Conversational Orchestration for Organic 6G

Masoud Shokrnezhad[1] and Tarik Taleb[2]
[1] *ICTFICIAL Oy, Espoo, Finland*; masoud.shokrnezhad@ictficial.com
[2] *Ruhr University Bochum, Bochum, Germany*; tarik.taleb@rub.de

***Abstract*—The Organic 6G vision of a network of networks spanning an edge–cloud continuum complemented by non-terrestrial resources requires, to realize its promise, service provisioning that is simple to operate, scalable across independently administered domains, and agile under domain churn (i.e., domains dynamically joining and leaving). Despite advances in cross-domain orchestration, many proposals rely on heavy integration fabrics, multi-layer coordinators, and deep telemetry pipelines that hinder deployability and amplify coordination overhead. We propose a lightweight, decentralized conversational orchestration framework based on Large Language Model (LLM)-driven domain agents. Each domain remains autonomous: an agent observes local state via tools, reasons in a closed loop, and exchanges summaries with neighboring agents over an Agent-to-Agent (A2A) overlay aligned with data-plane coupling. Fast feasible placement is enabled by periodic, routing-like dissemination of reachability advertisements (latency, bottleneck bandwidth, and compute capacity), while safe re-optimization, scaling, and migration are handled through event-driven requests and negotiation. To meet real-time constraints, we deploy a compact reasoning model trained with verifier-based self-verification and periodically refined online via shadow updates. Simulations show manageable, near-linear control-plane overhead as domains scale and during domain joins, and robust decision quality, including recovery after objective changes. We close by outlining future research directions for principled, secure, and uncertainty-aware agentic orchestration in Organic 6G.**

***Index Terms*—Organic 6G, Service Provisioning, Multi-Domain Orchestration, Agentic AI, Large Language Models, Non-Terrestrial Networks.**

I. Introduction

Organic 6G envisions the future mobile system as a software-centric *network of networks*, where heterogeneous radio, compute, and transport resources, spanning edge-to-cloud and including Non-Terrestrial Networks (NTN) assets, continuously appear, disappear, and reshape under independent administrative control [1]. This evolution aligns with the broader push toward *open networks* built on interoperable standards for cross-domain service delivery [2]. From a multi-domain perspective, this direction is not optional: end-to-end services inevitably traverse domains with distinct policies, trust boundaries, and operational constraints. Hence, 6G must be designed to function under decentralization and churn, supporting seamless domain onboarding/offboarding rather than assuming a fixed, single-operator topology. This structural dynamism (an infrastructure that must remain coherent and serviceable as resource domains continuously join, leave, and evolve) is the central challenge motivating this paper: any orchestration approach that assumes a static, globally known topology will fail in this setting.

However, an Organic 6G infrastructure is only useful if services can be provisioned over it. In this paper, *service provisioning* denotes the end-to-end process of turning a service description into running, reachable service instances by (i) selecting and placing compute across the edge–cloud continuum and (ii) establishing the required connectivity so users can be bound to those instances, while continuously adapting via scaling and migration as conditions change. Doing so across many independently administered domains requires methods that are **scalable** (bounded coordination cost as domains grow), **simple** (deployable and failure-resilient without heavy operational burden), and **agile** (plug-and-play domain join/leave, i.e., resource domains dynamically onboarding or going offline at runtime) [3]. While recent work has made progress toward cross-domain orchestration in 5G/6G settings, existing approaches often presume substantial architectural machinery (e.g., multi-layer coordinators, integration fabrics, deep telemetry/Artificial Intelligence (AI) pipelines) and pre-established federation agreements, which complicate plug-and-play dynamics and can drive coordination overhead upward with the number of domains [4]–[8].

To fill this gap, this paper proposes a lightweight, decentralized *conversational orchestration* approach based on Large Language Model (LLM)-driven domain agents that coordinate service provisioning decisions over an inter-domain overlay graph. The key insight is that LLM-based agents shift orchestration complexity from engineered architectural machinery (integration fabrics, coordinator hierarchies, raw telemetry aggregation) to goal-driven reasoning. Each domain remains autonomous – its agent optimizes locally using domain tools and policies – yet end-to-end coordination emerges by exchanging only goal-driven, summarized reachability information with neighboring agents via Agent-to-Agent (A2A) messaging. Concretely, the proposed design combines (i) periodic, routing-like dissemination of compact resource reachability advertisements and (ii) on-demand request/negotiation for safe re-optimization and migration, thereby keeping coordination bounded while supporting domain churn. This keeps the design **scalable** (localized exchanges, no centralized entity), **simple** (no heavy orchestration machinery), and **agile** (plug-and-play domain join/leave).

The remainder of this paper is organized as follows: Section II provides background and requirements; Section III presents the proposed agent-based architecture and decision process; Section IV evaluates the approach via simulations; and Section V discusses future directions and concludes this paper.

