Intent-Based 5G subscriber management: An MCP-Mediated Architecture with Human-in-the-Loop and Multi-Step LLM Benchmarking

Silvia Ruano Ferrer, Amr A. AbdelNabi, Jad Nasreddine, and Daniel Camps-Mur
i2CAT Foundation, Barcelona, Spain
{silvia.ruano, amr.abdelnabi, jad.nasreddine, daniel.camps}@i2cat.net

***Abstract*— Large Language Models (LLMs) are increasingly promoted as key enablers for Intent-Based Networking (IBN); however, their reliability in executing fragile control-plane operations remains insufficiently validated. This paper presents a Model Context Protocol (MCP)-mediated IBN framework connecting LLM reasoning engines to Open5GS-based 5G core through structured tool orchestration. Using this framework, we introduce a multi-step benchmark designed to evaluate the operational robustness of cloud-based frontier and locally deployable models under preservation-critical network-management constraints. The benchmark covers three task categories of increasing complexity: localized IP configuration, nested slice and APN updates, and subscriber-information modifications requiring strict preservation of unrelated state. The results show that while many models perform simple IP-management tasks with high accuracy, performance degrades substantially when executing multi-step operations over nested subscriber structures. In particular, locally deployable models exhibit recurrent failures in state preservation, often modifying unrelated fields while performing the requested updates. These findings indicate that reliable LLM-driven 5G automation is limited not only by natural-language understanding but also by the ability to perform deterministic, preservation-aware execution over complex telecom data structures. Overall, the benchmark highlights a critical gap between current LLM capabilities and the operational resilience required for production-grade IBN, while showing that cloud-based frontier models currently retain a measurable advantage in complex telecom configuration workflows.**

***Index Terms*—5G Core, Intent-Based Networking, Human-in-the-Loop, LLM Benchmark, Model Context Protocol, Open5GS.**

I. Introduction

The evolution of modern network infrastructures has shifted the management paradigm from imperative, manual workflows toward declarative Intent-Based Networking (IBN). Traditionally, configuring network functions requires granular intervention through Command Line Interface (CLI) commands or complex YANG model modifications using protocols such as NETCONF [1].

This work was supported by the EU's Horizon Europe research and innovation programme HORIZON-JU-SNS-2025-2 under the GENIUS project (Grant No. 101293076). It also received support from the COALESCE-6G project (PID2024-163028OB-I00), funded by MICIU/AEI/10.13039/501100011033/FEDER, EU.

As networks become increasingly heterogeneous, these manual approaches introduce a configuration gap characterized by operational latency and susceptibility to human errors. In 5G environments, incorrect subscriber-management operations may corrupt Access Point Name (APN) configurations, slice assignments, session parameters, or security credentials, leading to service disruption and control plane inconsistencies. IBN addresses this challenge by allowing operators to specify what networks should achieve rather than how individual parameters should be configured, reducing operational complexity while supporting dynamic service requirements [2]. However, telecom workflows require strict preservation of existing configurations during updates, making reliable execution significantly more challenging than simple intent translation.

Within this context, Large Language Models (LLMs) provide an intent interpretation layer capable of converting natural language requests into structured network actions. Compared with rule-based approaches, LLMs better handle linguistic variability and can generate candidate configuration parameters and service-based Application Programming Interface (API) requests. Nevertheless, many telecom operations require sequential retrieval, validation, modification, and verification procedures. Although recent LLMs show strong intent understanding capabilities, their reliability decreases during preservation-aware multi-step execution requiring state consistency, tool coordination, and deterministic configuration updates. This motivates architectures that combine LLM reasoning with controlled tool mediation and human oversight instead of relying on unconstrained generation [3].

To address these limitations, this work adopts the Model Context Protocol (MCP) as an orchestration layer between the reasoning engine and telecom infrastructure. MCP enables structured tool discovery, controlled API invocation, and execution mediation, allowing LLMs to operate within constrained operational boundaries. Through this interface, natural language intents are mapped into validated configuration payloads and 3GPP service based API calls [4].

Recent MCP-based benchmarks evaluate whether agents can discover tools, coordinate multi-step execution, and ground decisions using intermediate results. Similarly, studies on

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

multi-step reasoning highlight the importance of planning and iterative verification to reduce hallucinations and improve execution reliability [5]. In addition, Human-in-the-Loop (HiL) validation provides operator oversight during uncertain or preservation-sensitive actions, allowing the LLM to operate within a controlled execution loop that includes tool mediation, state validation, and verification mechanisms [6], [7]. Future extensions may further incorporate Network Digital Twin (NDT) validation to test configurations before deployment [8].

Autonomous network management is increasingly leveraging operational LLM-based architectures that combine reasoning models, standardized tool interfaces, and robust validation mechanisms to ensure system safety. Bridging the gap between intent and execution, this work develops an MCP-mediated framework designed for preservation-aware telecom intent processing over an Open5GS 5G core. Specifically, this paper makes the following contributions:

- **MCP-Mediated Preservation-Aware IBN Framework**: We design an architecture that integrates multiple LLM models with Open5GS through structured MCP-based orchestration for telecom control plane management, supporting subscriber workflows such as IP configuration, slice/APN updates, security modifications, and multi-step state-preserving operations.
- **Multi-Step Performance and Resiliency Benchmarking**: We evaluate frontier and open-source LLMs across different task complexities, comparing zero shot execution against in context learning/training using few shot operational examples. The results show that, despite improvements from prompt based guidance, current frontier models remain far from achieving the operational resiliency required for telecom control plane management, especially in complex multi-step and state preserving workflows. This direct comparison isolates the impact of prompt based guidance versus foundational model capacity in solving complex workflows, following GSMA Open-Telco LLM Benchmark principles.
- **Operational Reliability Analysis:** We provide structured evaluation artifacts and analyze execution failures related to workflow complexity, state preservation, endpoint selection, and tool-use reliability. To support the reproducibility of telecommunication LLM benchmarks, we openly publish our standardized input datasets and the corresponding absolute ground-truth outputs (the evaluation spreadsheets) as a benchmark artifact. The benchmark datasets and evaluation artifacts are publicly available at https://gitlab.i2cat.net/areas/mwi/agentic/intent-based-5g-subscriber-management.

