Trantzas *et al. J Wireless Com Network* (2025) 2025:42
https://doi.org/10.1186/s13638-025-02472-x

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Intent-driven network automation through sustainable multimodal generative AI

Kostis Trantzas[1*†], Dimitrios Brodimas[1†], Besiana Agko[1], Georgios Christos Tziavas[1], Christos Tranoris[1], Spyros Denazis[1] and Alexios Birbas[1]

†Kostis Trantzas and Dimitrios Brodimas contributed equally to this work.

*Correspondence: ktrantzas@ece.upatras.gr

[1] Department of Electrical and Computer Engineering, University of Patras, Rio Campus, 26504 Patras, Achaia, Greece

Abstract

The concept of Network-as-a-Service involves deploying and reconfiguring next-generation networks, in a flexible and dynamic manner, to always cater to the needs of the respective stakeholders. It presents a complex challenge to manage and orchestrate computational and telecommunication resources across the cloud-edge continuum, especially with the growing focus on cost efficiency and energy consumption. To address this complexity, several technology enablers are considered, but the recent advancements in Large Language Models research have inevitably brought Intent-Based Networking to the forefront. This paper explores the architecture and implementation of an intent-based automation framework that adheres to contemporary industry standards, while also considering sustainability. To achieve this, a translation pipeline is introduced, based on emerging multimodal Generative Artificial Intelligence models, which transforms a high-level description of desired network capabilities and supplementary deployment files into machine-consumable information digested by the network itself. Therefore, several state-of-the-art online and locally deployed models are compared. The ultimate motivation of this work is to validate the feasibility and accuracy of the proposed framework, promoting sustainability through minimal resource consumption and cost efficiency. Additionally, the framework ensures compatibility with modern orchestrators and next-generation Operational Support System that follow the same industry standards.

Keywords: Intent-based networking, Large language models, Sustainable multimodal generative artificial intelligence, Network standards, Management and orchestration

1 Introduction

In recent decades, mobile networks have undoubtedly become essential to daily life, transforming communication, work, and interaction in general. However, the past decade has seen a rapid expansion in their use across industries and everyday activities. While this technological advancement has reshaped connectivity, it has also introduced new challenges for the designer of next-generation (next-gen) networks to address. As mobile networks evolve, their future hinges mainly on the ability to adapt [1]. This mandates that contemporary networks must maintain their reproducibility, resilience, and reliability amid rising demands and recent sustainability goals, such as energy consumption and cost efficiency [2]. Moreover, with their growing ubiquity, programmability and

customization are increasingly crucial to meet the diverse societal and vertical industries' needs.

It is important to recognize that network requirements for different sectors are dynamic and may change over time and context. Therefore, the ability to provide the necessary abstraction to address this issue is realized by offering Network-as-a-Service (NaaS). However, to effectively deliver NaaS and handle the inherent complexities of scalability, network operators are expected to enhance the existing orchestration platforms. These next-gen orchestrators should manage end-to-end, cloud-edge deployments, and optimize timely network management and energy usage, while also ensuring seamless service delivery to promote the inclusion of non-technical stakeholders.

Over time, the convergence of networks and their seamless interoperability has been a primary focus for academic and industrial standardization bodies, both characterized by high technical expertise. Efforts have concentrated on creating unified communication protocols and data structures to facilitate third-party orchestration across diverse network infrastructures. The 3rd Generation Partnership Project (3GPP) stands at the forefront of this initiative as the principal architect of mobile networks. Meanwhile, organizations like the TeleManagement Forum (TMF) and GSM Association (GSMA) have played key roles in aligning industry requirements and networking layers with the involved stakeholders.

In parallel, the exponential growth of mobile networks and their widespread adoption has led to a surge of users, not necessarily accustomed to standards and competent in network intricacies. These users seek to utilize advanced services, often unconcerned about their deployment and operation. This situation has immensely heightened the need for cognitive and user-friendly expression of high-level requirements, which can be subsequently translated into network-compatible configuration. Naturally, amidst these ongoing efforts, areas in networking that can address this need have rapidly gained momentum, such as intent-based networking (IBN). Although the concept of IBN has existed for some time, the recent rise of large language models (LLMs), generative artificial intelligence (GenAI), and multimodal artificial intelligence (AI) has made its effective implementation more feasible than ever, ushering in a new era in network architecture and management.

This work extends our previous research [3] to propose a framework that uses state-of-the-art AI and machine learning (ML) tools, alongside widely adopted standards-based interfaces, to deliver NaaS in a digitally inclusive way, further progressing it with the demonstration of results from different employed GenAI models to promote the solution's sustainability. Specifically, this work comprises:

- A new intuitive graphical user interface (GUI) to capture stakeholder's intent, paving the way towards the seamless incorporation of multimodal aspects
- Context distillation based on high-level descriptions and supplementary deployment files by employing Multimodal GenAI models
- Translation of the provided intent into machine consumable standardized interfaces, leading to actual network deployment
- Comparison of state-of-the-art online and local GenAI models to achieve the above
- Results that advocate in favour of the sustainability of locally deployed models

For that matter, the remaining part of this paper is organized as follows. Section 2 describes the related work and standardization activities on NaaS and IBN. Section 3 presents the methodology behind the proposed framework, while Sect. 4 validates its rationality through experimentation presenting an indicative Proof of Concept (PoC). In Sect. 5, result analysis and discussion are conducted, followed by the conclusions in Sect. 6.

2 Related work

2.1 Intent-based networking

Intent can be defined as a set of operational objectives that a network is expected to achieve, and outcomes that it is supposed to deliver, introduced in a declarative manner without the need to specify the implementation details [4]. IBN aims towards networks that are fundamentally simpler to manage and operate and only require minimal intervention from the user [5]. While trying to define IBN, its functionalities can be clustered into two main categories, i.e. Intent Fulfillment and Intent Assurance. Intent Fulfillment is comprised of intent ingestion, translation, and orchestration, spanning from intent's expression to its implementation. Intent Assurance is the process in which users validate that the targets set during Intent Fulfillment are met. From the above, it is easy to perceive that Assurance is complementary to Fulfillment [6]. As this work is solely focussed on the Intent Fulfillment aspect of the IBN, special mention should be given to the intent expression. Based on [7], the latter can be achieved via Templates/GUIs, natural language processing (NLP), Intent-based languages, Intent Application Programmable Interfaces (APIs), and Grammar/Keyword-based processing. It is anticipated that the combination of an intuitive GUI and the NLP, with the use of GenAI, can provide the perfect balance when it comes to intent expression among inexperienced and domain-aware users, thus defining the aim of this paper.

