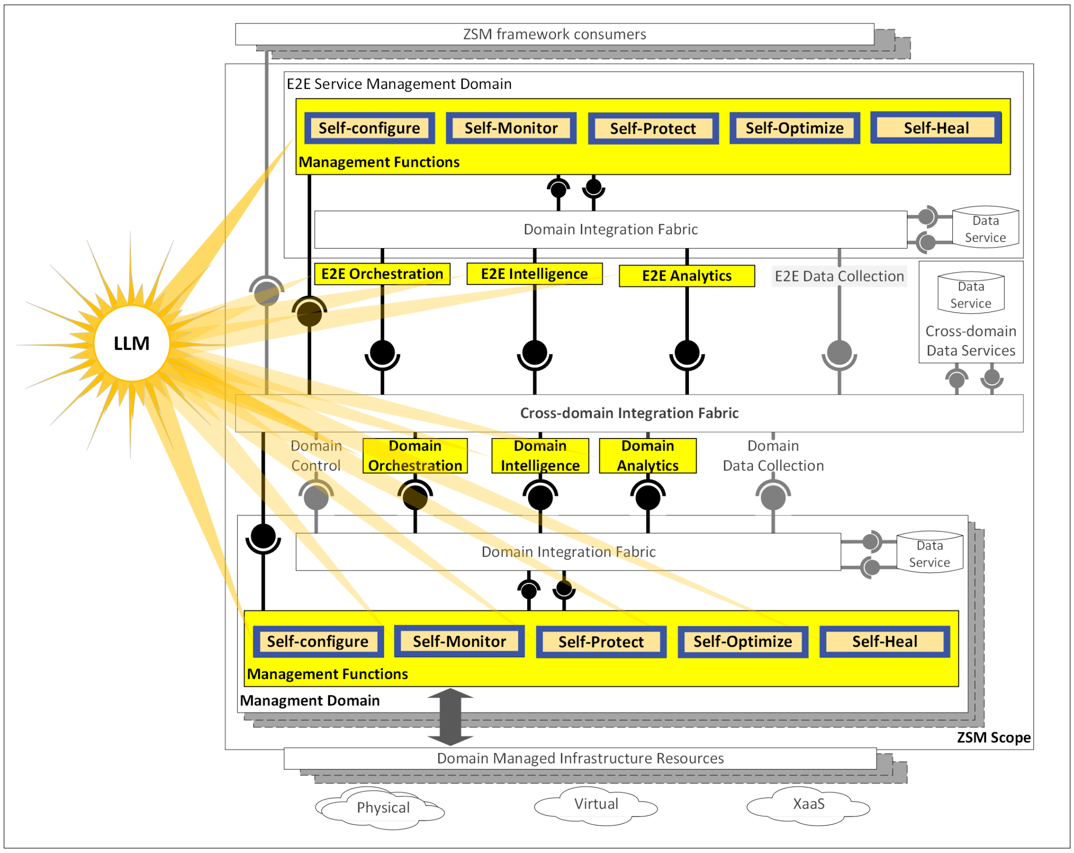

FIGURE 1. LLMs into ZSM components.

Augmented Generation module. NetGPT [11] proposes on-device LLMs for multi-agent intelligence in wireless zero-touch networks to improve user experience and network performance.

Security

SecureBERT [12] is a cybersecurity solution that leverages LLMs to detect and prevent fraud efficiently. This solution was trained on a vast corpus of customer data and security-related materials, including articles, books, surveys, blogs, and reports on threats, attacks, and vulnerabilities.

Using LLMs to realize FCAPS advances network management. However, ZSM requires the application of LLMs throughout its entire architecture to ensure end-to-end automation and orchestration. ZSM emphasizes comprehensive, autonomous network operations, which requires integrating LLMs in all layers and processes within the network to fully realize zero-touch management and service delivery.

LLM-Aided ZSM

To achieve ZSM objectives, supervised, unsupervised, semi-supervised, deep, and reinforcement learning models have been applied. However, to our knowledge, the use of LLM in ZSM has not yet been fully explored. LLMs can create conversational interfaces for network management, allowing operators to query the system using natural language and receive comprehensible explanations of AI-driven decisions with low complexity. Furthermore, LLMs can improve the automation of performance management and root cause analysis by providing more sophisticated analytical capabilities and improving the system's ability to learn from data, thus contributing to ZSM objectives such as self-management capabilities, closed-loop automation, and intent-driven management [13].