This work positions telecom-oriented LLM agents beyond natural language interfaces, considering them operational reasoning systems that must execute preservation sensitive control plane workflows under strict reliability constraints.

II. State-of-the-Art

The transition toward IBN aims to reduce operational complexity by allowing operators to specify high level objectives instead of manually configuring low level network parameters [1]. Recent advances have explored the use of LLMs bridging natural language intents and formal network actions. Manias et al. demonstrated customized LLMs for intent extraction in 5G core networks, translating operator requests into structured management schemas [2]. Similarly, benchmarks such as NetConfEval evaluate the ability of LLMs to generate network configurations and device level actions from high level requirements [4], while Net2d LLM integrates execution validation through NDT-based verification [8].

Although single-step intent translation has significantly advanced, reliable telecom automation requires multi-step reasoning, state tracking, and consistent execution. Existing LLMs remain challenged by long execution chains and hallucination risks [5]. To improve reasoning reliability, approaches such as Q* introduce deliberative planning strategies [5], while PromptChainer decomposes complex workflows into chained subtasks for more structured execution [6]. Furthermore, while contemporary inference techniques introduce natively ”constrained decoding” or ”JSON-mode” generation to enforce structural adherence, these mechanisms remain fundamentally limited by their inability to evaluate deep semantic constraints. They can guarantee syntactical validity (e.g., matching a schema’s structural keys), but they cannot prevent internal parameter hallucinations or unintentional erasure of unrelated, nested array elements, a limitation that directly drives state preservation failures during complex configuration re-writes.

Beyond reasoning alone, autonomous network management requires LLM agents to interact with external tools, APIs, and dynamic network states. The MCP provides a standardized interface for tool discovery and invocation, enabling structured interaction between reasoning models and operational environments. MCP-Bench highlights the importance of evaluating tool coordination and multi-step execution through live MCP servers [3]; however, recent studies also show that MCP-based integration introduces challenges related to instruction compliance, execution reliability, and orchestration overhead [7].

In telecommunications, MCP based approaches are emerging to support AI native network control. Tele-LLM-Hub introduces a domain-specific TeleMCP framework for multi agent coordination in 5G and O-RAN systems [9], while MCP-based Internet of Experts (IoX) enables LLMs to interact with deterministic wireless expert models [10]. Furthermore, dual MCP architectures with HiL authorization have been proposed to improve operational safety in edge network reconfiguration scenarios [11].

Despite these advances, existing works mainly address generic MCP benchmarking, intent extraction, or network coordination, leaving preservation aware telecom execution largely unexplored. In particular, limited attention has been given to integrating MCP-mediated LLMs with operational 5G core databases such as Open5GS for subscriber management involving nested configurations, state preservation, and multi step workflows. Moreover, systematic evaluations comparing zero shot and contextually trained LLMs under realistic telecom configuration tasks remain limited. Addressing this gap

requires architectures that combine structured intent interpretation, MCP-mediated execution, and operational validation to enable reliable LLM-driven 5G core management.

III. Dataset and Benchmark Design

This work uses datasets from the GSMA Open-Telco LLM Benchmarks initiative [12] and adapts them to executable Open5GS subscriber-management workflows. The resulting benchmark adds Open5GS-specific tool mappings, multi-step execution, state-preservation requirements, and final-state validation, evaluating whether an LLM can translate natural-language intents into correct API actions while preserving subscriber-state consistency.

A. Design Objectives

The benchmark evaluates three key capabilities required for AI-native telecom agents:

- **Intent-to-endpoint mapping**: Mapping operator intents to the correct executable API procedures.
- **State preservation**: Updating requested parameters while preserving unrelated subscriber configurations.
- **Sequential tool-use reasoning**: Executing dependent multi-step workflows through chained API operations.

B. Operational Workflows and Data Structure

The benchmark covers practical 5G subscriber-management workflows with different levels of operational complexity and state-preservation risk. We group them into three tiers:

- **Localized IP Management (Low Complexity)**: This tier handles straightforward, localized workflows such as IPv4 and IPv6 address assignment. Because these tasks generally only require modifying explicit schema fields, the operational reasoning burden is low, and the risk of collateral network damage is minimal.
- **Nested Slice and APN Management (High Complexity)**: These workflows involve network-slice provisioning and updates, and APN configuration management. They represent a significantly higher operational risk, forcing the system to selectively modify deeply nested JSON structures (like sst and sd identifiers) while perfectly preserving large portions of unrelated subscriber configurations exactly as they were.
- **Critical Subscriber Modifications (Severe Complexity)**: This tier represents the most fragile operations in the pipeline, such as security credential rotation (K, OP, OPc), AMBR modification, and Packet Data Network (PDN)-type updates. These operations demand strict preservation semantics; even minor execution errors can easily corrupt unrelated APN configurations, slice assignments, or IP addresses.