II. Background

A. Infrastructure

From a resource view, Organic 6G treats the infrastructure as a distributed continuum of connectivity and computation rather than a fixed, centralized stack. At the access edge, the Radio Access Network (RAN) (including disaggregated/vRAN

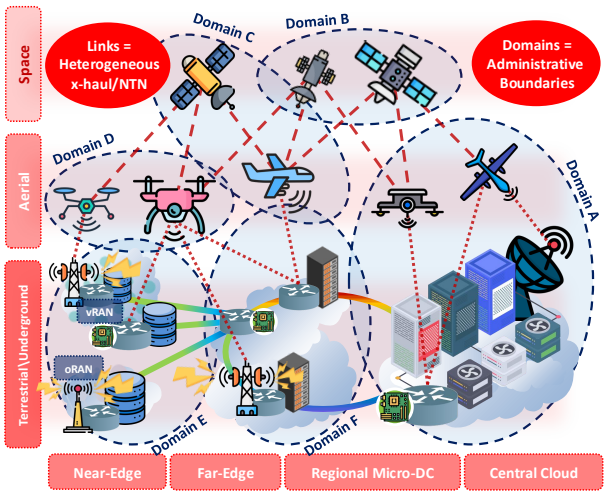

Fig. 1. Organic 6G infrastructure as a distributed continuum of connectivity and computation spanning terrestrial, aerial, and space assets from near-edge to central cloud, organized as a modular network of networks with isolated administrative domains interconnected via heterogeneous x-haul/NTN.

and Open RAN components) anchors the system by attaching User Equipment (UEs) and exposing radio resources that can be shaped per service. Beyond the RAN, computation spans in-network elements, near-edge nodes (co-located with or adjacent to the RAN), far-edge nodes (district/metro-level aggregation points), regional micro-data centers, and central clouds, forming a continuum that trades latency for capacity. The interconnect is inherently heterogeneous: front-/mid-/backhaul spans fiber, copper, terrestrial wireless, and NTN assets such as satellites, Unmanned Aerial Vehicles (UAVs), and High Altitude Platforms (HAPs)/Low Altitude Platforms (LAPs), each with distinct bandwidth, latency, and intermittency profiles [9]; this heterogeneity extends reach and resilience where terrestrial links are sparse or disrupted. Hardware heterogeneity (Field-Programmable Gate Arrays (FPGAs), Smart Network Interface Cards (SmartNICs), GPUs) further supports efficient data-plane execution at the edge.

At a system level, Organic 6G interprets this continuum as a modular *network of networks* built from isolated administrative domains, where each domain can own and operate a well-scoped subset of the overall resource pool. In practice, one entity may provide the access segment (e.g., a local RAN deployment), another may operate a portion of the compute continuum (e.g., near- or far-edge nodes, or regional micro-data centers), while additional entities may contribute transport/x-haul capacity or non-terrestrial resources. Each domain encapsulates its resources behind domain-local policies and management functions, enabling independent lifecycle management without assuming a static, globally engineered topology. The result is an infrastructure that remains structurally coherent even when its constituent resource domains appear, disappear, or change shape over time. The co-existence of these heterogeneous resources, each operated by an independent administrative domain with its own policies and capabilities, is what makes end-to-end service provisioning non-trivial: a single service may require compute at multiple tiers, connectivity across heterogeneous link types, and continuous adaptation as users move or conditions change, all without any single entity having global visibility or control. Fig. 1 provides an overview of the above. Unlike ephemeral networks (temporary by design) or end-to-end network slicing, which virtualizes resources over a globally orchestrated substrate, Organic 6G describes a persistent, multi-domain network of networks where the physical infrastructure itself evolves dynamically under independent administrative control, with no single entity holding a global view.

B. Service Provisioning

Building on the infrastructure continuum described above, service provisioning refers to the end-to-end process of turning a service description into running, reachable service instances over the available *computing* and *networking* resources. For example, a latency-sensitive service may be placed on a near-edge node in the user's current domain, with connectivity established through the local RAN and x-haul; as the user moves across domain boundaries, the instance is migrated to a near-edge node in the new domain to preserve Quality of Service (QoS). Concretely, it entails placing service instances across the edge–cloud spectrum and establishing RAN/x-haul connectivity to bind users to those instances [10]. In Organic 6G, provisioning is driven by stringent QoS objectives under user mobility and highly dynamic resource conditions: latency is often the dominant metric for interactive and control workloads, while availability captures whether sufficient compute and connectivity resources remain continuously accessible to sustain a user's service experience.

Beyond initial placement, service provisioning includes lifecycle adaptation operations that keep the service performant as conditions change. Scaling adjusts the number of active instances and their allocated resources to follow demand and to mitigate hotspots, whereas migration relocates an instance (or its execution context) to a different node to preserve QoS as users move or as local resources become scarce or unreliable. Together, placement, binding, scaling, and migration form the core of service provisioning in Organic 6G: because the environment is highly dynamic (user loads shift, users move, and infrastructure conditions fluctuate), a static one-time placement is insufficient, and continuous adaptation is required to maintain a valid mapping between users, service instances, and infrastructure resources so that QoS objectives remain satisfied. Section III describes how the proposed agent-based approach addresses these challenges through continuous reachability tracking, event-driven re-optimization, and online model refinement.