ZSM can be deployed as an intent-driven autonomous network that can perform self-configuration, self-monitoring, self-healing, self-optimization, and self-protection without human intervention. Figure 1 shows the ZSM architecture that highlights the components in which we consider LLMs to help manage the zero-touch network. The ETSI ZSM architecture comprises various components. *E2E orchestration* coordinates resources and services across multiple management domains. *E2E intelligence* leverages data analytics and AI for predictive analytics, anomaly detection, optimization, and adaptive learning. *Domain analytics* and *E2E analytics* are crucial to gaining insights by analyzing data within a domain or throughout the network. *E2E service* management ensures that services are delivered, maintained, and optimized by interacting with the *orchestration* and *intelligence* components. *Management functions* are responsible for performing autonomous tasks. *Domain data collection* and *E2E data collection* focus on gathering data to inform local or cross-domain management activities. The *Integration Fabric* facilitates data exchange between these domains, ensuring seamless connectivity and a unified data environment. *Cross-domain Data Services* enable data storage, sharing, and analysis across these domains, further supporting a unified data environment. *Domain control* manages and enforces policies within each domain, ensuring they align with overall *orchestration* and *intelligence* strategies. Closed-loop architecture automation, which includes observing, orienting, deciding, and acting functions, uses *cross-domain data services* to support self-configuration, self-monitoring, self-healing, self-optimization, and self-protection capabilities, facilitating a fully autonomous network management system.

LLM for ZSM Components

LLMs can be used to carry out the ZSM components highlighted as follows.

- *Domain Analytics and E2E Analytics* services can be enriched with LLMs to provide accurate domain-specific recommendations and root cause analysis to support decision-making within a domain and services spanning multiple domains. LLMs can be used, for instance, to gain insights about consumer behavior and network status to create new intra and cross-domain service strategies.
- LLMs can be integrated into *Domain Intelligence and E2E Intelligence* services to accomplish closed-loop automation by offering decision assistance, decision making, and assistance in the plan of action using the knowledge obtained from Domain Analytics and E2E Analytics. LLMs can suggest, for example, adjustments to resource allocations or routing configurations to improve services' quality and efficiency.
- In *Domain Orchestration and E2E Orchestration* services, LLMs can support autonomous service deployment by generating plans and configurations based on intent requirements and constraints. LLMs can also evaluate the feasibility of novel services and provide recommendations for optimizing resource utilization. LLM-NetCFG can support autonomous service deployment by generating configurations based on intent requirements and network constraints. Furthermore, LLM-NetCFG can assess the feasibility of previous changes to its deployment.
- When involving cognitive and closed control loops, network management functional areas can be matched to self-healing (faults), self-configuration, self-monitoring (accounting), self-optimization (performance), and self-protection (security). Self-configuration can leverage an LLM to generate configurations from network status and device knowledge combined with external templates. Self-monitoring can integrate LLMs to analyze the data for identification purposes, such as anomaly detection based on flow data. Self-healing can use LLM decision-making to perform corrective actions when a failure is detected or predicted. LLMs can enrich self-optimization by creating a feedback from the outcomes of optimization interventions of the previous decision-making processes and learning from past experiences and outcomes to understand the system's behavior and performance characteristics. Self-protection can integrate specialized LLM learning agents to improve system security based on fine-tuning with well-known attack datasets.
- Intents are pivotal for achieving ZSM and, in general, autonomous networks; network management functions cannot evolve and mature to full closed-loop autonomy without intents. LLMs can be used to build up intent managers responsible for translating business intents to service intents, service intents to resource intents, and resource intents to network configurations. LLM-NetCFG uses intents to identify the administrator's goal and decompose it in the required configuration steps to be achieved before generation.

Choosing LLMs For ZSM Components

Selecting an LLM for each self-management functional area framed in ZSM involves mainly model size, which determines how well the model would understand and generate language; the number of tokens, which determines how well the model can interpret data; fine-tuning capability, which allows for the creation of specialized models tailored to specific tasks; and forecasting capability, which enables the model to make accurate predictions based on data.