To practically evaluate these tiers, the benchmark utilizes highly structured tabular datasets (inspired by NetBench dataset [13]) that dictate a rigid state-machine logic for each task. Each dataset row contains structured fields used during evaluation and operational analysis:

- **Command** and **Context**: The natural-language telecom operator request and any supplementary operational context.
- **Answer**: The exact expected final subscriber JSON state, which serves as the absolute operational ground truth. During evaluation, the generated subscriber configuration is compared against the expected reference state using operationally relevant attributes, including IMSI, security credentials, IPv4/IPv6 configuration, AMBR parameters, slice descriptors, APN values, subscriber status, and PDN type.
- **IMSI**: Subscriber identifier.
- **Key/OPC**: Subscriber security credentials.
- **IPv4/IPv6**: IP address configuration parameters.
- **APN/sst/sd**: Slice and session-management parameters.
- **Steps**: A critical quantifier of procedural complexity, indicating the number of sequential API/tool invocations (ranging from 1 to 4 steps) required to safely execute the target configuration. Single-step tasks typically involve direct endpoint invocation, whereas multi-step tasks require chained operations such as subscriber retrieval, deletion, recreation, and sequential configuration updates.
- **Train**:A reference API/tool-call sequence that may be provided as an in-context example in the few-shot setting. No model parameters are updated during evaluation.
  A simplified training prompt structure is shown below:
  Intent: Assign subscriber `001010000000017` an unused IP.
  Tools: `get_all_subscribers()` → `config_ip(001010000000017,111.111.111.112).`
  Target: new subscriber-management request.
  Such demonstrations expose the expected workflow but may induce schema rigidity when models reuse irrelevant fields or values.
- **Task Family**: Fine-grained subtype label used for error analysis.
  The demonstration exposes the expected operational sequence and tool usage before the target query. However, demonstrations containing detailed parameter values or nested configurations may induce schema rigidity, causing some models to reuse irrelevant fields or values when processing subsequent requests.
- **Task Family**: Fine-grained subtype label used for error analysis.

To ensure rigorous and reliable evaluation, we employ two verification strategies depending on the operational intent of the command: **Response-Level Retrieval Verification** for read-only operations and **Post-Execution State Verification** for state-modifying operations. This distinction is necessary because relying solely on an LLM's natural-language output is insufficient for validating network modifications; an LLM may report successful execution even if the underlying API call failed, the JSON payload was malformed, or unrelated state variables were corrupted. Therefore, the evaluation separates

read-only retrieval tasks from active configuration changes:

- **Response-Level Retrieval Verification:** Used for read-only tasks, where success depends on correctly returning the requested information in the chat interface. The LLM's final response is matched against the expected value stored in the *Answer* field using flexible pattern matching. For example, for the query "What slice is subscriber 001010000000011 on?", the response is considered correct if it contains the expected values `[sst: 4, sd: TELEMACS]`, regardless of minor formatting differences.
- **Post-Execution State Verification:** Used for state-modifying tasks, where LLM confirmation alone cannot prove successful execution. After the LLM completes the requested operation, `get_subscriber()` is invoked directly via MCP, bypassing the LLM, to retrieve the live subscriber state and compare it with the expected JSON ground truth. For example, after the request "Update the IP to 88.44.22.119 for subscriber 001010000000013", the evaluator retrieves the subscriber record and verifies that the IP was updated to `88.44.22.119` while all unrelated configuration fields remain unchanged. This also captures failures caused by unsuccessful API calls, malformed payloads, or unintended state corruption.

C. REST-Based Subscriber-Management Framework

During execution, the LLM must interact with a REST-based subscriber-management framework inspired by Open5GS operational procedures. The system is evaluated on its ability to extract telecom parameters and generate correct sequential executions targeting specific API endpoints, including:

- `/subscriber/add`
- `/subscriber/get/{imsi}`
- `/subscriber/delete/{imsi}`
- `/subscriber/config_ip/{imsi}/{ip}`
- `/subscriber/config_ip6/{imsi}/{ip}`
- `/subscriber/config_type/{imsi}/{type}`
- `/subscriber/update_slice/{imsi}/{apn}/{sst}/{sd}`
- `/subscriber/update_apn/{imsi}/{apn}/{slice_index}`
- `/subscriber/update_ambr/{imsi}`
- `/subscriber/update_status/{imsi}/{status}`

These APIs emulate realistic telecom operational workflows and require endpoint selection, telecom parameter extraction, sequential reasoning, and state-aware subscriber modification. Several benchmark tasks additionally require preserving existing subscriber configurations while modifying only the requested parameters.

D. Dataset Distribution

The complete benchmark contains 163 task instances divided into training and testing subsets. Table I summarizes the dataset composition.

TABLE I: Dataset Distribution Across Categories

Category	Train	Test	Total
IP	28	23	51
Slice	50	37	87
Change_info	0	25	25
Total	78	85	163

For the IP and Slice categories, the benchmark is divided into demonstration and evaluation subsets. In the few-shot setting, examples from the demonstration subset are provided as in-context demonstrations, while the evaluation subset is used for testing. In contrast, all 25 `Change_info` instances are used as evaluation queries, and no in-context demonstrations are provided for this category. No model parameters are updated in any setting.

The *Slice* category represents the largest subset due to the diversity of slice and APN configurations. In contrast, the *Change_info* category exhibits the highest procedural complexity because many tasks require chained subscriber-management operations.

Task complexity distribution is as follows: 58 tasks requiring 1 step, 58 tasks requiring 2 steps, 43 tasks requiring 3 steps, and 4 tasks requiring 4 steps. The distribution demonstrates that the benchmark contains both simple endpoint-selection tasks and higher-complexity operational workflows requiring sequential reasoning across multiple dependent API actions. Consequently, the benchmark evaluates both endpoint-selection accuracy and procedural reasoning capability under varying operational complexity levels.

In summary, the proposed benchmark provides a realistic evaluation environment for telecom-oriented LLM agents operating under practical network-management constraints. By combining telecom-specific reasoning, API execution, state-aware configuration management, and procedural-complexity analysis, the benchmark aligns with the objectives of the GSMA Open-Telco LLM Benchmarks initiative toward reproducible evaluation of AI systems for future telecom automation.

IV. Architecture

The proposed architecture follows a Client-Host-Server model mediated by the MCP, as illustrated in Fig. 1. The framework bridges natural language telecom intents and executable 5G Core operations by combining LLM-based reasoning, structured MCP tool orchestration, and Open5GS based network control.

The *User Interface Layer* enables operators to express network-management objectives using natural-language commands instead of low-level configuration procedures. Within this layer, the chatbot agent is the architectural hub of the

system. It maintains simultaneous connections to two external services: the LLM and the MCP server. It coordinates all data flow between them. At startup, the agent establishes a Server-Sent Events (SSE) connection to the MCP server, completes the MCP handshake, and retrieves the available tool manifest. A transient conversation is then created with the LLM model and defines the assistant's role and operational rules. In addition, the agent retrieves the current tool schemas from the MCP server, reformats them, and submits them to the LLM model.