The aforementioned considerations do not imply that IBN is intended to establish a completely new communication paradigm. Instead, it primarily builds upon existing network frameworks that have already facilitated network provision and configuration automation in an attempt towards the autonomic network, such as software-defined networking (SDN), network function virtulization (NFV), and Network Policies. Major Standards Development Organizations (SDOs) have pioneered that attempt by exploring the application of AI into networks [8], but also intent management [9] and modelling [10]. Naturally, open-source projects have also emerged, such as Open Network Operating System (ONOS) [11], an SDN controller originally designed to provide large-scale, high speed and performant networks also directly incorporating intent interfaces [12], OpenDaylight [13], initially developed for datacenter networks, but promptly evolved to support multiple network domains, and Open Network Automation Platform (ONAP) [14], a comprehensive platform for orchestration, management, and automation of network with intent aspects [15].

Moreover, extensive research has been conducted in the field of intent ingestion and translation, as presented in our previous work [3]. The most notable mention was [16], where the authors developed an intent language that is translated into P4 templates. Lately, as expected, the related research heavily directed towards the employment of LLMs. For instance, this work [17] explores several LLM-based architectures for Edge

Intelligence. The authors of [18] propose a pipeline that decomposes intents and generates the required actions via a policy-based abstraction, exploring the few-shot learning capability of LLMs, which they attempt to validate through the development of the respective Intent Management System in [19], consecutively. Also, this work [20] introduces an intent-driven management and orchestration system that transforms TMF intent APIs and models into continuously monitored deployments. Nevertheless, to the best of our knowledge, no stable implementation has yet been developed that addresses the IBN concept, exploring multimodal aspects and different LLM models and deployments, while simultaneously enclosing the network automated deployment adhering by widely utilized standards set by major SDOs. Therefore, this gap represents the key motivation for this paper.

2.2 Network-as-a-service

The introduction of IBN in Sixth-Generation (6G) wireless networks aims to address the lack of intuitiveness, flexibility, and security issues commonly encountered in traditional networks. IBN plays a key role in translating users' business intentions into network configuration, operational, and maintenance strategies, rendering them essential for the design of AI-enabled 6G networks. However, truly meaningful support of IBN implies the programmability of the network as a prerequisite, both for deployment and configuration, often referred to as NaaS.

Considering NaaS, the Open Gateway [21] initiative is the most recent and notable endeavour of GSMA, TMF, CAMARA Project [22], and Linux Foundation along with 25 operators. This initiative represents the future paradigm shift whereby network operators expose and monetize network capabilities to third-party application service providers and developers, through common APIs. Since its launch, the main focus of the initiative was drawn towards the exposure layer to third-party customers, directly or through aggregators, via the Service and the Service Management APIs.

Although the mutual development of these APIs among the operators and the third-party stakeholders facilitated the well-sought abstraction of the network, skyrocketing their adoption and usage, the need for the transversal (non-service specific) functionality, that is required to make and realize a product out of the Open Gateway services, is still apparent.

The latter is captured by the initiative's Operate APIs offering the Operation, Administration and Maintenance (OAM) functionality, e.g. service fulfilment, supervision, and assurance, on top of which aggregators offer NaaS to the customers. These APIs are commonly utilized by the operator's IT stacks, including Operation and Business Support Systems, widely implementing TMF's reference digital architecture employing over 70 Open APIs [23]. In summary, a modern platform should be able to: (i) capture the customer's high-level business requirements within the domain context of networking and (ii) translate them into a defined set of available network slices, formalizing the corresponding service level agreement (SLA) with the customer. Therefore, the platform would need to determine the service's end state and the respective compliance with the imposed restrictions following best practices.

While the initiative is relatively new and still developing, several subsets of its components have been implemented by various standalone projects. One of those efforts stems

from the European Telecommunications Standards Institute (ETSI) Software Development Group for OpenSlice (SDG OSL) which develops an open-source service-based Operational Support System (OSS) to deliver NaaS, implementing numerous TMF APIs towards that goal [24]. Furthermore, it incorporates an inherent mechanism to express the business requirements that GSMA dictates in General Slice Template (GST) [25], to formulate a SLA and consequently a network slice, into a TMF Service Specification, thus rendering them machine consumable, as described in [26]. For the reasons above, it becomes naturally apparent that OpenSlice could act on the Operate APIs plane of the Open Gateway initiative.

3 Methods

In this section, we introduce a deployment framework that leverages the potential of state-of-the-art AI and ML tools to capture a multimodal intent, in addition to standardized APIs for intelligent network orchestration, as depicted in Fig. 1. In this proposed framework, the user initiates the process by specifying high-level business intent and providing supplementary files to request a network slice. These files may include deployment descriptors, such as Helm charts, Network Service Descriptors (NSDs), and archives or even previously requested slice descriptions. The request is processed by the Dialogue block, which extracts insights from the input and forwards both the input and insights to the LLM component for contextual analysis. Subsequently, the LLM generates a Network Slice Template (NEST) [25] in JSON format, which is sent to the Dialogue block for the user to review and confirm whether the suggested slice description meets the intended requirements. If the user is not satisfied, this process is repeated; otherwise, the NEST JSON is handed over to the Translator component, where it is formatted according to the Operate APIs of a next-gen OSS, such as a TMF Service Order request. In succession, this request is forwarded to the OSS, where the infrastructure administrator conducts a feasibility check for the appealed network slice. Upon approval, the OSS proceeds with fulfilling the captured Service Order, employing lower-level Operate APIs, e.g. TMF Resource Ordering, to entrust the realization of the network slice to the