Self-configuration, self-monitoring, self-healing, self-optimization, and self-protection require a large model size and high token limit to handle the complexity, diversity, and volume of data involved in ZSM domains (Management Domains and E2E Service Management Domains). These self-properties also need solid analytical and forecasting features to operate proactively and accurately. Remarkably, Self-configuration demands fine-tuning support to adapt configurations to specific network environments and requirements.

LLM-NetCFG: Operation and Deployment

This section introduces the proposed LLM-NetCFG to achieve network self-configuration using LLM.

Overall Operation

The proposed LLM-NetCFG comprises the modules LLM, Verifier, and Orchestrator. These modules allow generating and verifying network configurations. The LLM module is responsible for the deployment and operation of an LLM in a local environment. It should guarantee that the model can receive questions and return answers. Also, it is required that the deployed model can be tuned so that model creativity can be nullified to receive consistent answers. The Verifier module receives a network configuration and checks its syntax errors. Moreover, it must check if the applied configuration achieves the primary goal of identifying inconsistencies or incongruities. The Orchestrator module coordinates automated tasks across the LLM module and Verifier module in different steps, allowing individual tasks to work together to generate and verify configurations.

LLM-NetCFG operates as follows (Fig. 2):

1. The Orchestrator crafts custom prompts so the LLM module can achieve the classification and translation requirements expressed by the network administrator in natural language to low-level requirements. It also crafts other prompts so the LLM module can create the configurations.
2. It uses the crafted prompts offered by the Orchestrator to perform the translation and configuration generation tasks (details below); before deployment.
3. The Verifier module checks the correctness and applicability of every generated configuration (a configuration can include a set of individual configurations since meeting an

ZSM can be deployed as an intent-driven autonomous network that can perform self-configuration, self-monitoring, self-healing, self-optimization, and self-protection without human intervention.

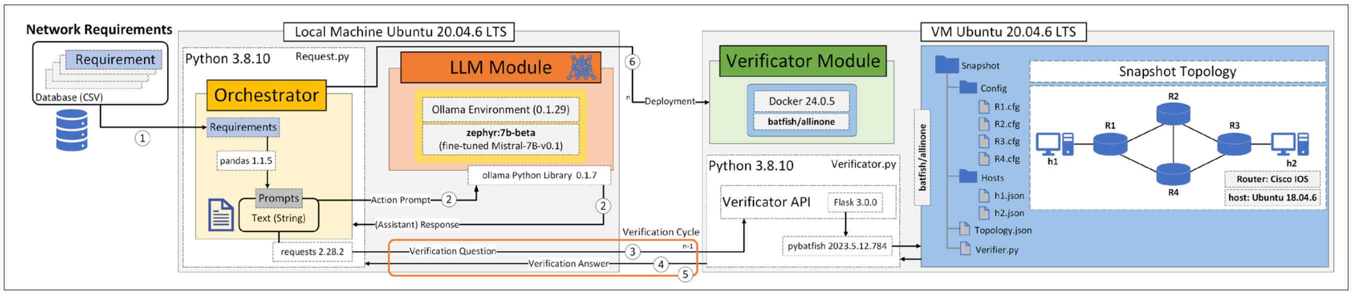

FIGURE 2. LLM-NetCFG structure.

intention can need modifying several devices set up); if this module detects errors or incongruities.

4. The Verifier generates and sends a report with suggestions to improve the Orchestrator that, in turn, feedback (via a prompt) to the LLM-Generator with the actual configuration to make changes (tune the configuration from step 2).
5. The Verifier notifies the Orchestrator when a configuration approves the verification process. A threshold value is necessary to avoid an infinite loop when the model does not generate a configuration that the Verifier approves.
6. The LLM-Orchestrator sends the approved configuration for deployment to a ConfigsRepo.