C. Problem Statement & Challenges

The core problem addressed in this paper is the optimization of service provisioning over an Organic 6G infrastructure. Given the distributed pool of *computing* resources (from near-edge to central and non-terrestrial nodes) and *networking* resources (the RAN and heterogeneous transport/x-haul), an orchestrator must (i) collect and maintain sufficiently accurate system information across the network of networks, (ii) decide provisioning actions, including placement, user–instance

binding, scaling, and migration, and (iii) enforce these actions on the underlying resources. This closed-loop must operate under stringent QoS requirements (e.g., latency) while the system exhibits high dynamism. Importantly, any practical solution must satisfy three system-level requirements beyond single-domain optimality [3]: **scalability** (coordination cost must remain bounded as domains grow, avoiding centralized bottlenecks and single points of failure); **simplicity** (the architecture must be deployable and failure-resilient without heavy operational burden); and **agility** (domains must support plug-and-play onboarding/offboarding so the system remains functional as resource domains appear, disappear, or change shape).

Recent work has investigated cross-domain orchestration for service provisioning in multi-domain 5G/6G environments. Giannopoulos *et al.* [4] proposed ACROSS, a two-level architecture where domain orchestrators are supervised by a cross-domain coordinator through a standardized integration fabric, relying on deep telemetry and AI-assisted zero-touch provisioning automation. Molner *et al.* [5] introduced AIORA, advocating virtual continuums spanning multiple segments and a cross-segment coordination substrate built atop open interfaces and nested AI-driven closed loops. Dalgitsis *et al.* [6] addressed inter-operator continuity via cloud-native slice federation, translating slice templates into federated templates exchanged over operator federation interfaces to instantiate slices in visited domains. Benlloch-Caballero *et al.* [7] studied end-to-end slicing for security automation, coupling distributed sensing and topology tracking with an orchestration loop and cross-domain enforcement agents. Finally, Santos *et al.* [8] proposed a hierarchical orchestration scheme for end-to-end network slicing, introducing a higher-level hyperstrator to coordinate distributed per-segment orchestrators across domains, while relying on pre-defined inter-orchestrator interfaces and a central coordination point for lifecycle management. While the achievements are valuable, they still fall short of the **simplicity**, **agility**, and **scalability** requirements emphasized above: they typically presume substantial architectural machinery (integration fabrics, multiple orchestration layers, deep telemetry/AI pipelines, or custom enforcement components), pre-established interoperability and federation agreements that complicate plug-and-play domain join/leave, and coordination patterns whose overhead exponentialy grows with the number of domains and slices (e.g., centralized supervision, heavy telemetry aggregation, or increasing inter-orchestrator messaging).

These gaps motivate our lightweight, decentralized agent-based coordination approach in Section III, designed to keep coordination bounded, operational complexity low, and domain dynamics seamless at Organic 6G scale.

III. Proposed Approach

A. Architecture

We build our solution around *LLM-based agents*: autonomous control entities that observe their environment, reason, and act in closed loop. LLMs are chosen over analytical or pure data-driven alternatives because their goal-driven reasoning generalizes across heterogeneous domain configurations

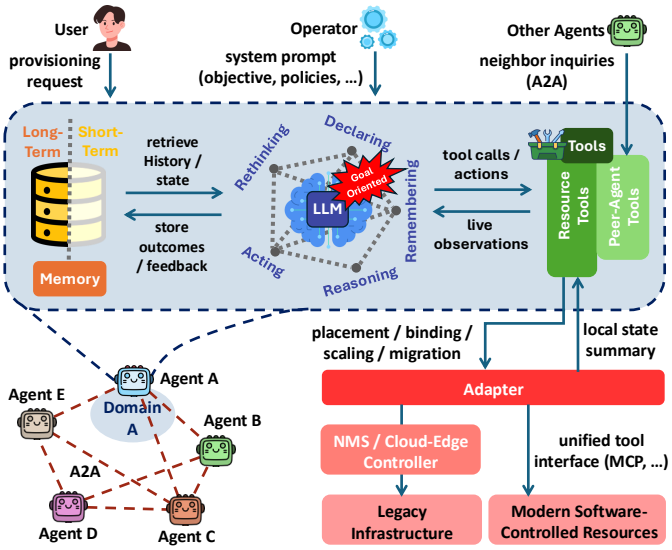

Fig. 2. Illustration of the proposed LLM-based domain agent and its inter-domain overlay control plane for Organic 6G service provisioning. The LLM is the reasoning core; memory provides working context and long-term history; the tool layer interfaces with domain resources and peer agents via A2A; and the adapter bridges the agent to both modern software-defined infrastructure (e.g., via MCP) and legacy NMS/API stacks.

and adapts to objective changes without re-engineering (the accompanying Reinforcement Learning (RL)-based specialization then ensures task-specific optimization competence). Here, the LLM serves as the agent's reasoning core with five internal capabilities: *goal setting* (what to optimize), *decision synthesis* (generating candidate actions), *memory* (storing and retrieving state and feedback), *action execution* (enforcing decisions via tools), and *self-correction* (revising the reasoning path based on outcomes). These capabilities are realized with three modules: the LLM itself, a memory store (short-term working context plus persistent long-term history), and a tool layer that interfaces with resources and peer agents. An overview of the proposed agent architecture is shown in Fig 2. This design is deliberately goal-oriented: rather than relying on predefined orchestration pipelines, the agent *selects which observations to request*, *which tools to invoke*, and *which control actions to apply* based on the currently declared goals and constraints. The claimed system properties (simplicity, agility, scalability) are architectural properties of the orchestration system, not of the model: the model's computational cost is local to each domain and does not add cross-domain coordination overhead, and Section III-C describes the use of a compact Small Language Model (SLM) to keep per-domain inference latency low.