The submitted intent is processed by the *LLM Model* that performs intent interpretation, telecom parameter extraction, workflow selection, and generation of the required tool invocation sequence. Finally, it returns a structured plan of one or more tool calls. This plan is presented to the client for review before any execution occurs. The latter may approve the plan, cancel it, or refine individual operations through natural language instructions, which are resolved into modified tool calls via a secondary constrained LLM call. The framework supports different frontier and open-source LLM models and enables preservation aware multi-step reasoning for operations requiring subscriber retrieval, modification, and validation.

The *MCP Server* decouples the reasoning engine from direct network access through an MCP client/server architecture based on structured JSON-RPC communication [14]. The MCP client, integrated in the chatbot agent, manages context and execution coordination, while the MCP server exposes controlled telecom tools for subscriber provisioning, retrieval, APN updates, slice modification, and 5G Core API interactions. In this work, MCP is used as the common orchestration and tool-access layer across all evaluated models; the evaluation does not isolate MCP as an independent source of performance improvement.

The *API Management layer* provides API management functions including request validation, protocol translation, and execution mediation between MCP tools and telecom infrastructure. Finally, the *Core Network Layer* integrates an Open5GS based 5G Core. Subscriber operations are applied through `Open5GS DBCTL`, which interacts with the subscriber database. Although the architecture supports HiL validation, the experimental evaluation in this work focuses on autonomous LLM execution and does not independently quantify the effect of human intervention.

Overall, the architecture enables intent-based telecom control, MCP-mediated execution, and benchmarking of LLM operational reasoning under realistic 5G subscriber-management workflows.

V. Performance Evaluation Results

Fig. 2 presents the accuracy results of the evaluated models under both training assisted and no training conditions across the three telecom operational task categories: localized IP management (*IP Tasks*), nested slice and APN management (*Slice Tasks*), and critical subscriber modifications (*Change_info Tasks*). Accuracy is defined as the percentage of benchmark tasks for which the final subscriber state exactly

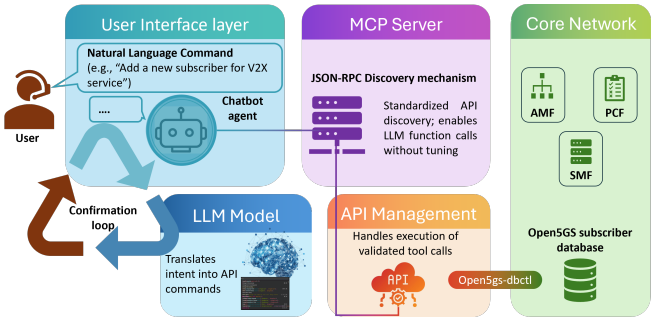


Fig. 1: System architecture showing the interaction between the MCP components and the 5G core [4].

matches the expected reference output and is computed as the ratio of correctly completed tasks to the total number of tasks within the corresponding category. Each benchmark query set was independently executed five times.[1]

To facilitate interpretation of the results, the evaluated models are grouped into two categories. The first category, referred to as ***Cloud-Based Frontier Models***, includes *claude-sonnet-4-6*, *gemini-2.5-pro*, *gemini-3-flash-preview*, *gemini-3.1-pro-preview-customtools*, *gpt-4o-mini*, *gpt-5.5*, *llama-3.3-70b-instruct*, *deepseek-v3.2:671b*, and *qwen3-235b-a22b-instruct*. These models represent large-scale frontier systems accessed through cloud-based or externally hosted inference environments. The second category, referred to as ***Locally Deployable Models***, includes *gpt-oss-120b*, *gpt-oss-20b*, *granite4*, *mistral-medium-3*, *mistral-nemo:12b*, *mistral-small:3.1b*, *mistral-small3.2b*, *gemma3-tools*, *qwen2.5-coder:32b*, and *qwen3-coder:30b*. These models represent systems that can be deployed in local, private, or self-hosted environments, making them relevant for privacy-sensitive telecom operations and controlled network-management deployments. In many of these, the letter "b" and the number preceding it indicate the model's size in billions of parameters, though several vendors choose to keep these exact figures proprietary and have never publicly disclosed the parameter counts for models like Claude 3.5 Sonnet or Gemini 1.5 Pro.

Overall, three consistent trends emerge across all evaluated models. First, training assisted prompting improves performance across most task categories, indicating that example driven guidance helps models learn the operational workflow required to complete telecom management tasks. The average accuracy increases by 8.9% (resp., 39.8%) when using training with ***Locally Deployable Models*** in the *IP* category (resp., *Slice* category). These gains become 10.5% (resp., 16.6%) with ***Cloud-Based Frontier Models***. It can be noted that the training gain is large in absolute accuracy terms in ***Locally Deployable Models***, especially for complex queries such as in the *Slice tasks*. Second, accuracy decreases as task complexity and workflow depth increase, particularly for operations requiring

[1]For each model and evaluation setting, accuracy is reported as the mean across the five runs. The 95% confidence intervals are computed from the variation in accuracy across these runs using Student's t-distribution.