LLM-NetCFG aims to provide the self-configure property to the ZSM architecture. It distinguishes itself from previous proposals by constructing a domain-adapted model that demands low resources; it has been specifically designed, trained, and optimized for self-configuration tasks. An LLM model allows LLM-NetCFG to realize the ZSM configuration by converting a network intent described in a high-level natural language description into a configuration to achieve the network requirement goal. LLM-NetCFG integrates the validation step to prevent the model from introducing syntax errors when generating configuration commands. In particular, LLM-NetCFG is model-agnostic and modular, providing flexibility to integrate other management functions and facilitate its enhancement by specialized modules oriented to fine-tuning.

Prompts for Translation and Generation

Figure 3 exemplifies the LLM-Orchestrator tasks, automation prompts, and output examples. Initially, it uses the classification prompt to detect the configuration type received as input, an ACL described textually. Then, it uses the translation prompt to get a low-level description of such an input. Finally, it produces an executable configuration corresponding to the ACL rule by the configuration prompt. Each prompt includes a *system* role to instruct the LLM on the task to perform; an *assistant* role to illustrate the LLM output's shape, and a *user* role to provide the LLM with the inputs for the task to perform.

The *system* of the Requirement Classification Prompt identifies the type of Intent. The Classification prompt indicates its *assistant* role: "*In the first line of your answer, specify* {*type*}..." of Intent. In the user role, this prompt should input the Intent provided by the network administrator and the list of types of Intent. The *type* can be, for instance, CP for configuration interface properties.

The Requirement Translation Prompt specifies the system and assistant roles as follows: "*Your task is to take the high-level user* {*Intent*} *and define a* {*low_level_description*} *of the Intent in JavaScript Object Notation* (*JSON*)." In the *user* role, this prompt must input the high-level requirement to translate and the status of the network to the LLM. The status can be obtained from the verifier.

The Configuration Generation Prompt specifies the system role as a network administrator responsible for creating network device configuration files to meet the translated Intent. The *user* role provides input related to the general description of network devices and their brands and models. The assistant *role provides the answer structure that* the LLM must generate.

As LLMs usually explain their answers, it is necessary to inform them that they cannot provide explanations: "*Just answer with the required configuration files to satisfy the* {*low_level_description*}. *You are not allowed to provide any explanation*." Also, to assist Automation module in the verification process, a separator is used per device configuration: "*Indicate at the beginning which device the configuration should be applied and separate each device section with the special characters* ˜ ˜ ˜ "; this character is selected since it is not explicitly used in the configuration commands.

Use Case: Interface, ACL, Router, and Tunnel Configuration

This section introduces the proposed system by showing how it can generate and verify ZSM network configurations to support several tasks, such as creating and removing connections between devices and hosts, creating access control lists, setting OSPF routing rules, and creating secure communication using tunnel technologies.

Prototype and Test Environment

Figure 2 presents the LLM-NetCFG prototype available at https://github.com/oscarGLira/LLM-based-Intelligent-Configuration-Validation-Framework.git. The Generator was developed using Zephyr 7B b [14], an Open Source LLM fine-tuned by HuggingFace (i.e., a Mistral-based model), which excels in generating high-quality text and understanding complex prompts. Its improved context retention and fine-tuning performance make it adaptable to specific tasks. Its efficient processing and language generation suit tasks requiring intricate and well-defined responses.

Zephyr was selected as the base for LLM-NetCFG because it offers a large model size, high token limit, fine-tuning flexibility, and robust