From an intra-domain point of view, we place one agent at the top of each administrative resource domain, responsible for domain-local resource allocation in support of service provisioning. The domain agent continuously collects and summarizes local state (e.g., available compute/network capacity, current allocations, local policies, and observed QoS), receives provisioning requests, and enforces actions such as placement, user–instance binding, scaling, and migration. It can also revise existing allocations to remain optimized as demand and infrastructure conditions change. The agent connects

to the domain through an adapter layer: for modern, software-controlled resources it can interact directly via a unified tool interface (e.g., Model Context Protocol (MCP) connectors toward devices/controllers), while for legacy infrastructures it operates through existing management stacks such as NMSs or cloud/edge controllers that expose monitoring and actuation APIs.

From an inter-domain point of view, the agents form an overlay control-plane *graph*, where each node corresponds to a domain agent and each edge represents a communication relationship [11]. We align this overlay with data-plane coupling: if two domains are connected (or interdependent) in the data plane, their agents are connected in the control plane and can exchange information about relevant cross-domain state. Importantly, by placing the coordination logic in the agents, domains are not required to implement new cross-domain coordination standards; instead, agents communicate using A2A protocols on the control plane. As a result, the architecture directly improves: **scalability** by limiting coordination to goal-driven, neighbor-to-neighbor exchanges rather than raw global state collection; **simplicity** by shifting orchestration logic from fragile, manually engineered workflows to an agentic closed loop over tools and feedback; and **agility** by supporting plug-and-play join/leave at the domain level (a domain onboards by deploying its agent+adapters and establishing edges to adjacent domains).

B. Decision Making

Our agents coordinate cross-domain provisioning using two complementary message-passing modes on the overlay control plane. The first is *change-triggered dissemination*, which maintains a routing-like view of feasible resources; the second is *on-demand request/negotiation*, which supports safe re-optimization and migration. In both cases, each domain agent remains authoritative only within its own administrative boundary: it advertises summarized reachability information, but enforces allocations (admission, placement, binding, scaling, migration) solely through its domain-local tool layer and policies. An overview is illustrated in Fig. 3.

Change-triggered dissemination (table-driven). A domain agent summarizes the feasibility of reaching its *local* computing resources from its data-plane *inter-domain ingress/egress points*. Concretely, for each reachable compute resource it derives a compact *resource advertisement* combining end-to-end latency, path bottleneck bandwidth, and available compute capacity. The agent then shares this advertisement with neighboring agents over A2A. Upon receipt, a neighbor performs a routing-style update: for each advertised destination it adds its own intra-domain access latency to the destination, tightens the available bandwidth to reflect the new bottleneck, and records the next-hop neighbor toward that destination. This process repeats whenever a domain agent observes a material change in its intra-domain resources or connectivity (or receives an updated advertisement from a neighbor that changes its table). By repeating these local updates, agents build and refresh a distributed *resource reachability table* that maps feasible remote computing resources to their predicted end-to-end latency and available capacities, without requiring raw global telemetry collection.