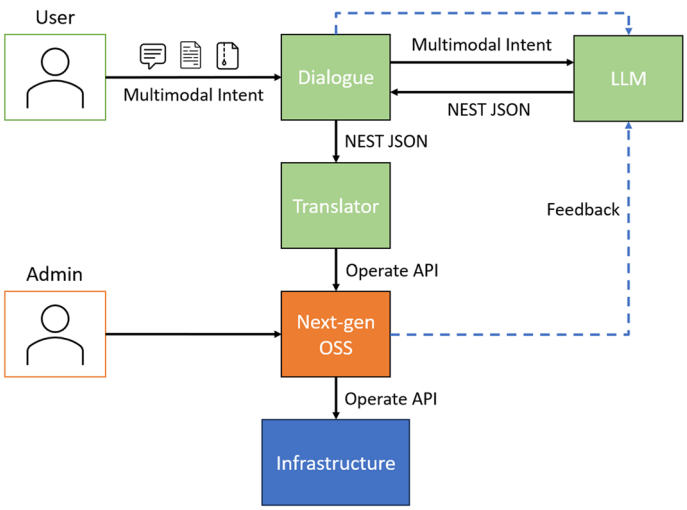

Fig. 1 Proposed LLM-enabled intent framework

infrastructure. Following, the remaining section presents each intent-related (green) element of this novel architecture in detail.

3.1 Dialogue

This element serves as the facilitator within the framework, acting as an external access point for user interaction. It accepts a high-level description of the required network characteristics and any deployment-related files as input. The block analyses the input for key terms, then forwards both the keywords and the original input to the LLM block. The Dialogue component retrieves the NEST JSON generated by the LLM, which the user can optionally review. Subsequently, the generated NEST is passed on to the Translator block.

The framework aims to adopt both a zero-touch and a co-design concept. The former dictates that the Dialogue component can be configured to automatically forward the generated NEST JSON without user review, minimizing human intervention, thus rendering it more suitable for interfacing with external applications, whereas the latter suggests that the Dialogue block is enhanced with an intuitive GUI that enables the user to reap the benefits of an LLM, reviewing the dialogue outcome along the way. Nevertheless, this review can also be inputted back to the LLM to train it in real-time (top blue dashed arrow in Fig. 1).

3.2 Large language model

GenAI, particularly LLMs, have gained significant attention in the NLP field due to their ability to produce coherent and contextually relevant text. These models are trained on vast datasets, enabling them to comprehend and produce human-like responses to various queries. Constructed using deep learning techniques, especially utilizing a type of neural network called a transformer, LLMs have proven highly effective across a range of NLP tasks. The transformer network serves as the core of an LLM, consisting of multiple layers of self-attention mechanisms and feed-forward neural networks. When generating responses, the model employs the self-attention mechanism to assess the importance of different words or tokens in a sequence. Text input for LLMs is typically tokenized into smaller units like words or subwords, with each token assigned a unique numerical representation known as an embedding. This tokenization enables the model to systematically analyse text and capture relationships between tokens. During inference, the model receives a sequence of tokens as input and generates a probability distribution over possible subsequent tokens, ultimately selecting the one with the highest probability as its output.

In our previous work [3], we considered using few-shot learning [27] to equip the general pre-trained model with insights and knowledge of the networking domain. This transfer learning approach leveraged a minimal amount of information, including definitions, descriptions, examples, and behaviour specifications (e.g. network engineer), as prompts [28]. These prompts were employed to fine-tune and customize the models, avoiding the need to adjust a model's entire parameter set. Moreover, this technique enabled the addition of new tasks to the model without erasing or interfering with previous tasks. Few-shot learning is highly effective for large pre-trained models in acquiring domain-specific knowledge from limited examples. However, it relies on prompts

processed against a few trillions of parameters, which can often demand significant computational resources, surpassing those required for the employing tasks. Combined with the cost constraints of many commercial models, this highlights the need for smaller, more resource-efficient models that reduce energy consumption and overall expense. Nevertheless, the transition must ensure that the response quality and accuracy are maintained by applying the same training techniques.

Figure 2 demonstrates our proposed model training pattern, based on three main service types in 5G and beyond networks: Enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low Latency Communications (URLLC), and Massive MachineType Communications (mMTC). The eMBB segment includes attributes like Peak Up Link, Peak Down Link, Broadband, and High Throughput. URLLC focuses on mission-critical features such as Reliability, Low Latency, and Real-Time communication. The mMTC segment pertains to Internet of Things (IoT) applications and includes attributes like Density, Massive Connectivity, and Low Power. This chart is used to depict the different use case categories that the introduced technology aims to address and serves as a foundation for training, so as to explain patterns in the generated datasets. In addition to the classification of service types, the model can be provided with complete NEST JSON files that correspond to specific service types [25], yielding an already cognitive basis for the Translator component to convert them into machine-consumable requests. Furthermore, during the inference phase, the user's intent can be optionally augmented with deployment descriptor files, e.g. supplementary JSON files. This enables the LLM to access additional information and refine the output in collaboration with the user, thus leading to even more precise NEST JSON outcomes for the Translator.

3.3 Translator

As future networks shift from traditional SLA enforcement, which relies on static, policy-based rules, and instead evolve towards more elastic and dynamic SLAs, it becomes crucial to employ a communication model that maintains adaptability while adhering

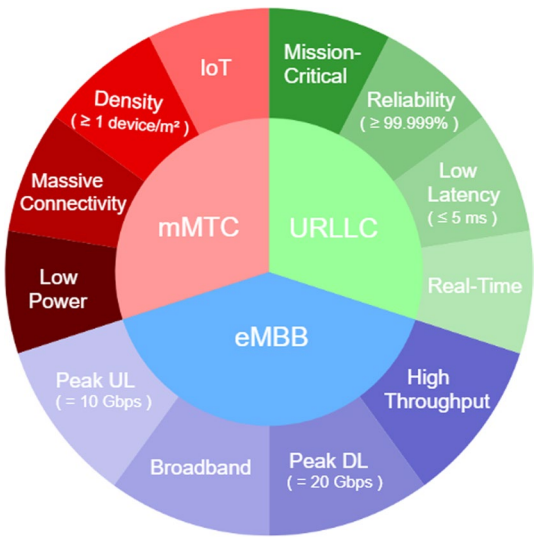

Fig. 2 Network knowledge insights to training data

to standardization, so as to avoid segmentation. Modern SLAs now incorporate Key Value Indicators (KVIs), such as energy efficiency and security aspects, alongside the traditional Key Performance Indicators (KPIs), like latency and throughput. While the policy-based approach was sufficient for legacy SLAs, which focussed on measurable KPIs, contemporary SLAs involve more abstract, non-quantifiable values, often stemming from non-native network context languages, making the traditional approach inadequate. An alternative method is therefore required to address these complexities.