Fig. 2: Accuracy by required steps for *IP*, *Slice*, and *Change_info* tasks under training and no training settings with the bars showing mean accuracy with 95% confidence intervals; shading separates ***Cloud-Based Frontier Models*** and ***Locally Deployable Models***.

multiple sequential API interactions and intermediate state retrieval. Third, the performance gap between ***Cloud-Based Frontier Models*** and ***Locally Deployable Models*** becomes more pronounced for tasks involving modifications to nested subscriber configurations and preservation of existing subscriber state. In Fig. 3 we show the impact of the number of parameters in a model on the average accuracy depending on the number of steps. For the massive ***Cloud-Based Frontier Models***, we combined them in one row called Cloud APIs showing the average over these models. As it can be seen in the figure, the general trend is that the accuracy decreases when the number of parameters decreases and/or the number of steps increases. The special cases are related to the different models from different vendors. These findings suggest that telecom LLM agent performance is constrained primarily by reliable state preserving execution rather than telecom intent understanding alone. While most models successfully identify the requested telecom operation and select a plausible execution path, failures frequently occur during workflow execution. In particular, many errors arise from state preservation failures, where a model correctly applies the requested modification but unintentionally alters, removes, or fails to preserve unrelated subscriber configuration parameters, resulting in an incorrect final subscriber state. Additional failure modes include omitted intermediate steps, incorrect API sequencing, incomplete configuration updates, and loss of contextual information across multi-step procedures. Collectively, these results indicate that robust state management and procedural consistency remain the principal challenges for autonomous telecom configuration agents.

1) IP Tasks: The *IP Tasks* category achieves the highest accuracy among all evaluated task families because the requested modifications are limited to a small set of explicit subscriber attributes. These tasks primarily involve IPv4 address assignment, IPv6 address assignment, PDN session type configuration, and subscriber creation with predefined network settings.

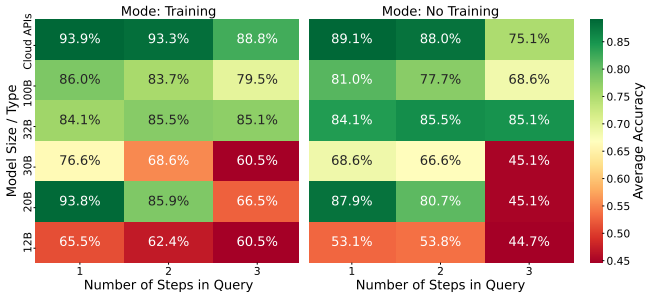


Fig. 3: Heatmap of the impact of models' number of parameters and query complexity (i.e., number of steps) on the accuracy w/wo training.

Representative benchmark operations include assigning unique IPv4 addresses, allocating unused IPv6 addresses, creating subscribers with predefined address configurations, and modifying PDN session types between IPv4-only (`type=1`), IPv6-only (`type=2`), and IPv4v6 (`type=3`) operation modes. Most IP tasks correspond to one step and two step workflows. Direct address assignment typically requires a single API operation, while unique address allocation requires retrieval of existing subscriber information followed by address selection and configuration. Because these operations affect only a limited number of subscriber parameters, the mapping between the natural language request and the required API sequence is relatively direct. Consequently, IP tasks place less emphasis on complex state preservation and multi-step workflow execution than the other benchmark categories. Although most IP workflows are relatively simple, the observed failures are concentrated in retrieval and update procedures requiring address validation and subscriber state verification prior to configuration updates.

The numerical results reflect this lower operational complexity. Figure 2(a) presents the training assisted results, where *gemini-3-flash-preview* and *gemini-3-pro-preview* achieve perfect accuracy (100.00%), followed by *claude-sonnet-4-6* (99.13%), *gemini-2.5-pro* (95.65%), and *gpt-5.5* (94.14%). Figure 2(b) presents the no training results, where *gemini-2.5-pro* achieves the highest accuracy (97.39%), followed by *claude-sonnet-4-6* (95.65%) and *gemini-3-flash-preview* (93.91%). Among the ***Locally Deployable Models***, *mistral-medium-3* and *gpt-oss:20b* achieve the strongest IP performance; notably, *gpt-oss:20b* massively outperforms its much larger sibling, *gpt-oss:120b* (90.20% vs. 28.63% with training), demonstrating remarkable parameter-efficiency and superior adherence to JSON schema formatting during tool execution.

Failure analysis reveals that the remaining errors are predominantly operational rather than semantic in nature. Even the highest-performing models occasionally fail when address allocation requires retrieval and validation of existing subscriber state before configuration updates. Typical failure modes include selecting an incorrect configuration endpoint (e.g., invoking an IPv4 update instead of an IPv6 update), confusion between PDN session types, failure to verify address uniqueness before assignment, and inconsistencies when updating or reconstructing subscriber configurations. Among the ***Cloud-Based Frontier Models***, *qwen3-235b-a22b-instruct* performs strongly overall, with failures largely confined to no-training unique-IP allocation, subscriber-creation workflows, and IPv4/IPv6 session-type edge cases. Similarly, *llama-3.3-70b-instruct* generally identifies the correct subscriber-management operation but exhibits a higher frequency of failures involving unique IP allocation, PDN-type modifications, and subscriber-state preservation. *deepseek-v3.2* also demonstrates strong IP performance, achieving 94.78% accuracy in the no-training setting and 89.57% with training. Detailed analysis shows that training improves the most complex three-step IP workflows, achieving 100% accuracy, but reduces performance on simpler one-step and two-step tasks, resulting in a higher overall no-training score. The observed failures are concentrated in unique-IP allocation and retrieval-based configuration workflows rather than direct address assignment tasks. The observed performance degradation on simpler tasks when using training-assisted prompting, particularly notable in models like *deepseek-v3.2*, can be attributed to **in-context negative transfer**. Analysis of excess failures reveals that autoregressive LLMs frequently overfit to the complex JSON schema provided in the few-shot examples, treating them as rigid templates rather than flexible guides. Consequently, models aggressively paste inappropriate configurations into simpler tasks, leading to parameter hallucination (e.g., copying explicit hex keys from the training prompt), structural cross-contamination (e.g., forcing nested IPv6 session formatting onto IPv4 requests), and the unintended deletion of necessary nested fields that were absent from the training template. Thus, while training examples successfully guide complex multi-step workflows, their induced schema rigidity actively impairs performance on simpler, isolated updates.

Among the ***Locally Deployable Models***, *gpt-oss:20b* and *mistral-medium-3* correctly identify the required operation in most cases but exhibit more execution-related errors than the leading frontier models. In contrast, weaker models including *mistral-small3.1*, *mistral-nemo:12b*, and *gemma3-tools* frequently fail to complete the required workflow correctly, suggesting limitations in procedural execution rather than telecom intent interpretation.