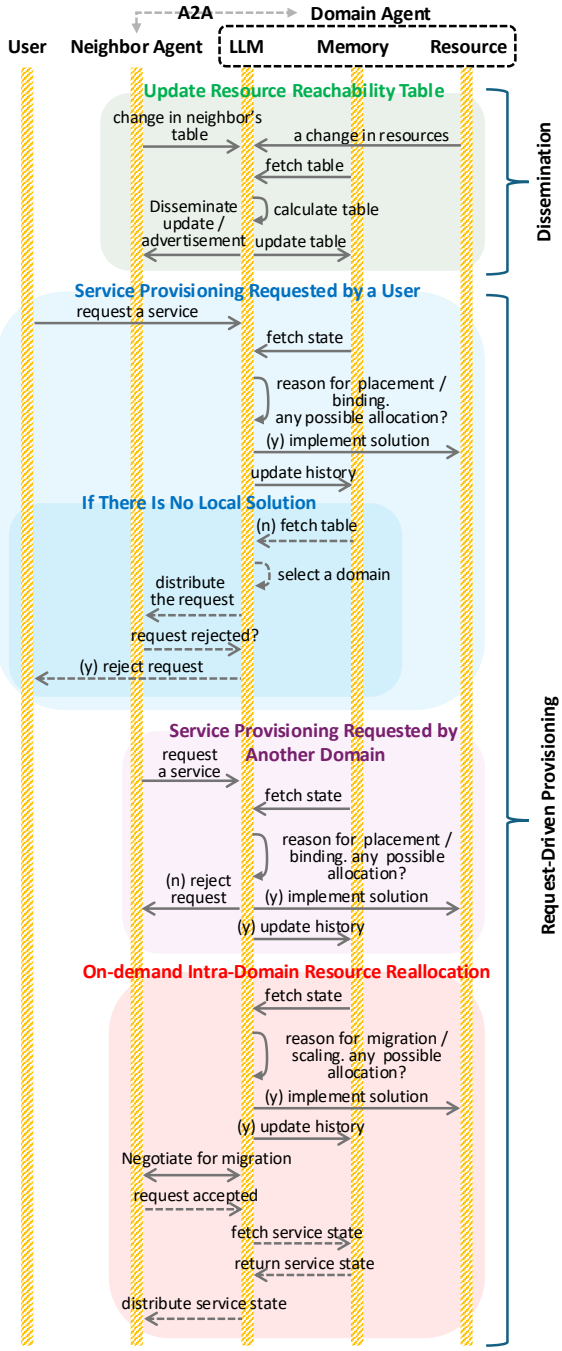

Fig. 3. High-level procedures for change-triggered reachability dissemination and service provisioning on the inter-domain agent overlay. For tractability, the figure omits detailed per-hop mechanisms (e.g., multi-hop forwarding, soft reservations/commit).

Request-driven provisioning (per-request). When a provisioning request arrives at a domain, the receiving agent first checks whether it can satisfy the request locally (subject to domain policies and current allocations). Otherwise, it consults its reachability table to select a feasible remote destination whose advertised compute and bandwidth capacities satisfy the request and whose predicted end-to-end latency meets the request's QoS constraints; among candidates, it prefers the one minimizing latency. The request is then forwarded hop-by-hop

on the control plane until it reaches the destination domain agent. Each intermediate agent installs a *soft reservation* on its segment and the outbound inter-domain link; once the destination domain allocates compute, a confirmation travels back and each domain *commits* its reservation, binding the control-plane chain to an end-to-end data-plane path while preserving domain autonomy. Because feasibility information is already available in the table and reservations are established incrementally per hop, this mode supports low decision latency for per-request allocations while keeping coordination bounded to neighbor exchanges.

On-demand negotiation (event-driven). Disseminated tables are intentionally coarse: they enable fast, feasible placement, but they do not by themselves guarantee that subsequent *changes* (e.g., intra-domain path rebalancing, resource reallocation, or migration) preserve each service's QoS. For these cases, agents switch to an on-demand mode in which an agent initiating a change (e.g., flow rerouting, migration) negotiates with the agent(s) responsible for the affected service instance(s) via A2A messaging, exchanging the additional state needed (updated path characteristics, migration context) to confirm that end-to-end QoS is preserved before enforcement. While such negotiation introduces additional control latency, it is used for optimization and lifecycle adaptation rather than the critical path of initial admission, thereby enabling improved optimality without imposing a centralized orchestrator.

C. Training & Inference

To operate as domain-top controllers, the proposed agents must generate provisioning decisions that are *fast* yet sufficiently *intelligent* for multi-constraint optimization. Accordingly, each domain agent is instantiated with an SLM *pretrained on strong reasoning tasks* as its reasoning core (DeepSeek-R1-Distill-Qwen-7B in the evaluation): this choice preserves low inference latency while providing the reasoning competence needed for reliable, goal-driven control (training procedure and evaluation are detailed in what follows and in Section IV). However, service provisioning in Organic 6G requires more than generic reasoning: the model must internalize domain policies, optimize under multiple criteria (e.g., latency and resource efficiency), and produce *actionable* allocations that satisfy QoS constraints. To this end, the SLM is specialized using RL with *verifiable* feedback signals, following the self-verification paradigm popularized by recent reasoning models [12].

Offline self-verification training. Fig. 4 summarizes the offline self-verification procedure. A set of training *contexts* is constructed to reflect the foundations in Section II and the decision process in this section: each context encodes (i) a domain view of compute and network resources, (ii) local policies and constraints, (iii) representative provisioning requests, and (iv) target optimization criteria. For each context, the SLM generates a reasoning trace and a candidate provisioning action (placement, user–instance binding, scaling, or migration). A stronger verifier LLM then assesses the output and assigns a *multi-objective* reward vector capturing three aspects: (1) *reasoning validity* (is the allocation supported by a coherent

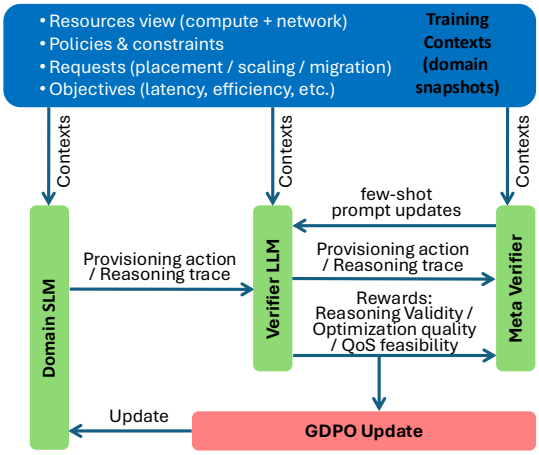

Fig. 4. Offline self-verification training of domain SLMs. Training contexts (domain snapshots) elicit reasoning traces and provisioning actions; a strong verifier assigns multi-objective rewards (reasoning validity, optimization quality, and QoS feasibility) that drive GDPO updates. Verifier reliability is improved via meta-verifier–guided few-shot prompt refinement.