In this work, this challenge is addressed by leveraging the well-established GST, as seen in the previous section, to generate SLAs based primarily on the user's intent, assisted by GenAI. However, to also ensure compliance with the Open Gateway platform, the proposed Translator block also incorporates a mechanism for converting the generated SLA (NEST JSON) into a TMF-compliant model, namely the Service Order, as illustrated in Fig. 3, which can be directly utilized by any TMF Operate API consumer. This transformation process ensures seamless integration with standardized frameworks, such as next-gen OSSs.

4 Experimentation

The following experimentation procedure took place at the Patras5G testbed [29] which is equipped with competitive computational capabilities and a broad range of telecommunication equipment, enabling the overall endeavour. Specifically, it leverages GenAI to transform an expression of high-level business requirements to the corresponding Operate API call, which is forwarded to a next-gen OSS to facilitate NaaS, offering the appropriate network slice. Building on our previous work [3], this new PoC repeated the same experiment using updated versions of commercial online/remote models but introduced a locally deployed model, as well. While our previous experimentation aimed to validate that GenAI can serve as a technology enabler for providing NaaS, this work specifically focuses on the sustainability of the solution by exploring ways to minimize resource consumption and improve cost efficiency. It soon became apparent that

Fig. 3 NEST JSON transformation to TMF Service Order

achieving these goals was only possible through local model deployment, which allows better control over resource utilization, model size, and fine-tuning.

Both the Dialogue and the Translator blocks were developed as microservices and deployed over Kubernetes. The experimentation framework employs two key APIs: (i) OpenAI's API to remotely connect with GPT-3.5 Turbo and the newer GPT-4 [30] and (ii) Ollama API to establish a local connection with Mistral 7B [31]. Additionally, utilizing these APIs enabled the development of a user-friendly GUI for model interaction, eliminating the need for a Command Line Interface (CLI) as in our previous work. This enhancement not only improves user experience and inclusion but also introduces a co-design concept through a user-LLM dialogue, which is also captured at the PoC. The dialogue supports free-text comprehension as well as understanding deployment-specific inputs, such as JSON text, during the network slice negotiation. Eventually, the generated inference is forwarded via the Translator block into OpenSlice, which serves as the next-gen OSS to fulfil and deliver the network slice request.

In terms of hardware, the local LLM and Ollama API were hosted on a virtual machine with 8 CPUs, 16 GB RAM, and 10 GB of free disc space. This machine was also equipped with an NVIDIA RTX A6000 Graphics Processing Unit (GPU), featuring 48 GB of RAM, which was available at the time. This setup effortlessly managed the model's parameters, far exceeding the anticipated hardware requirements. Comparable inference times could be achieved with more commonly available GPUs. While an even less resource-intensive setup, without a GPU, could also be used, it would result in slower inference times as a trade-off.

Finally, the presented experimentation involves a user-LLM interaction comprising a total of six prompts. Specifically, it begins with three prompts provided to the LLM, as part of the few-shot training, simultaneously processed with the user's first prompt. The user interacts with the LLM through the GUI, introducing three additional prompts: the initial network slice request, a negotiation based on the first slice offered, and a final confirmation to order the slice for deployment.

4.1 Realizing the LLM

To provide the necessary contextual awareness for both remote and local models, the few-shot learning technique was applied, as explained in Sect. 3.2. This involved compiling an initial classification of common intent descriptions (prompts) into slice types, as shown in Table 1.

Although, to fully leverage GenAI capabilities, the training process required the model to recognize that these intent descriptions were supplemented with specific NEST JSON files, which represented and characterized different slice types, as referenced in [25]. An example of such a JSON file describing the mMTC slice type is presented as follows:

```
{
   "Availability":"99,9",
   "Slice quality of service":"9",
   "Supported device velocity":"2",
   "UE density":"100000"
}
```

Table 1 Classification example of common intent descriptions (prompts)

I want a network slice to support a massive IoT I want a network slice to support a wide number of devices I want to support a large number of devices. I want a network slice for industrial use	mMTC
I want a network slice to support URLLC I want a network slice to support low latency communication. I want a network slice to support critical communication I want fast and reliable communication I want a network slice for XR	URLLC
I want a network slice to support eMBB I want a network slice to support wide broadband communication I want a network slice with extended channel capacity I want a network slice for 4 K videos. I want heavy traffic communication	eMBB

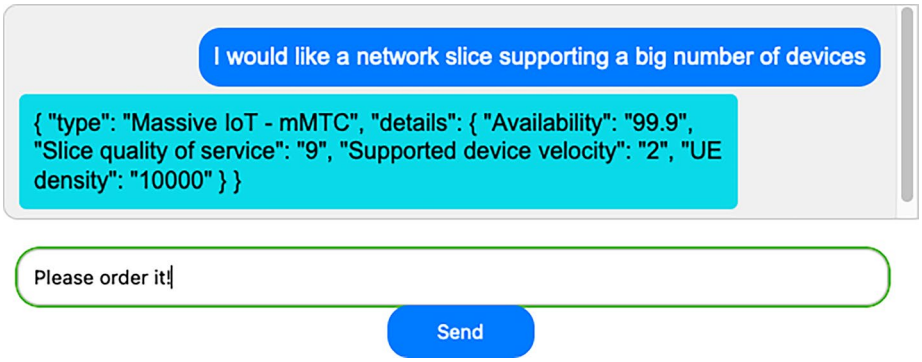

Fig. 4 Evaluated mMTC NEST generation

The described classification, along with the sample JSON templates, serves as the foundation upon which users can build to negotiate and co-design the requested network slice with the LLM, as seen in the following section.