The strongest models, particularly *gemini-3-flash-preview*, *gemini-3-pro-preview*, *gemini-2.5-pro*, *claude-sonnet-4-6*, *qwen3-235b-a22b-instruct*, and *deepseek-v3.2*, demonstrate highly reliable performance across both training and no training settings. The relatively small performance gap between frontier and locally deployable models indicates that most IP Tasks primarily evaluate endpoint selection, parameter extraction, and basic workflow execution rather than advanced procedural reasoning.

These findings suggest that IP related subscriber-management tasks are largely solved from an intent understanding perspective. The high accuracy achieved by both frontier and several locally deployable models indicates that endpoint selection and parameter extraction

are generally well handled. The remaining failures originate primarily from execution level issues, including incorrect API selection, address validation errors, and occasional state preservation failures. Consequently, further improvements are likely to depend more on workflow validation, deterministic tool execution, and state management mechanisms than on enhanced telecom domain reasoning capabilities.

2) Slice Tasks: The *Slice Tasks* category is substantially more challenging than the *IP Tasks* category because the requested modifications occur within deeply nested subscriber structures. These tasks require manipulation of APN definitions, slice/service types (`sst`), slice descriptors (`sd`), session configurations, QoS parameters, and AMBR values while simultaneously preserving unrelated subscriber information. Representative benchmark operations include creating subscribers for services such as `AGRICULTURE`, `AUTONOMOUS_DRIVING`, and `DRONE_DELIVERY`, adding new APN entries; modifying existing APN configurations; updating slice/service types and slice descriptors; creating additional slices; and modifying slice specific session parameters. Unlike IP-related operations, Slice Tasks require simultaneous preservation of security credentials (`k`, `opc`), APN definitions, QoS parameters, AMBR values, and previously configured slices. Most slice tasks correspond to two step and three step workflows. Successful execution frequently requires retrieval of the existing subscriber profile, identification of the target APN or slice entry, execution of the requested modification, and verification that unrelated fields remain unchanged. Because modifications are applied to nested subscriber structures, failures become significantly more likely when the model does not correctly preserve the existing subscriber state.

The numerical results reflect this increased operational complexity. Figure 2(c) presents the training-assisted results, where *gemini-3-flash-preview* achieves the highest Slice accuracy (94.05%), followed by *gemini-3-pro-preview* (89.19%), *claude-sonnet-4-6* (86.49%), and *gpt-4o-mini* (83.87%). Figure 2(d) presents the no-training results, where *claude-sonnet-4-6* achieves the highest Slice accuracy (88.65%), followed by *gemini-3-flash-preview* (80.54%) and *gemini-3-pro-preview* (79.46%). Among the ***Locally Deployable Models***, *mistral-medium-3* achieves the strongest overall Slice performance, although a larger performance gap relative to the frontier models is observed than in the IP category.

Failure analysis reveals that Slice errors are dominated by state-preserving execution failures rather than misunderstandings of the requested telecom operation. Typical failure modes include selecting `update_slice` when `update_apn` is required, modifying the wrong APN entry, overwriting existing slice definitions, resetting preserved `sst`/`sd` values, using an incorrect slice index, or unintentionally modifying security credentials while updating slice parameters. Models also frequently fail to retrieve the current subscriber state before applying modifications, resulting in loss of previously configured information. Among the ***Cloud-Based Frontier Models***, *qwen3-235b-a22b-instruct* performs strongly under training-assisted prompting but exhibits failures involving APN/slice routing, nested slice modifications, preservation of unrelated session parameters, and incorrect handling of `sst`/`sd` values in the no-training setting. *llama-3.3-70b-instruct* identifies the intended operation in many cases but exhibits a substantially higher frequency of APN routing errors, incorrect nested updates, false not-found outcomes, tool execution loops, and unintended modification of fields that should remain unchanged. *deepseek-v3.2* achieves identical aggregate Slice accuracy in both training and no-training settings (82.70%), indicating that training provides limited benefit for this category. Analysis of repeated failure cases shows recurring difficulties with specific APN and slice modification scenarios across multiple iterations, suggesting stable hard cases rather than random execution errors. Among the ***Locally Deployable Models***, *mistral-medium-3* remains comparatively robust, while weaker models such as *mistral-small3.1*, *mistral-nemo:12b*, and *gemma3-tools* frequently fail to preserve configuration consistency across multi-step slice modifications.

The strongest models, particularly *claude-sonnet-4-6*, *gemini-3-flash-preview*, *gemini-3-pro-preview*, *qwen3-235b-a22b-instruct*, and *deepseek-v3.2*, maintain high performance despite the increased procedural complexity of Slice Tasks. However, the larger performance gap between frontier and locally deployable models compared with the IP category indicates that successful slice management requires substantially stronger workflow execution and state management capabilities.

These findings suggest that slice tasks evaluate considerably more than telecom intent interpretation. The primary challenge lies in reliable state preserving execution, where models must modify specific subscriber attributes while maintaining the consistency of all unrelated configuration elements. As a result, performance is constrained less by telecom domain knowledge and more by the ability to execute multi-step configuration workflows without introducing unintended changes to the subscriber state.

3) Change_info Tasks: The *Change_info Tasks* category represents the most operationally demanding benchmark category in the no training setting. In this experiment, models were evaluated using the full dataset as input queries, but without training assisted examples or reference tool-sequence demonstrations. Therefore, the models had to infer the required subscriber-management procedure directly from the task instruction and the available API/tool interface. This setting provides a stricter assessment of zero-shot procedural reasoning, state preservation, and multi-step execution reliability. The tasks in this category include security credential rotation using `k` and `opc`, subscriber status updates, AMBR modifications, IPv4 and IPv6 reconfiguration, PDN-type changes, and subscriber recreation workflows. Unlike IP Tasks, which usually involve a narrow and explicit field update, and Slice Tasks, which primarily target APN and slice-related parameters, *Change_info Tasks* often require modifying one part of the subscriber profile while preserving the remaining configuration exactly. This includes retaining IP addressing, slice/session information, QoS and AMBR values, subscriber status, and

security-related fields that are not part of the requested change.