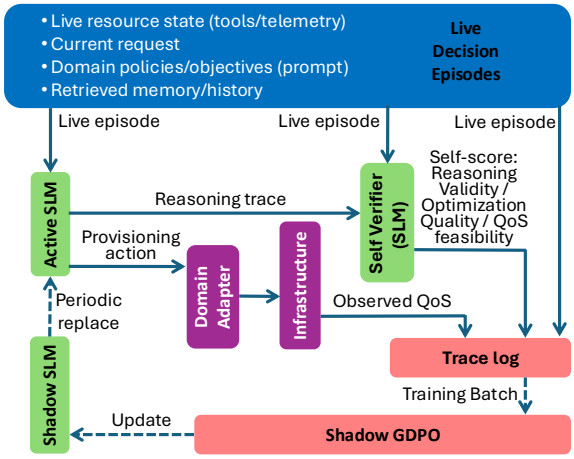

Fig. 5. Online inference and periodic refinement with shadow updates. The deployed SLM agent retrieves memory, queries live state via tools, enforces provisioning actions through the domain adapter, and logs traces together with observed QoS. Traces are batched to update a shadow copy offline (e.g., GDPO), which periodically replaces the deployed model.

and complete reasoning path, avoiding unjustified shortcuts [13]), (2) *optimization quality* (how well the proposed decision matches the declared objectives such as latency minimization under capacity constraints), and (3) *QoS feasibility* (whether the requests can be served within their QoS bounds given the advertised resources). These heterogeneous rewards are used to update the SLM with *Group reward-Decoupled Normalization Policy Optimization (GDPO)*, which stabilizes training by normalizing each reward component before aggregation, thereby preserving learning signal resolution in multi-reward settings [14]. In parallel, the verifier is improved through a lightweight bootstrapping loop: a meta-verifier reviews verifier judgments, and the resulting feedback is injected into the verifier prompt (few-shot style) to progressively reduce systematic errors, without requiring additional training infrastructure.

Online inference and periodic refinement. Fig. 5 illustrates the online closed loop and the periodic shadow-update refinement strategy. After offline training, the SLM-

based agent is deployed on top of each domain with access to its memory and tool layer, and with a system prompt encoding the domain's policies and optimization criteria. At inference time, the agent follows the closed loop described earlier: it retrieves relevant history, collects live state via tools, proposes a decision, and enforces it through the domain adapter. Reliability is improved by reusing the same self-verification loop as in offline training, which also serves as a lightweight hallucination guard: outputs violating QoS feasibility or lacking coherent reasoning are penalized before enforcement. For each decision, the SLM produces a reasoning trace and an allocation, then evaluates its own output using the same verifiable, multi-objective reward criteria learned during training. To adapt to evolving traffic and infrastructure conditions without blocking real-time control, inference traces (requests, state snapshots, actions, rewards, and observed QoS) are accumulated into batches, and a *separate* copy of the SLM is updated periodically offline (e.g., with GDPO). From time to time, the updated copy replaces the working SLM [15]. This *shadow update* strategy decouples live decision making from continual retraining, while enabling steady improvement under non-stationary Organic 6G dynamics.

IV. Simulations

A. Scenario A: Control-Plane Evaluation

We quantify *control-plane message volume* to assess the overhead of the dissemination in Section III, focusing on (i) reachability-table formation and (ii) re-convergence after a domain joins. The results are illustrated in Fig. 6.A. We simulate $N \in \{10, 20, 30\}$ administrative domains whose agents form an overlay aligned with data-plane coupling. Each curve is averaged over 20 independent runs. In each run, N domains (each exposing 3 local compute resources summarized by its agent) are connected by a random inter-domain graph with target average degree 4, and the agents execute table-driven dissemination. Each time slot represents one management epoch (consistent with orchestration decision intervals on the order of seconds), so convergence within tens of slots corresponds to table formation on the order of minutes, appropriate for provisioning-level control. When a local table change occurs, the agent's message-preparation latency is measured and then normalized to a 1–3 time-slot interval before the update is emitted to neighbors; delivery succeeds with probability $p = 0.9$ and reliability is enforced by ACK-based retransmissions. To match the accounting in Fig. 6.A, each changed table entry is treated as a separate update message (sent in parallel within the same time slot), and we report update messages (ADV) only. The initial burst occurs while agents rapidly populate reachability tables with previously unknown destinations; once tables stabilize, change-triggered updates cease and the message rate collapses to near zero. At time slot 60, a new domain joins and attaches to a random subset of existing domains, introducing only incremental reachability entries, which yields a smaller transient before re-convergence. Overall, the overhead remains manageable, and for fixed average degree it increases approximately linearly with the number of domains, as more destinations must be disseminated over neighbor-only exchanges.