4.2 User and LLM co-designing the network slice

The process begins when the user inputs its high-level intent through the Dialogue Block's GUI. Following, the LLM translates this intent into the relevant NEST JSON format. An example is shown in Fig. 4, where the user requests a network slice in a descriptive manner, and the LLM accurately maps it to an mMTC type with corresponding characteristics. A key point here is that the LLM does not simply uncritically replicate the training data. For instance, when the user specifies*"big number of devices"*, the LLM interprets this as a deviation from the typical*"massive number"* and adjusts the JSON field accordingly, reducing the default UE density by a factor of 10 (from 100,000 to 10,000).

Similarly, Fig. 5 presents another implicit user request for a different slice type, with the LLM generating the respective network slice. The user then interacts with the LLM to adjust the configuration to better suit its needs. This negotiation occurs in a chat-based environment, simulating a conversation with a network administrator. At this designated example, the user negotiates the reduction of *"availability"* of the slice.

Using the Figure's 5 example, once the co-designed network slice, representing a SLA expressed in the form of a NEST JSON, is accepted by a user's prompt, it is sent to the

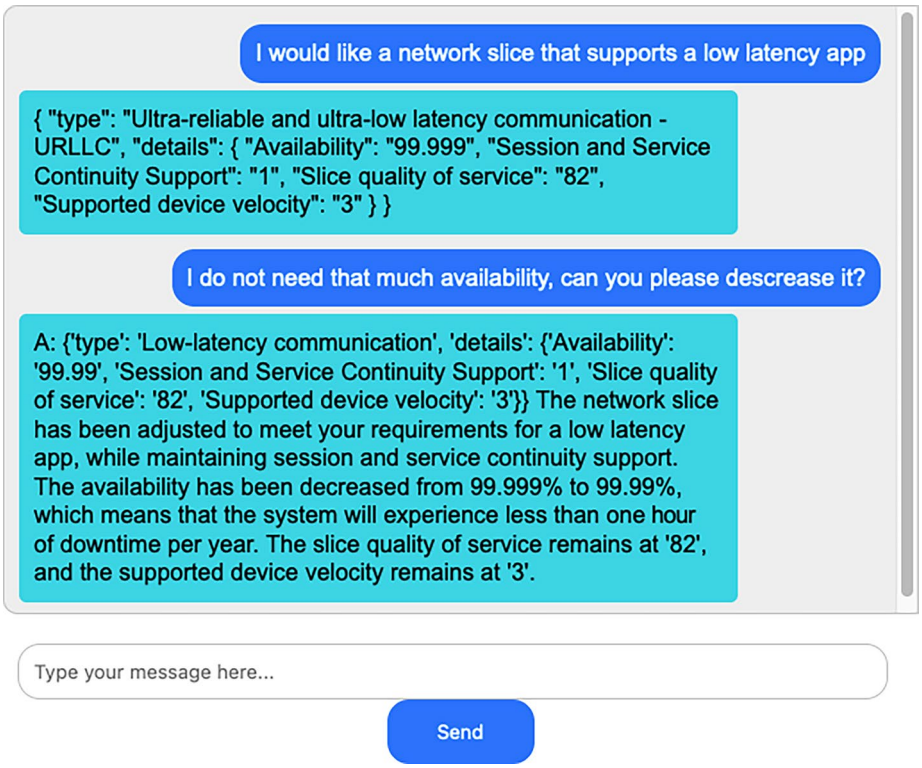

Fig. 5 Interaction and co-design with the LLM

Translator block. This block converts it into a next-gen OSS-ready Operate API request, as detailed in Sect. 3.3. It is important to note that this step does not modify the generated NEST JSON but only transforms it to fit the Operate API context, making it compatible with the employed next-gen OSS, namely OpenSlice. The specific Operate API used for this interaction is the TMF641 Service Order API, marking the transition from high-level business intent to NaaS delivery.

4.3 Network deployment

As expected, the final step receives the negotiated SLA, which is captured by OpenSlice. Since the request payload, illustrated in Fig. 5, is already in a native OSS context (TMF Service Order) after its transformation from the Translator Block, it can be directly processed, as shown in the Service Order management GUI in Fig. 6.

The fulfillment of the Service Order request is delegated to the respective integral component of OpenSlice, namely the Service Orchestrator, which undertakes the transition from the service layer to the infrastructure domain and the employment of the corresponding resources. In the presented deployment, the overall network provisioning is captured by the *5G Connectivity and Internet Access* service. This high-level service is further decomposed into the *User Plane Function (UPF)*, *Core*, and *Connect Radio* services, which sequentially deploy the UPF separately from the rest of the 5G core and finally connect the relevant radio nodes to the latter.

Naturally, the aforementioned process cannot be achieved only by the information included in the negotiated SLA but requires further deconstruction into