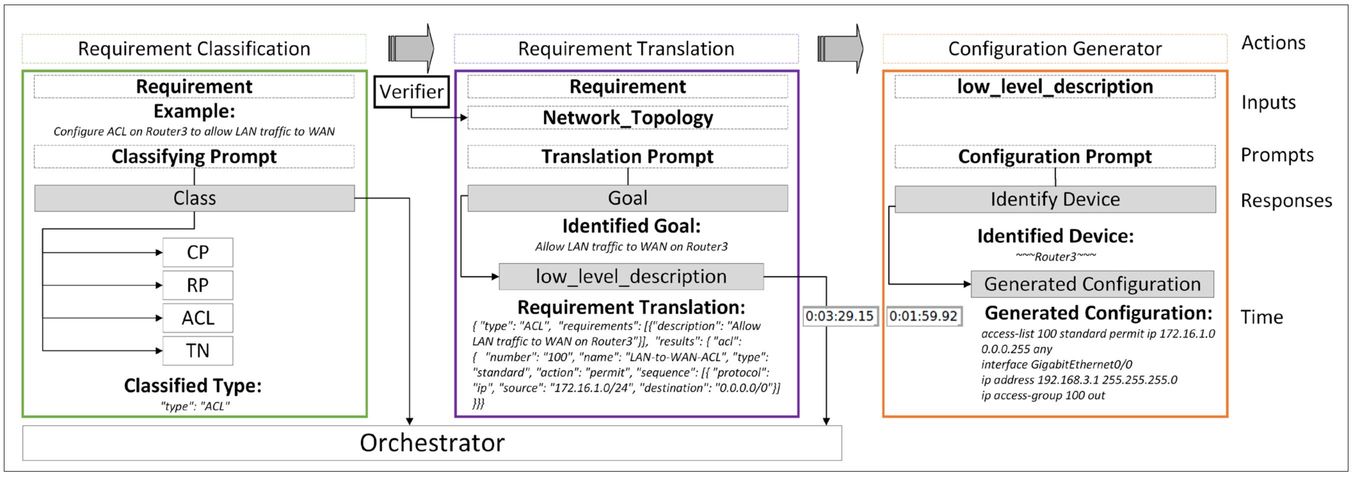
Requirement Classification
Requirement Translation
Configuration Generator
Actions
Requirement
Example:
Configure ACL on Router3 to allow LAN traffic to WAN
Classifying Prompt
Class
CP
RP
ACL
TN
Classified Type:
"type": "ACL"
Verifier
Requirement
Network_Topology
Translation Prompt
Goal
Identified Goal:
Allow LAN traffic to WAN on Router3
low_level_description
Requirement Translation:
{ "type": "ACL", "requirements": [{"description": "Allow LAN traffic to WAN on Router3"}], "results": { "acl": { "number": "100", "name": "LAN-to-WAN-ACL", "type": "standard", "action": "permit", "sequence": [{ "protocol": "ip", "source": "172.16.1.0/24", "destination": "0.0.0.0/0"}] }}}
0:03:29.15
0:01:59.92
low_level_description
Configuration Prompt
Identify Device
Identified Device:
~~~Router3~~~
Generated Configuration
Generated Configuration:
access-list 100 standard permit ip 172.16.1.0 0.0.0.255 any
interface GigabitEthernet0/0
ip address 192.168.3.1 255.255.255.0
ip access-group 100 out
Inputs
Prompts
Responses
Time
Orchestrator

FIGURE 3. Classification, translation and generation prompts — example.

forecasting capabilities. These characteristics are essential for handling the complexity and volume of network data involved in self-configuration, as previously discussed. Zephyr's large model size and token limit ensure it can process diverse and intricate network configurations and conditions.

The performance of Zephyr-7B has been evaluated using standardized benchmarks. A benchmark evaluated its general knowledge and reasoning capabilities in a wide range of topics (Abstract Reasoning Corpus). It was also tested for its ability to complete sentences contextually appropriately (HellaSwag) and its proficiency in university-level subjects such as humanities, science, and mathematics (Massive Multi-task Language Understanding). In addition, the model's accuracy was tested by providing accurate answers (TruthfulQA) or using common sense reasoning abilities (Winogrande).

Zephyr-7B was also evaluated using MT-Bench, which measures a model's ability to handle conversations that follow specific instructions, and AlpacaEval, which assesses a model's ability to respond to conversations based on one instruction. These benchmarks are used to rank LLMs. In these evaluations, Zephyr-7b scored 7.34 and 90.60, respectively, placing it at the top of all models of the same size. It outperformed other models, such as Falcon, Guanaco, Llama2, and Vicuna, with over 33 billion parameters, demonstrating the superior performance of Zephyr-7B. These findings show Zephyr-7B to be a reliable model. It performs better across various domains than larger models, especially in general knowledge, natural language understanding, and common sense reasoning, which strengthens the model's ability to perform the configuration generation task while being resource-efficient to be deployed locally with limited resources.

The Verifier was built as a Batfish [15] application (i.e., an Open Source network analysis tool that allows the detection of errors and correctness in network configurations) using Pybatfish API and Flask (i.e., a Python library helpful in offering the Verificator as a REST-based service).

The LLM-based Orchestrator and Generator modules were executed on a host machine with 16GB of RAM, a 2TB disk, and an Intel® Core™ i5 processor; this hardware configuration allowed Zephyr to run efficiently. The Batfish application was executed on a Virtual Machine running Ubuntu 20.04 server with 4GB of RAM.

Figure 2 also depicts the network for which the prototype generated configurations. The network follows a partial mesh topology with four Cisco routers. Two end-users are connected to R1 and R3. The topology and configuration details (about interfaces, OSPF for R1, R2, R3, and R4) are also available in a JSON file and in a configuration directory of our Git Hub.

Experiments and Results

In experimentation, we used a dataset with 90 resource configuration intents distributed in four types: CP, RP, ACL, and TN. Each Intent is a high-level requirement description of what the LLM-NetCFG is expected to achieve (Figure 3 exemplifies the Intent and Output configurations.). CP, RP, ACL, and TN include requirements related to interfaces, access lists, routing, and tunnel configurations, respectively, and are required in the verification process.

Figure 4 shows how LLM-NetCFG correctly classified 92.2 percent of requirements, which means that it efficiently understands the intents' goals, gathers relevant configuration information (e.g., from the Internetwork Operating System commands), and generates specific goals (in the tests for Cisco devices). LLM-NetCFG created a new class named *Other* for misclassified requirements. It had problems detecting not well-described ACL requirements; for example, an ACL defined to filter traffic SNMP was classified as *Other* (i.e., SNMP, a non-defined class) and not as ACL. The *Other* class highlights the LLM's hallucination problem, which can negatively impact ZSM configurations. It underscores the importance of precise high-level requirement definitions for optimal model understanding. Misclassification can affect the verification process, as the Orchestrator may only perform a syntax check rather than a specific verification tailored to the requirements, posing a significant challenge for integrating LLMs into real networking solutions.

Figure 5 shows that LLM-NetCFG spent five to seven minutes handling non-complex intent goals (incorporating a new device in the network, turning off a network interface, and setting up an access control list). This result is because the system needed just a cycle of classification and
~~~