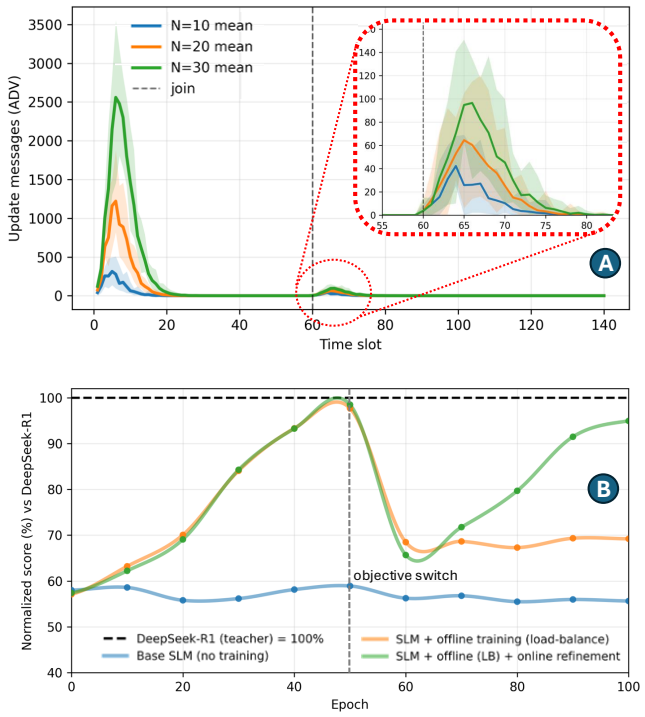

Fig. 6. Simulation results for the two scenarios. (A) Scenario A: Message-passing overhead of distributed reachability-table dissemination, including a domain join at time slot 60. Lines show the mean number of update messages per time slot; shaded regions denote min–max across random topologies for each N. (B) Scenario B: Normalized score (%) relative to DeepSeek-R1 (verifier LLM baseline fixed at 100%). The SLM improves with offline specialization on load-balance (epochs 0–50); after switching to a min-latency objective (epochs 50–100), only the variant with online refinement recovers toward the baseline.

B. Scenario B: Training and Refinement Evaluation

This scenario validates the two key claims of Section III-C: that offline RL specialization brings the SLM to near-verifier-level provisioning performance, and that online refinement enables recovery under objective changes. We evaluate the offline and online specialization loop using DeepSeek-R1 as the strong verifier LLM and DeepSeek-R1-Distill-Qwen-7B as the deployed SLM reasoning core, over a synthetic heterogeneous multi-domain infrastructure whose parameters (12 domains, inter-domain latencies 2–20 ms, bandwidths 0.5–10 Gbps, and per-domain compute pools spanning 32–128 vCPU-equivalents) span the plausible operating range of 6G deployments. We generate a dataset of 3000 service-provisioning scenarios over this infrastructure. Training proceeds in two phases. During epochs 0–50, the SLM is specialized offline toward a *load-balance* objective (balancing resource utilization). From epoch 50 onward, the objective switches to *min-latency* (minimizing overall provisioning latency). Both phases are also subject to QoS and capacity feasibility. We evaluate at checkpoints every 10 epochs and report a normalized score under the active objective, where DeepSeek-R1 is used as the 100% reference. Fig. 6.B compares three variants: (i) a base SLM without training, (ii) SLM with offline training (trained for load-balance only), and (iii) SLM with offline training followed by online refinement via periodic shadow updates. Offline training drives the SLM close to the DeepSeek-R1 reference on the load-balance objective. After

the objective switch, the offline-only variant drops and remains lower, whereas online refinement restores performance as the model adapts to the new min-latency criterion. Overall, the results indicate that the proposed offline+online specialization yields sufficient capability to maintain high accuracy relative to the verifier LLM under both stationary operation and objective changes. Practically, achieving near-verifier-level scores on the min-latency objective means the deployed SLM produces placement and binding decisions that approach the quality of a full LLM, translating directly to tighter end-to-end service latency for users while keeping inference local to each domain. Note that Scenario A and Scenario B evaluate complementary aspects: the former assesses control-plane dissemination overhead (independent of the decision-maker), while the latter evaluates SLM reasoning quality and adaptation.

V. Conclusion & Future Work

In this paper, we studied service provisioning over an Organic 6G infrastructure by viewing the system as a heterogeneous edge–cloud–NTN continuum composed of independently administered resource domains. We formulated the problem as a continuous placement-and-binding loop under stringent QoS targets, subject to scalability, simplicity, and agility requirements. To address these, we proposed a decentralized architecture based on LLM-based domain agents connected by an A2A overlay graph: agents used table-driven dissemination to build reachability views for fast feasible placement, and switched to event-driven negotiation for safe re-optimization and migration. Simulations validated that neighbor-only dissemination induced manageable control-plane overhead, and that an SLM reasoning core specialized via verifier-based self-verification and periodic refinement maintained high decision quality, including under objective changes.