Most *Chang_info Tasks* require multi-step execution. A correct response typically involves retrieving the current subscriber state, identifying the exact target fields, applying the requested modification, and verifying that all unrelated fields remain unchanged. Security credential rotation and subscriber recreation are especially challenging because they require reconstructing or updating sensitive subscriber attributes while maintaining consistency across the complete subscriber profile. As a result, this category stresses not only telecom domain understanding but also procedural correctness and exact state-preserving execution.

Figure 2(e) presents the no training results for *Change_info Tasks*. Under the full dataset input query setting, *gemini-3.1-pro-preview-customtools* achieves the strongest observed performance, with an estimated accuracy of 87.20%. It is followed by *claude-sonnet-4-6* at 71.20%, *deepseek-v3.2* and *gemini-3-flash-preview* at 65.60%, *gemini-2.5-pro* at 64.80%, and *qwen3-235b-a22b-instruct* at 64.00%. Among the locally deployable models, *mistral-medium-3* obtains the strongest result, reaching 59.20%. These results show that even strong models experience a substantial performance drop in *Change_info Tasks* when no training examples are provided. Failure analysis shows that the dominant error source is not the inability to identify the intended operation, but the inability to preserve the complete subscriber state after applying the operation. Across the no training failure logs, *state_mismatch* errors account for the majority of observed failures, followed by *tool_round_limit* failures. This indicates that many models either applied the requested update but changed unrelated fields or entered incomplete/repeated tool-execution loops without reaching a valid final subscriber state.

The most failure-prone task family is *security_rotation_and_recreate* tasks, where subscriber authentication credentials, mainly `k` and `opc`, are updated through a state preserving reconstruction workflow. Thus, the model must retrieve the existing subscriber profile, apply only the requested credential changes, and recreate or update the subscriber while preserving all unrelated fields such as APN, slice, IP, AMBR, status, and PDN type. This family accounts for the majority of failures overall and dominates the failures of several high-performing models. For example, all observed failures for *claude-sonnet-4-6* occur in this family, while the failures of *gemini-2.5-pro*, *gemini-3-flash-preview*, *deepseek-v3.2*, *gpt-5.5*, and *qwen3-235b-a22b-instruct* are also heavily concentrated in security credential rotation and subscriber reconstruction tasks. These failures typically involve correct recognition of the requested `k`/`opc` update, but incomplete preservation of surrounding fields such as slice/session configuration, IP configuration, AMBR values, or subscriber status. Other task families, including AMBR updates, subscriber status changes, and IPv4/IPv6 reconfiguration, produce fewer failures for the strongest frontier models but remain challenging for several locally deployable models. In these cases, failures often arise from partial updates, incorrect preservation of PDN-type information, missing IPv6 fields, altered AMBR values, or incomplete API sequences. Models such as *mistral-nemo:12b*, *mistral-small3.1*, *gemma3-tools*, and *granite4:small-h* show broader failure patterns across multiple task families, suggesting weaker robustness in multi-field subscriber reconfiguration. These results indicate that *Change_info Tasks* evaluate a stricter capability than basic API selection. Most models can often infer the correct subscriber-management intention, but successful completion requires exact preservation of the pre-existing subscriber state across multi-step workflows. The gap between frontier and locally deployable models is therefore mainly explained by differences in state tracking, procedural consistency, and the ability to execute complex subscriber modifications without unintended side effects. The results show that zero-shot model performance is primarily constrained by state preserving execution rather than telecom terminology alone.

For IP and Slice tasks, training prompting improves performance for many models by exposing the expected operational workflow, including subscriber retrieval, endpoint selection, and state preservation. The largest gains are observed in the *Slice Tasks*, where successful execution requires preserving complex subscriber configurations across multi-step workflows. These results indicate that in-context demonstrations can help models follow the required procedural structure, particularly when tasks involve dependent API calls and nested subscriber state. However, training prompting does not eliminate failures in preservation sensitive operations, and some models still exhibit incorrect updates, incomplete state reconstruction, or unintended modification of unrelated fields. These findings suggest that prompt-based guidance alone is insufficient for reliable autonomous telecom operation and should be complemented by schema-constrained outputs, explicit state validation mechanisms, and preservation-aware execution guardrails.

VI. Discussion and Challenges

The evaluation results highlight several critical challenges in deploying LLMs for autonomous telecom configuration, moving beyond basic intent comprehension to reveal structural limitations in current generative architectures.

A prominent operational challenge is the phenomenon of **in-context negative transfer**. While training-assisted prompting generally improves performance on complex, multi-step workflows, it was observed to actively degrade accuracy on simpler, isolated tasks. Autoregressive models frequently overfit to the complex JSON schemas provided in few-shot examples, treating them as rigid templates rather than flexible guides. This schema rigidity leads to parameter hallucination and structural cross-contamination. For example, models erroneously copied explicit hexadecimal security keys from training prompts into live configurations or forced nested IPv6 session formatting onto standard IPv4 requests. Consequently, providing comprehensive training context can inadvertently impair the execution of simpler operations.

Furthermore, the generation of telecom-specific payloads is severely constrained by **tokenization** traps. Deep subscriber

configurations heavily rely on long alphanumeric sequences, such as IPv6 addresses and 32-character hexadecimal security credentials (e.g., K, OP, OPC keys). Standard sub-word tokenizers fragment these strings unpredictably. During the reconstruction of the subscriber state, this fragmentation mathematically increases the probability of autoregressive syntax degradation, causing models to truncate, hallucinate, or alter critical strings. This token-length vulnerability directly drives the high rate of "state mismatch" errors observed in the Change_info tasks.