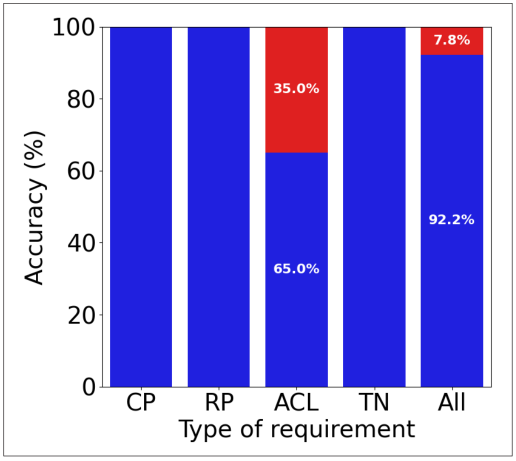

FIGURE 4. Accuracy in classifying requirements' type.

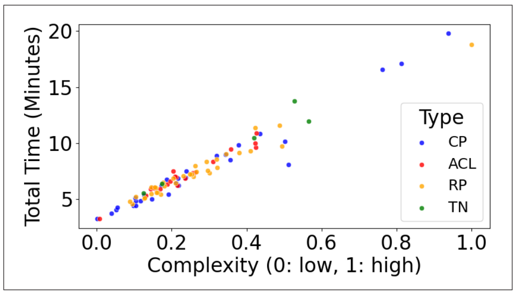

FIGURE 5. Total processing time by requirement complexity for each class.

translation of the intent's goal and generation and verification of configuration to meet it. The LLM-NetCFG took more than 10 minutes to cope with complex intents (including several goals) because it needed several verification cycles. Using fine-tuning techniques to reduce the model's time response for facing tasks with time requirements is also a relevant challenge to beat when realizing LLM-based network management.

Future Research Directions

LLMs for ZSM Management Functions: What can be done?