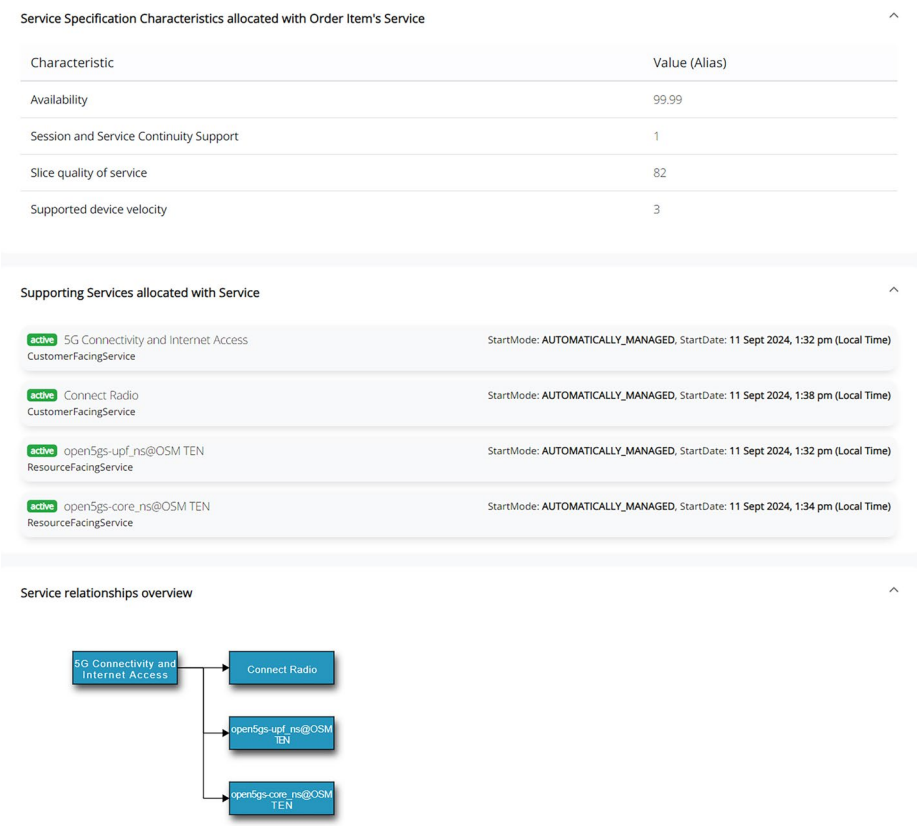

Fig. 6 OpenSlice Service Order management GUI

implementation-specific details at the orchestration layer. The conditions for this breakdown are provided by the Service Designer/Infrastructure Administrator during the design phase of the high-level service, leveraging OpenSlice's capability to define complex conditions and actions throughout the service operation via the Lifecycle Management (LCM) rules [32]. At the demonstrated deployment, before each service provision, the designated implementation information is extracted and accessed from the original SLA as seen in Fig. 7, based on the designer's LCM rules. Specifically, the *Supported device velocity* of "3" implies a vehicular speed, hence the outdoor nodes are selected for the *Connect Radio* service. On the same principle, the *Availability* of "99.99" hints towards a deployment at a typical performance compute node for the *UPF* and *Core* services, instead of the high-availability equivalent.

5 Results and discussion

In this section, we revisit the previously introduced scenario, which follows the process from capturing user intent to applying few-shot training and ultimately to network deployment, to evaluate its overall time. We also provide a breakdown of execution times using different models. The employed models comprise two common remote ones, GPT-3.5-turbo and GPT-4, along with a locally deployed model, i.e. Mistral 7B. As already mentioned, each iteration of the experiment involves a total of six prompts:

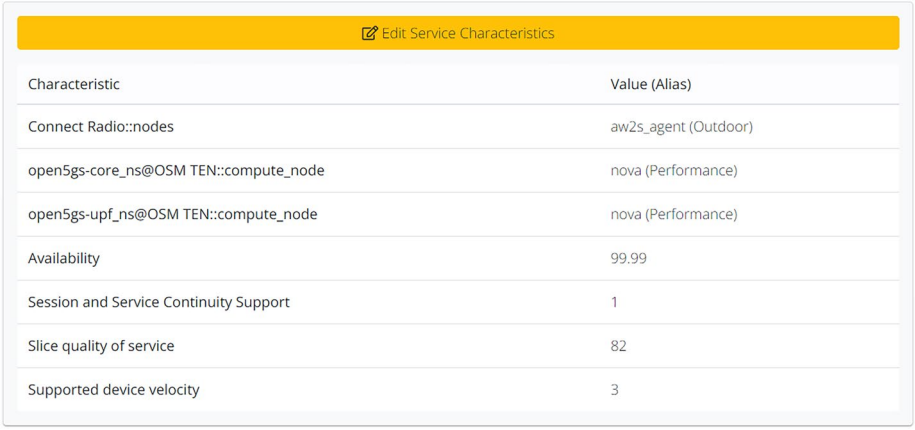

Fig. 7 Service implementation characteristics

three for few-shot training and three for user interaction. When the user first interacts with the GUI to provide the initial high-level description of the required network slice, four prompts are executed in sequence; three training prompts followed by one user prompt. Naturally, this results in an initial delay that is four times longer than the time for a single prompt. After this, the user negotiates the generated network slice once and confirms its deployment through two additional prompts.

Since user input delay cannot be precisely measured or consistently factored into the total process time, it has been excluded from the respective evaluation, which focuses solely on the model's response times. The experimentation's execution time results are presented in Table 2, while the typical limits on requests per minutes (RPMs) and tokens per minutes (TPMs) for the remote models are shown in Table 3. These limits are crucial for our experimentation, as exceeding them causes the active session to break and requires a restart. Another key aspect is that managing these limits is the responsibility of the integrator, which complicates the process with the need to introduce artificial delays or perform token trimming. The network deployment time across the iterations varied between 432 and 554 s, so an average was used for comparison to ensure that this metric did not affect the analysis.

The results indicated a significant 54-second deviation in response times between the older online model (GPT-3.5-turbo) and the local model (Mistral). This disparity is

Table 2 Experimentation's execution time comparison

	Mistral	**GPT-3.5-turbo**	**GPT-4***
Model's aggregated response time (s)	12	66	12
Network deployment time (s)	502	502	502
Overall time (s)	514	568	514

Table 3 Models' API usage restrictions

	Mistral	**GPT-3.5-turbo**	**GPT-4***
TPM	–	40,000	30,000
RPM	–	3	500

attributed to the RPM limit imposed on GPT-3.5-turbo, as shown in Table 3. The low RPM threshold was quickly reached with just the initial three few-shot training prompts, forcing a mandatory 1-minute delay before continuing with the user's codesign and acceptance prompts, which added considerable time to the process. It is important to note that this difference was not due to the local model being inherently superior, but rather because it is not subject to usage restrictions like the online models. Even a scenario that combined the few-shot training prompts into one (reducing the total experimentation to four prompts) would not have resolved this issue without compromising the co-design process, which is a key advantage of LLMs and a reason to employ them in the process.