Looking forward, we outline several fundamental research directions for making agentic orchestration a principled and robust foundation for Organic 6G:

- **Formal cross-domain agentic orchestration:** develop formal models for cross-domain trade-offs and scalability limits (including domain-count thresholds at given performance targets); current simulations demonstrate feasibility up to 30 domains with near-linear overhead, but formal bounds remain open.
- **Uncertainty-aware decision theory:** design methods that represent uncertainty and provide tail-risk QoS guarantees under non-stationary dynamics (e.g., fluctuating link latency or intermittent compute availability at near-edge nodes).
- **Security for agentic control planes:** establish threat models and defenses for A2A-mediated orchestration (e.g., a malicious domain agent injecting false reachability advertisements to attract or divert traffic, confused-deputy behavior, or compromised tool responses), together with adversarial benchmarks.
- **Multi-agent stability and conflict resolution:** study agent-population conflicts and oscillations (e.g., two neighboring agents repeatedly migrating the same service instance back and forth, or a cascade of reallocations triggered by a single domain change), and design lightweight arbitration and convergence mechanisms.
- **Mechanism design for truthful coordination:** study incentive-compatible protocols for truthful resource advertisement (i.e., without incentive to overstate or understate available resources for strategic gain) and fair sharing, aligning local utilities with end-to-end QoS.
- **Learning and adaptation under real-time constraints:** quantify adaptability–cost trade-offs: safe online learning, energy/latency budgets, and privacy-preserving or federated adaptation in heterogeneous environments.

Acknolwedgement

This work was, in part, supported by BMFTR, Germany (6GEM+, Grant 16KIS2411), and the EU Horizon Europe programme (6G-Path, Grant 101139172).

References

[1] M. I. Corici, F. Eichhorn, R. Bless *et al.*, "Organic 6G Networks: Vision, Requirements, and Research Approaches," *IEEE Access*, vol. 11, pp. 70 698–70 715, 2023.

[2] S. Ghosh, H. Asgari, D. Hond *et al.*, "Future Open Networks Cross-Domain Cognitive Orchestration: A Novel Design Paradigm," *IEEE Access*, vol. 13, pp. 105 911–105 951, 2025.

[3] "Horizon Europe Call for Advanced Architectures Systems and Technologies," Accessed: Feb. 3, 2026. [Online]. Available: https://tinyurl.com/2wyjkap6

[4] D. Giannopoulos, G. Katsikas, K. Trantzas *et al.*, "ACROSS: Automated zero-touch cross-layer provisioning framework for 5G and beyond vertical services," in *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, Jun. 2023, pp. 735–740, iSSN: 2575-4912.

[5] N. Molner, L. Rosa, F. Risso *et al.*, "AIORA: An AI-Native Multi-Stakeholder Orchestration Architecture for 6G Continuum," *IEEE Network*, pp. 1–1, 2025.

[6] M. Dalgitsis, N. Cadenelli, M. A. Serrano *et al.*, "Cloud-Native Orchestration Framework for Network Slice Federation Across Administrative Domains in 5G/6G Mobile Networks," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 9306–9319, Jul. 2024.

[7] P. Benlloch-Caballero, A. Matencio-Escolar, J. Bernal Bernabe *et al.*, "E2E Network Slicing for Enhanced Cybersecurity, Orchestration, Automation and Response in 5G/6G: The RIGOUROUS Approach," *Journal of Network and Systems Management*, vol. 34, no. 1, p. 22, Nov. 2025.

[8] J. F. Santos, W. Liu, X. Jiao *et al.*, "Breaking down network slicing: Hierarchical orchestration of end-to-end networks," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 16–22, 2020.

[9] M. Shokrnezhad, H. Yu, T. Taleb *et al.*, "Toward a Dynamic Future With Adaptable Computing and Network Convergence (ACNC)," *IEEE Network*, vol. 39, no. 2, pp. 268–277, Mar. 2025.

[10] M. Farhoudi, M. Shokrnezhad, and T. Taleb, "Service Registration, Indexing, Discovery, and Selection: An Architectural Survey Toward a GenAI-Driven Future," *IEEE Access*, vol. 13, pp. 209 680–209 722, 2025.

[11] C. Wittner, "Communication Methods in Multi-Agent Reinforcement Learning," Jan. 2026, arXiv:2601.12886. [Online]. Available: http://arxiv.org/abs/2601.12886

[12] DeepSeek-AI, D. Guo, D. Yang *et al.*, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," Jan. 2025, arXiv:2501.12948. [Online]. Available: http://arxiv.org/abs/2501.12948

[13] T. Xie, Z. Gao, Q. Ren *et al.*, "Logic-RL: Unleashing LLM Reasoning with Rule-Based Reinforcement Learning," Feb. 2025, arXiv:2502.14768. [Online]. Available: http://arxiv.org/abs/2502.14768

[14] S.-Y. Liu, X. Dong, X. Lu *et al.*, "GDPO: Group reward-Decoupled Normalization Policy Optimization for Multi-reward RL Optimization," Jan. 2026, arXiv:2601.05242. [Online]. Available: http://arxiv.org/abs/2601.05242

[15] Q.-A. Dang and C. Ngo, "Reinforcement Learning for Reasoning in Small LLMs: What Works and What Doesn't," Mar. 2025, arXiv:2503.16219. [Online]. Available: http://arxiv.org/abs/2503.16219