LLM for Self-Configuration: Device configuration troubleshooting extends beyond merely verifying syntax errors. Even when configurations appear error-free, they can still be insecure due to inadequate practices. LLMs can be integrated as sophisticated configuration correctness checkers, ensuring compliance and security throughout the configuration process. As LLM-based systems have improved coding practices, they can enhance automatic configuration systems.

Intent-Handling with LLM: Translating business goals into technical configurations can be complex and error-prone, impacting operational efficiency. An LLM-based autonomous intent mapper that interprets natural language input can streamline self-configuration by converting business goals into precise technical settings. The intent mapper can enhance operational efficiency and reduce configuration errors.

LLM for Self-Healing: Current ML-based methods for failure localization need to be scalable to analyze high-dimensional logs in moderate-size networks. Integrating LLMs into the ZSM architecture can assist in fault detection, localization, and diagnosis. When jointly used monitoring tools, LLMs can make the system more proactive in self-healing. Efficient data analysis techniques for leveraging LLMs' Knowledge Augmentation capabilities should be incorporated into network management systems capable of self-healing.

LLM for Self-Monitoring: Managing and analyzing large volumes of network data to generate actionable insights can be challenging. LLMs can overcome such a challenge with a zero-touch data analytics capability that enhances decision-making, improves service performance, and facilitates proactive network optimization by providing real-time visibility into network performance and usage patterns. Also, ZSM Data Services can benefit from LLMs for Root Cause Analysis, making identifying and resolving network issues more manageable and timely.

LLM for Self-Optimization: Due to the large volumes of data involved, detecting traffic patterns that lead to network performance degradation is complex. LLMs could assist in self-optimization by analyzing extensive data to detect the network segments, links, and devices that avoid QoS and QoE meetings, which is pivotal for achieving more reliable and efficient network performance.

LLM for Self-Securing: Detecting anomalies and security threats in large networks can be complicated. LLMs can enhance anomaly detection systems by supporting the understanding and processing extensive network logs, alerts, and contextual information to identify deviations from normal behavior. Also, LLMs can help security teams uncover the underlying causes of security incidents and breaches and generate actionable recommendations for mitigating risks, which is fundamental to self-healing.

Challenges Using LLMs

When integrating LLMs into the ZSM architecture, it is essential to note that these models present specific intrinsic challenges to their architecture and characteristics that need to be addressed. The most relevant are discussed below.

Hallucinations: LLMs may generate incorrect, nonsensical, or fake responses. Techniques like fine-tuning and prompt engineering, using refined and domain-specific information, can mitigate these hallucinations and improve performance.

Explainable LLMs: LLMs work as opaque models, which may limit their adoption in network management. Investigating explainability techniques, such as using the same model to justify its actions, can provide interpretability and transparency to LLM-based solutions.

LLM for Networks: Since there is no "Theory of LLM for Networks" for realizing network management with LLMs, specific datasets must be worked around to leverage LLMs' potential in networking. Additionally, coordinating LLM-based FCAPS into a closed cognitive control loop requires an orchestration layer to ensure that intents are met at the business, service, and resource levels.

Conclusion

Using LLMs to realize intent-driven self-capabilities framed into ZSM opens up a wide range of research directions and application possibilities for zero-touch networks. Leveraging the full potential of LLMs in autonomous network management will require devising novel frameworks for fast prototyping and proper training, fine-tuning and orchestrating FCAPS, and adequate interfacing between large models and real networks to ensure effectiveness and minimize exposure to security incidents. This article presented how LLMs can be used to design ZSM self-configuration agents, introduced LLM-NetCFG to automatically configure network devices from intents, and delved into the opportunities and challenges of integrating LLMs into ZSM.

Acknowledgment

This research was partially sponsored by CNPq grant # 405940/2022-0 and CAPES grant # 88887.954253/2024-00 and São Paulo Research Foundation #2023/00673-7.

References

[1] ETSI ISG ZSM, "Zero Touch Network & Service Management (ZSM) Standards," *ETSI, Tech. Rep.*, 2018.

[2] C. Yang *et al.*, "SMART Intent-Driven Network Management," *IEEE Commun. Mag.*, vol. 61, no. 1, 2023, pp. 106–12.