On the other hand, the response times for the newer GPT-4 and the local model were nearly identical. While GPT-4's higher RPM limit was sufficient for our example, its lower TPM limit compared to its predecessor could have posed similar issues in more complex scenarios with extensive few-shot training data or enhanced co-design aspect. Furthermore, it is also important to note the online models' API usage requires a paid subscription, further aggravated by usage restrictions and tiers.

Although commercial online models, such as the above, excel at generalizing context, their ability to adapt to specific domains is limited, unless pre-trained on relevant datasets. Their only means of acquiring domain-specific expertise is by few-shot training and often require paid subscriptions and enforce strict limits on requests and token usage, which can result in notable response delays and increased costs unless higher-priced, unlimited-use packages are selected. Such factors can severely impact the sustainability of long-term experimentation, particularly in cost-sensitive environments. Last but not least, working with models with such a large number of parameters comes with a substantial cost of computational resources, and due to that, a significant environmental impact.

This highlights the emerging need to transition to smaller, more resource-efficient models that minimize energy consumption. However, this shift must ensure that the quality and accuracy of the responses are not compromised. In this context, local LLMs offer a more flexible alternative, as they can be fine-tuned or further trained on domainspecific data, making them better suited for specialized tasks, while maintaining the accuracy of a larger model. Even though the initial setup of a local model requires investment in resources like GPUs and RAM, the cost becomes comparable to that of commercial models after about three months of daily use. Local models also provide immediate responses when supported by adequate hardware and avoid the limitations of tokens and requests. Additionally, with local models, updates and adjustments are fully controlled, ensuring consistent performance and the ability to implement incremental improvements. This control is especially valuable in ensuring that the model's outputs align with domain-specific needs over time.

6 Conclusions

This work defined and demonstrated that the realization of NaaS could be significantly improved by utilizing GenAI models in conjunction with the adoption of standardized network APIs. The proposed framework enhances digital inclusion by allowing users to express traditional network requirements in a more intuitive and

flexible manner, while maintaining seamless interoperability with next-gen OSSs that ultimately deliver the described NaaS.

Furthermore, this study evaluates the framework's performance by comparing state-of-the-art remote and local LLMs in terms of overall execution time. The results of this work seem also to converge with the sustainable development goals (SDGs) to reduce energy consumption and promote cost efficiency and sustainability, as they indicate that local LLMs with fewer parameters can match or even outperform their resource-intensive online counterparts in specialized tasks, while simultaneously lowering operation and maintenance costs.

For future work, we plan to leverage local LLMs to expand towards the multimodal input provided to the framework. While the current implementation can effectively process text-based, high-level input or network-related JSONs, expressed in text format, we aim to extend its capabilities to include a broader range of data formats, such as images or entire project folders. This extension will enable not only the configuration and provision of a network supporting a user's application but also the deployment of the application itself, further enhancing user experience and promoting digital inclusion.

List of Abbreviations

3GPP	3rd Generation Partnership Project
6G	Sixth-Generation
AI	Artificial Intelligence
API	Application Programmable Interface
CLI	Command Line Interface
eMBB	Enhanced Mobile Broadband
ETSI	European Telecommunications Standards Institute
GenAI	Generative Artificial Intelligence
GPU	Graphics Processing Unit
GSMA	GSM Association
GST	General Slice Template
GUI	Graphical User Interface
IBN	Intent-Based Networking
IoT	Internet of Things
KPI	Key Performance Indicator
KVI	Key Value Indicator
LCM	Lifecycle Management
LLM	Large Language Model
ML	Machine Learning
mMTC	Massive Machine-Type Communications
NaaS	Network-as-a-Service
NEST	Network Slice Template next-gen next-generation
NFV	Network Function Virtulization
NLP	Natural Language Processing
NSD	Network Service Descriptor
OAM	Administration and Maintenance
ONAP	Open Network Automation Platform
ONOS	Open Network Operating System
OSS	Operational Support System
PoC	Proof of Concept
RPM	Requests per minute
SDG	Sustainable Development Goal
SDG OSL	Software Development Group for OpenSlice
SDN	Software-Defined Networking
SDO	Standards Development Organization
SLA	Service Level Agreement
TMF	TeleManagement Forum
TPM	Tokens per minute
UPF	User Plane Function
URLLC	Ultra-Reliable and Low Latency Communications

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13638-025-02472-x.

Additional file 1.

Acknowledgements
The authors of this paper are grateful to the personnel of the Patras5G laboratory at the University of Patras and the p-NET Competence Center for the provision of necessary resources and technical support to conduct the experiments.

Author contributions
KT devised the original idea, drafted the manuscript, and served as the editor. DB conducted the AI-related research and collaborated with BA on its implementation. GCT contributed to the related work section. CT introduced the multimodal concept and reviewed the paper. SD and AB supervised the research work and reviewed the paper.

Funding
The work of the authors has been partially supported by the Horizon Europe Projects ACROSS (Grant Agreement No. 101097122), FIDAL (Grant Agreement No. 101096146), and P2CODE (Grant Agreement No. 101093069).