Finally, while the MCP successfully standardizes tool discovery and API invocation, current orchestrations lack intrinsic semantic validation. The architecture currently trusts the LLM to generate the entire JSON payload accurately in a single pass before execution. To mitigate the prevalent state preservation failures observed in this benchmark, future telecom architectures must evolve. Implementing **schema-constrained generation** [15], [16], which forces the model's output sequence to strictly adhere to predefined API data models, can eliminate syntax-level hallucinations. Furthermore, integrating dynamic verification loops that algorithmically compute the delta between the proposed state changes and the existing subscriber profile before API commitment will be essential to ensure collateral configurations remain uncorrupted.

VII. Conclusion

This paper presented an MCP-mediated architecture to evaluate the operational reliability of Large Language Models (LLMs) in executing Intent-Based Networking workflows within an Open5GS 5G core environment. Through a rigorous multi-step benchmark, we demonstrated that while current LLMs can effectively translate natural language intents into basic network actions, autonomous telecom management is fundamentally limited by a model's capacity for strict, preservation-aware execution.

Our evaluation revealed a pronounced performance gap between model tiers. Cloud-based frontier models generally maintained higher accuracy than locally deployable models across complex workflows, but they still exhibited failures in preservation-sensitive operations. In contrast, locally deployable models suffered significant degradation when forced to navigate deeply nested JSON configurations, such as slice descriptors and APN parameters, routinely failing to preserve unrelated subscriber data during targeted updates. Ultimately, the primary bottleneck for telecom LLM agents is not intent understanding, but rather the flawless preservation of network state across multi-step API workflows.

To bridge this operational gap and ensure the safety of autonomous telecom control planes, future work will focus on the integration of Network Digital Twins (NDT) to simulate and validate generated configurations in a sandboxed environment prior to live deployment. Additionally, developing dynamic, closed-loop verification mechanisms will be critical to guaranteeing structural and semantic consistency, ensuring that the reliability of 5G automation is not solely dependent on the generative limits of the underlying language model.

In addition to runtime validation architectures, our future work will explore domain-specific continuous pre-training and supervised fine-tuning (SFT) of the underlying language models using authoritative 3GPP technical specifications, alongside the detailed engineering documentation of the 5G Core and Radio Access Network (RAN) components. By embedding deep, cellular-specific structural knowledge directly into the model's parametric memory rather than relying solely on fragile in-context learning templates, we aim to significantly reduce tokenization traps and schema rigidity. This dual strategy of foundational network documentation training paired with runtime deterministic verification loops represents the next evolutionary step toward truly robust, safe, and autonomous telecom configuration agents.

References

[1] A. Clemm *et al.*, "Intent-Based Networking - Concepts and Definitions," RFC 9315, Oct. 2022.

[2] D. M. Manias *et al.*, "Towards Intent-Based Network Management: Large Language Models for Intent Extraction in 5G Core Networks," in *2024 20th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2024.

[3] Z. Wang *et al.*, "MCP-bench: Benchmarking tool-using LLM agents with complex real-world tasks via MCP servers," in *Workshop on Scaling Environments for Agents*, 2025. [Online]. Available: https://openreview.net/forum?id=2InRbaYve7

[4] C. Wang *et al.*, "NetConfEval: Can LLMs Facilitate Network Configuration?" *Proceedings of the ACM on Networking, Volume 2, Issue CoNEXT2*, Jun. 2024.

[5] ——, "Q*: Improving Multi-step Reasoning for LLMs with Deliberative Planning," 2024. [Online]. Available: https://arxiv.org/abs/2406.14283

[6] T. Wu *et al.*, "PromptChainer: Chaining Large Language Model Prompts through Visual Programming," in *ACM Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '22, New York, NY, USA, 2022.

[7] W. Song *et al.*, "Help or Hurdle? Rethinking Model Context Protocol-Augmented Large Language Models," 2025. [Online]. Available: https://arxiv.org/abs/2508.12566

[8] J. Menezes *et al.*, "Net2d-LLM: Translating Structured Network Intents into CLI using LLMs with Execution in a Network Digital Twin," in *Anais da XXII Escola Regional de Redes de Computadores (ERRC)*. Porto Alegre, RS, Brasil: SBC, 2025, pp. 54–60. [Online]. Available: https://sol.sbc.org.br/index.php/errc/article/view/39181

[9] P. Gajjar *et al.*, "Tele-LLM-Hub: Building Context-Aware Multi-Agent LLM Systems for Telecom Networks," 2025. [Online]. Available: https://arxiv.org/abs/2511.09087

[10] Z. Liu *et al.*, "Model Context Protocol-based Internet of Experts For Wireless Environment-aware LLM Agents," 2025. [Online]. Available: https://arxiv.org/abs/2505.01834

[11] E. Baena *et al.*, "Demo: Human-in-the-loop agentic reconfiguration of edge 5g networks via dual-mcp and llm reasoning," in *MobiHoc '25*. New York, NY, USA: Association for Computing Machinery, 2025, pp. 391–392. [Online]. Available: https://doi.org/10.1145/3704413.3765314

[12] GSMA, "Gsma launches open-telco llm benchmarks to advance ai in telecoms," February 2025. [Online]. Available: https://www.gsma.com/newsroom/press-release/gsma-open-telco-llm-benchmarks-launches-to-advance-ai-in-telecoms/

[13] [NetoAi], "Netbench dataset," 2025. [Online]. Available: https://netoai.ai/

[14] JSON-RPC Working Group, "JSON-RPC 2.0 Specification," https://www.jsonrpc.org/specification, accessed: 01-06-2026.

[15] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Guiding llms the right way: fast, non-invasive constrained generation," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.

[16] S. Geng, H. Cooper, M. Moskal, S. Jenkins, J. Berman, N. Ranchin, R. West, E. Horvitz, and H. Nori, "Jsonschemabench: A rigorous benchmark of structured outputs for language models," 2025. [Online]. Available: https://arxiv.org/abs/2501.10868