[3] C. Wang *et al.*, "Making Network Configuration Human Friendly," 2023.

[4] R. Mondal et al., "What do LLMs need to Synthesize Correct Router Configurations?" *Proc. HotNets*, New York, NY, USA, 2023, pp. 189–95.

[5] Y. Huang *et al.*, "Large Language Models for Networking: Applications, Enabling Techniques, and Challenges," *IEEE Network*, 2024, pp. 1–1, Early Access.

[6] J. Wang *et al.*, "Network Meets ChatGPT: Intent Autonomous Management, Control and Operation," *Int'l. J. Commun. Netw. Inf. Syst.*, vol. 8, no. 3, 2023, pp. 239–55.

[7] Z. Wang *et al.*, "RCagent: Cloud Root Cause Analysis by Autonomous Agents With Tool-Augmented Large Language Models," *CoRR*, vol. abs/2310.16340, 2023; available: https://doi.org/10.48550/arXiv.2310.16340.

[8] S. Soman and R. H. G., "Observations on LLMs for Telecom Domain: Capabilities and Limitations," *AI-ML Systems*, NY, USA: ACM, 2024.

[9] Y. Shen *et al.*, "Large Language Models Empowered Autonomous Edge AI for Connected Intelligence," *IEEE Commun. Mag.*, 2024, pp. 1–7, Early Access.

[10] R. Zhang *et al.*, "Interactive AI with Retrieval-Augmented Generation for Next Generation Networking," *IEEE Network*, 2024, pp. 1–1, Early Access.

[11] Y. Chen *et al.*, "NetGPT: An AI-Native Network Architecture for Provisioning Beyond Personalized Generative Services," *IEEE Netw.*, 2024, pp. 1–1, Early Access.

[12] E. Aghaei *et al.*, "Securebert: A Domain-Specific Language Model for Cybersecurity," *Security and Privacy in Commun. Networks*, F. Li, K. Liang, Z. Lin, and S. K. Katsikas, Eds. Cham: Springer Nature Switzerland, 2023, pp. 39–56.

[13] M. Liyanage *et al.*, "A Survey on Zero Touch Network and Service Management (ZSM) for 5G and Beyond Networks," *J. Netw. Comput. Appl.*, vol. 203, 2022, p. 103362.

[14] L. Tunstall *et al.*, "Zephyr: Direct Distillation of LM Alignment," *CoRR*, vol. abs/2310.16944, 2023; available: https://doi.org/10.48550/arXiv.2310.16944.

[15] A. Fogel *et al.*, "A General Approach to Network Configuration Analysis," *NSDI*, Oakland, USA: USENIX Association, 2015, pp. 469–83.

Biographies

Oscar G. Lira [GS'17] is pursuing his Ph.D degree in Computer Science at the State University of Campinas, Campinas, Brazil, and M.Sc in Computer networks and computer security and his degree in Telecommunications engineering in University Galileo of Guatemala, Guatemala. His research interests include network and service management, network virtualization, network softwarization, machine learning, and large language models.

Oscar M. Caicedo [GS'11, M'15, SM'20] is a full professor at the Universidad del Cauca, Colombia, where he is a member of the Telematics Engineering Group. He received his Ph.D. degree in computer science (2015) from the Federal University of Rio Grande do Sul, Brazil. His research interests include network and service management, network functions virtualization, software-defined networking, machine learning and large language models for networking, and network softwarization. He serves as Series Editor for the IEEE Communications Magazine Series on Network Softwarization and Management and Associate Editor of the IEEE Networking Letters.

Nelson L. S. da Fonseca [M'88, SM'01] received the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, CA, USA, in 1994. He is currently a Full Professor with the Institute of Computing, State University of Campinas, Campinas, Brazil. He has authored 400+ papers and supervised 70+ graduate students. Dr. Fonseca is currently the IEEE ComSoc Strategic Planning Chair and President Elected (2025). He served as VP-Conferences, VP Technical and Educational Activities, VP Publications, and VP Member Relations, Director of Conference Development, Latin America Region, and On-Line Services. He is the former Editor-in-Chief of the *IEEE Communications Surveys and Tutorials*.