Availability of data and materials
The datasets generated and/or analysed during the current study are not publicly available due to a concurrent effort to further enrich and publish them, but are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 29 October 2024 Accepted: 6 May 2025
Published online: 03 June 2025

References

1. W.E.A. Kellerer, Adaptable and data-driven softwarized networks: review, opportunities, and challenges. Proc. IEEE **107**(4), 711–731 (2019). https://doi.org/10.1109/JPROC.2019.2895553
2. M. Masoudi, M.G. Khafagy, A. Conte, A. El-Amine, B. Francoise, C. Nadjahi, F.E. Salem, W. Labidi, A. Sural, A. Gati, D. Bodere, E. Arikan, F. Aklamanu, H. Louahlia-Gualous, J. Lallet, K. Pareek, L. Nuaymi, L. Meunier, P. Silva, C. Cavdar, Green mobile networks for 5g and beyond. IEEE Access **7**, 107270–107299 (2019). https://doi.org/10.1109/ACCESS.2019.2932777
3. D. Brodimas, K. Trantzas, B. Agko, G.C. Tziavas, C. Tranoris, S. Denazis, A. Birbas.Towards intent-based network management for the 6g system adopting multimodal generative AI, in *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)* (2024), pp. 848–853. https://doi.org/10.1109/EuCNC/6GSummit60053.2024.10597022
4. A. Clemm, L. Ciavaglia, L.Z. Granville, J. Tantsura. Rfc 9315: Intent-based networking concepts and definitions (2022)
5. Li, C., Havel, O., Olariu, A., Martinez-Julia, P., Nobre, J., Lopez, D.: Rfc 9316: Intent classification (2022)
6. L. Pang, C. Yang, D. Chen, Y. Song, M. Guizani, A survey on intent-driven networks. IEEE Access **8**, 22862–22873 (2020). https://doi.org/10.1109/ACCESS.2020.2969208
7. A. Leivadeas, M. Falkner, A survey on intent-based networking. IEEE Commun. Surv. Tutor. **25**(1), 625–655 (2023). https://doi.org/10.1109/COMST.2022.3215919
8. ETSI GS ENI: System Architecture (2023). https://www.etsi.org/deliver/etsigs/ENI/001099/005/03.01.0160/gseni005v030101p.pdf Accessed 11 October 2024
9. ETSI GR ZSM:Intent-driven autonomous networks;Genericaspects (2023). https://www.etsi.org/deliver/etsigr/ZSM/001099/011/01.01.0160/grZSM011v010101p.pdf Accessed 11 October 2024
10. TM Forum: Intent in Autonomous Networks V1.3.0 (IG1253) (2023). https://www.tmforum.org/resources/introductory-guide/ig1253-intent-in-autonomous-networks-v1-3-0/ Accessed 11 October 2024
11. Open Network Operating System (ONOS): (2015). https://opennetworking.org/onos/ Accessed 11 October 2024
12. R.A. Addad, D.L.C. Dutra, M. Bagaa, T. Taleb, H. Flinck, M. Namane. Benchmarkingthe ONOS Intent interfaces to ease 5G service management (2022). https://arxiv.org/abs/2201.01407
13. OpenDaylight: (2014). https://www.opendaylight.org/ Accessed 11 October 2024
14. Open Network Automation Platform (ONAP): (2017). https://www.onap.org/ Accessed 11 October 2024
15. ONAP: Support for Intent Framework and Intent Modeling (2021). https://wiki.onap.org/display/DW/Support+for+Intent+Framework+and+Intent+Modeling Accessed 11 October 2024
16. M. Riftadi, F. Kuipers.P4I/O: Intent-Based Networking with P4, in *2019 IEEE Conference on Network Softwarization (NetSoft)* (2019), pp. 438–443. https://doi.org/10.1109/NETSOFT.2019.8806662
17. O. Friha, M. Amine Ferrag, B. Kantarci, B. Cakmak, A. Ozgun, N. Ghoualmi-Zine, Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. IEEE Open J. Commun. Soc. **5**, 5799–5856 (2024). https://doi.org/10.1109/OJCOMS.2024.3456549

18. K. Dzeparoska, J. Lin, A. Tizghadam, A. Leon-Garcia. Llm-based policy generation for intent-based management of applications, in *2023 19th International Conference on Network and Service Management (CNSM)* (2023), pp. 1–7. https://doi.org/10.23919/CNSM59352.2023.10327837
19. K. Dzeparoska, A. Tizghadam, A. Leon-Garcia, Emergence: an intent fulfillment system. IEEE Commun. Mag. **62**(6), 36–41 (2024). https://doi.org/10.1109/MCOM.001.2300270
20. A. Boutouchent, A.N. Meridja, Y. Kardjadja, A.M. Maia, Y. Ghamri-Doudane, M. Koudil, R.H. Glitho, H. Elbiaze, Amanos: An intent-driven management and orchestration system for next-generation cloud-native networks. IEEE Commun. Mag. **62**(6), 42–49 (2024). https://doi.org/10.1109/MCOM.003.2300367
21. GSMA: The Ecosystem for Open Gateway NaaS API development (2023). https://www.gsma.com/futurenetworks/wp-content/uploads/2023/05/The-Ecosystem-for-Open-Gateway-NaaS-API-development.pdf Accessed 11 October 2024
22. CAMARA project. https://camaraproject.org/ Accessed 11 October 2024
23. TM Forum: GB1022 ODA Functional Architecture Guidebook (2021). https://projects.tmforum.org/wiki/display/PUB/GB1022+ODA+Functional+Architecture+Guidebook+v1.1.0 Accessed 11 October 2024
24. ETSI SDG OpenSlice. https://osl.etsi.org/ Accessed 11 October 2024
25. GSMA: NG.116 - Generic Network Slice Template (2023). https://www.gsma.com/newsroom/wp-content/uploads/NG.116-v9.0.pdf Accessed 11 October 2024
26. C. Tranoris. Openslice: An opensource OSS for Delivering Network Slice as a Service. CoRR https://arxiv.org/abs/2102.03290 (2021)
27. Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: a survey on few-shot learning. ACM Comput. Surv. (2020). https://doi.org/10.1145/3386252
28. Y. Zhu, Y. Wang, J. Qiang, X. Wu, Prompt-learning for short text classification. IEEE Trans. Knowl. Data Eng. (2023). https://doi.org/10.1109/TKDE.2023.3332787
29. Patras5g. https://wiki.patras5g.eu/ Accessed 11 October 2024
30. OpenAI, al: GPT-4 Technical Report (2024). https://arxiv.org/abs/2303.08774
31. A.Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D.S. Chaplot, D. 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, W.E. Sayed. Mistral7B (2023). https://arxiv.org/abs/2310.06825
32. ETSI SDG OpenSlice : Lifecycle Management Rules. https://osl.etsi.org/documentation/latest/naas/lcmrulesintro/ Accessed 11 October 2024

Publisher's Note